



Kostenloses eBook

LERNEN

iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#iOS

Inhaltsverzeichnis

Über.....	1
Kapitel 1: Erste Schritte mit iOS.....	2
Bemerkungen.....	2
Anmerkungen.....	2
Verwandte Stack Overflow-Tags.....	2
Versionen.....	2
Examples.....	3
Standard-Single-View-Anwendung erstellen.....	3
Hallo Welt.....	11
Neues Projekt starten.....	11
Beschriftung hinzufügen.....	15
Code hinzufügen.....	17
App im Simulator ausführen.....	17
Weitermachen.....	18
Xcode-Schnittstelle.....	18
Navigator-Bereich.....	21
Die Herausgeber.....	21
Ressourcen und Elemente im Versorgungsbereich.....	24
Verwalten Sie Aufgaben mit der Arbeitsbereich-Symbolleiste.....	27
Erstellen Sie Ihr erstes Programm in Swift 3.....	28
Erstellen Sie Ihr erstes Programm.....	28
Kapitel 2: 3D Touch.....	34
Examples.....	34
3D Touch mit Swift.....	34
3 D Touch Objective-C-Beispiel.....	35
Kapitel 3: AFNetworking.....	37
Examples.....	37
Blockieren des Beendigungsblocks für einen benutzerdefinierten Thread.....	37
Kapitel 4: AirDrop.....	38

Examples.....	38
AirDrop.....	38
Kapitel 5: AirPrint-Tutorial in iOS.....	39
Examples.....	39
AirPrint-Druck Banner-Text.....	39
Kapitel 6: Alamofire.....	41
Syntax.....	41
Parameter.....	41
Examples.....	41
Eine Anfrage machen.....	41
Automatische Validierung.....	41
Antwortbehandlung.....	41
Manuelle Validierung.....	42
Antworthandler.....	42
Verkettete Antwort-Handler.....	42
Kapitel 7: Ändern der Größe von UIImage.....	43
Parameter.....	43
Examples.....	43
Ändern Sie die Größe eines Bildes nach Größe und Qualität.....	43
Kapitel 8: Antrag auf Antragsbewertung / Überprüfung.....	44
Einführung.....	44
Examples.....	44
Bewerten / Überprüfen Sie die iOS-Anwendung.....	44
Kapitel 9: AppDelegate.....	45
Einführung.....	45
Examples.....	45
Alle Zustände der Anwendung durch AppDelegate-Methoden.....	45
AppDelegate-Rollen:.....	46
Eine URL-spezifische Ressource öffnen.....	46
Umgang mit lokalen und Remote-Benachrichtigungen.....	47
Kapitel 10: App-Einreichungsprozess.....	49
Einführung.....	49

Examples.....	49
Einrichten von Bereitstellungsprofilen.....	49
Archivieren Sie den Code.....	49
IPA-Datei exportieren.....	51
Laden Sie die IPA-Datei mit dem Application Loader hoch.....	52
Kapitel 11: App-ID erstellen.....	54
Examples.....	54
In-App-Kaufprodukte erstellen.....	54
Sandbox-Benutzer erstellen.....	56
Kapitel 12: App-Transportsicherheit (ATS).....	57
Parameter.....	57
Bemerkungen.....	58
Examples.....	58
Laden Sie den gesamten HTTP-Inhalt.....	58
HTTP-Inhalt selektiv laden.....	59
Endpunkte erfordern SSL.....	59
Kapitel 13: App-weiten Operationen.....	61
Examples.....	61
Holen Sie sich den besten UIViewController.....	61
Systemereignisse abfangen.....	61
Kapitel 14: ARC (automatische Referenzzählung).....	62
Examples.....	62
Aktivieren / Deaktivieren von ARC für eine Datei.....	62
Kapitel 15: attributedText in UILabel.....	64
Einführung.....	64
Examples.....	64
HTML-Text in UILabel.....	64
Legen Sie eine andere Eigenschaft für Text in einem einzelnen UILabel fest.....	64
Kapitel 16: Auf Netzwerkverbindung prüfen.....	66
Bemerkungen.....	66
Vorsichtsmaßnahmen.....	66
Examples.....	66

Erreichbarkeitslistener erstellen	66
Beobachter zu Netzwerkänderungen hinzufügen	66
Warnung, wenn das Netzwerk nicht verfügbar ist	67
Warnung, wenn die Verbindung zu einem WIFI- oder Mobilfunknetz wird	67
Überprüfen Sie, ob eine Verbindung zum Netzwerk besteht	67
Kapitel 17: Automatisches Layout	69
Einführung	69
Syntax	69
Examples	69
Festlegen von Einschränkungen programmgesteuert	69
Pinning	69
Breite und Höhe	70
Im Behälter zentrieren	70
So verwenden Sie das automatische Layout	71
Zentrumsbeschränkungen	71
Raumansichten gleichmäßig	75
UILabel intrinsische Größe	77
Wie man mit Auto Layout animiert	88
Lösen Sie den UILabel-Prioritätskonflikt	90
UILabel & Parentview-Größe Gemäß Text in UILabel	93
Visual Format Language Basics: Einschränkungen im Code!	100
Gemischte Verwendung des automatischen Layouts mit nicht automatischem Layout	102
Proportionales Layout	102
NSLayoutConstraint: Constraints im Code!	104
Kapitel 18: AVPlayer und AVPlayerViewController	107
Bemerkungen	107
Examples	107
Wiedergabe von Medien mit AVPlayerViewController	107
Ziel c	107
Schnell	107
Wiedergabe von Medien mit AVPlayer und AVPlayerLayer	107

Ziel c.....	107
Schnell.....	108
AVPlayer-Beispiel.....	108
Kapitel 19: AVSpeechSynthesizer.....	109
Syntax.....	109
Parameter.....	109
Examples.....	109
Erstellen eines grundlegenden Textes zum Sprechen.....	109
Kapitel 20: AWS SDK.....	110
Examples.....	110
Laden Sie ein Bild oder ein Video mit AWS SDK in S3 hoch.....	110
Kapitel 21: Beacons mit CoreBluetooth konfigurieren.....	113
Einführung.....	113
Bemerkungen.....	113
Einige wichtige Punkte.....	113
Scannen Sie nach SERVICE UUID.....	113
So entdecken Sie SERVICE UUID ohne Dokumentation.....	113
Konvertieren Sie Daten in UInt16 und umgekehrt.....	114
Examples.....	114
Namen aller Bluetooth Low Energy (BLE) anzeigen.....	114
Verbinden Sie und lesen Sie den wichtigsten Wert.....	116
Schreibe den Hauptwert.....	117
Kapitel 22: Behandeln Sie mehrere Umgebungen mit Makro.....	119
Examples.....	119
Behandeln Sie mehrere Umgebungen mit mehreren Zielen und Makros.....	119
Kapitel 23: Benutzerdefinierte Auswahlmethoden für UITableViewCells.....	130
Einführung.....	130
Examples.....	130
Unterscheidung zwischen Einzel- und Doppelauswahl in Zeile.....	130
Kapitel 24: Benutzerdefinierte Auswahlmethoden für UITableViewCells.....	131
Examples.....	131

Unterscheidung zwischen Einzel- und Doppelauswahl in Zeile.....	131
Kapitel 25: Benutzerdefinierte Schriftarten.....	132
Examples.....	132
Einbetten benutzerdefinierter Schriftarten.....	132
Benutzerdefinierte Schriftarten mit Storyboard.....	133
UIKit + IBExtensions.h.....	133
UIKit + IBExtensions.m.....	134
Anwenden benutzerdefinierter Schriftarten auf Steuerelemente in einem Storyboard.....	135
Hinweise (Vorsichtsmaßnahmen).....	136
Gotchas (deux).....	136
Ergebnis.....	137
Proben.....	137
Umgang mit benutzerdefinierten Schriftarten.....	137
Umgehung der benutzerdefinierten Schriftart.....	137
Kapitel 26: Benutzerdefinierte Tastatur.....	139
Examples.....	139
Custom KeyBoard-Beispiel.....	139
Kapitel 27: Benutzerdefinierte UIViews aus XIB-Dateien.....	147
Bemerkungen.....	147
Examples.....	147
Verdrahtungselemente.....	147
Wie kann UIView mit XIB benutzerdefiniert wiederverwendbar werden?.....	168
Kapitel 28: Benutzerdefiniertes UITextField.....	170
Einführung.....	170
Examples.....	170
Benutzerdefiniertes UITextField zum Filtern von Eingabetext.....	170
Benutzerdefiniertes UITextField zum Deaktivieren aller Aktionen wie Kopieren, Einfügen usw.....	171
Kapitel 29: Bilder asynchron laden.....	172
Examples.....	172
Einfachster Weg.....	172
Überprüfen Sie, ob die Zelle nach dem Download noch sichtbar ist.....	172

Kapitel 30: Block	174
Syntax.....	174
Examples.....	174
UIView Animationen.....	174
Benutzerdefinierter Abschlussblock für benutzerdefinierte Methoden.....	174
Erfasste Variable ändern.....	175
Kapitel 31: CAAAnimation	176
Bemerkungen.....	176
Examples.....	176
Animieren Sie eine Ansicht von einer Position zur anderen.....	176
Ziel c	176
Schnell	176
Animierte Ansicht - Wurf.....	176
ZIEL C.....	176
SCHNELL.....	177
Ansicht drehen.....	177
Blick schütteln.....	177
Push-View-Animation.....	178
Ziel c.....	178
Schnell.....	178
Kapitel 32: CAGradientLayer	179
Syntax.....	179
Parameter.....	179
Bemerkungen.....	179
Examples.....	179
CAGradientLayer erstellen.....	179
CGGradientLayer mit mehreren Farben erstellen.....	180
Erstellen eines horizontalen CAGradientLayer.....	181
Erstellen eines horizontalen CAGradientLayer mit mehreren Farben.....	182
Animieren einer Farbänderung in CAGradientLayer.....	183
Kapitel 33: CALayer	185

Examples.....	185
Einen CALayer erstellen.....	185
Partikel mit CAEmitterLayer erstellen.....	185
Emitter-Ansicht mit benutzerdefiniertem Bild.....	186
So fügen Sie einen UIImage zu einem CALayer hinzu.....	187
Erscheinungsbild ändern.....	187
verbunden.....	191
Anmerkungen.....	191
Transformationen zu einem CALayer hinzufügen (übersetzen, drehen, skalieren).....	191
Grundlagen.....	191
Konfiguration.....	192
Übersetzen.....	193
Rahmen.....	194
Drehen.....	195
Mehrere Transformationen.....	195
Ein Hinweis zu Ankerpunkt und Position.....	196
Siehe auch.....	197
Animationen deaktivieren.....	197
Abgerundete Ecken.....	197
Schatten.....	197
Kapitel 34: Carthage iOS-Setup.....	199
Examples.....	199
Carthage Installation Mac.....	199
Kapitel 35: CAShapeLayer.....	200
Syntax.....	200
Bemerkungen.....	200
Examples.....	200
Grundlegende CAShapeLayer-Operation.....	200
Rechteck zeichnen.....	204
Kreis zeichnen.....	205
CAShapeLayer-Animation.....	205

Kapitel 36: CGContext-Referenz	207
Bemerkungen.....	207
Examples.....	207
Linie zeichnen.....	207
Text zeichnen.....	207
Kapitel 37: CLLocation	209
Examples.....	209
Entfernungsfilter mit.....	209
Abrufen des Benutzerstandorts mithilfe von CLLocationManager.....	209
Kapitel 38: CloudKit	212
Bemerkungen.....	212
Unterstützte Typen	212
Examples.....	212
App zur Verwendung mit CloudKit registrieren.....	212
CloudKit Dashboard verwenden.....	213
Datensatztypen	213
Daten in CloudKit speichern.....	213
Einen Aufnahmeschlüssel erstellen	213
Schnell.....	214
Aufnahme machen	214
Schnell.....	214
Ziel c.....	214
Hinweis	214
Zugriff auf den Container	214
Schnell.....	214
Speichern der Datensätze in der CloudKit-Datenbank	214
Schnell.....	215
Kapitel 39: Code-Signatur	216
Examples.....	216
Bereitstellungsprofile.....	216
Bereitstellungsprofiltypen	216

Entwicklung.....	216
Verteilung.....	216
Kapitel 40: Codierbar.....	217
Einführung.....	217
Examples.....	217
Verwendung von Codable mit JSONEncoder und JSONDecoder in Swift 4.....	217
Kapitel 41: Content Hugging / Inhaltskomprimierung im Autolayout.....	219
Bemerkungen.....	219
Examples.....	219
Definition: Intrinsische Inhaltsgröße.....	219
Kapitel 42: Core Graphics.....	221
Examples.....	221
Erstellen eines Kerngrafikkontexts.....	221
Core Graphics-Kontext.....	221
Einen Kontext machen.....	221
Schnell.....	221
Ziel c.....	221
Präsentieren der gezeichneten Leinwand für den Benutzer.....	222
Schnell.....	222
Ziel c.....	222
Kapitel 43: Core SpotLight in iOS.....	223
Examples.....	223
Core-Spotlight.....	223
Kapitel 44: CoreImage-Filter.....	233
Examples.....	233
Beispiel für den Kernbildfilter.....	233
Kapitel 45: CTCallCenter.....	239
Examples.....	239
Fangen Sie Anrufe von Ihrer App auch aus dem Hintergrund ab.....	239
CallKit - ios 10.....	240
Kapitel 46: CydiaSubstrate Tweak.....	242

Einführung	242
Bemerkungen	242
Theos installieren	242
Examples	242
Erstellen Sie einen neuen Tweak mit Theos	242
Verwenden Sie nic, um ein neues Projekt zu erstellen	242
Überschreibe die Methode zum Speichern von iOS-Screenshots	243
Kapitel 47: Daten zwischen View Controllern übergeben (mit MessageBox-Konzept)	244
Einführung	244
Examples	244
Einfache Beispielnutzung	244
Kapitel 48: Daten zwischen View-Controllern übergeben	245
Examples	245
Segmente verwenden (Daten weiterleiten)	245
Verwenden des Delegatenmusters (Zurückgeben von Daten)	246
Schnell	247
Ziel c.	248
Schnell	248
Ziel c.	249
Daten rückwärts mit "Abwickeln" umleiten	249
Weitergabe von Daten mithilfe von Closures (Zurückgeben von Daten)	250
Callback-Schließung (Block) verwenden, um Daten zurückzuleiten	251
Durch Zuweisen von Eigenschaften (Weiterleiten von Daten)	252
Kapitel 49: Debuggen von Abstürzen	254
Examples	254
Informationen zu einem Absturz suchen	254
Der rote Pfeil	255
Die Debugger-Konsole	255
Die Stapelverfolgung	255
Das Debuggen von SIGABRT und EXC_BAD_INSTRUCTION stürzt ab	256
Debuggen von EXC_BAD_ACCESS	256

Kapitel 50: Deep Linking in iOS	260
Bemerkungen	260
Examples	260
Öffnen einer App basierend auf ihrem URL-Schema	260
Hinzufügen eines URL-Schemas zu Ihrer eigenen App	260
Schritt 1: Registrieren Sie ein URL-Schema in Info.plist:	261
Schritt 2: Behandeln Sie die URL in der UIApplicationDelegate	261
Schritt 3: Führen Sie je nach URL eine Aufgabe aus.	262
Einrichten eines Deeplinks für Ihre App	262
Kapitel 51: DispatchGroup	265
Einführung	265
Examples	265
Einführung	265
Kapitel 52: Dynamischer Typ	269
Bemerkungen	269
Examples	269
Rufen Sie die aktuelle Inhaltsgröße ab	269
Schnell	269
Ziel c	269
Benachrichtigung zur Änderung der Textgröße	269
Schnell	269
Ziel c	269
Passende dynamische Schriftgröße in WKWebView	270
Schnell	270
Umgang mit der bevorzugten Änderung der Textgröße ohne Benachrichtigungen unter iOS 10	271
Schnell	271
Kapitel 53: Erstellen Sie ein benutzerdefiniertes Framework in iOS	272
Examples	272
Erstellen Sie ein Framework in Swift	272
Kapitel 54: Erstellen Sie ein Video aus Bildern	273

Einführung	273
Examples	273
Erstellen Sie ein Video aus UIImages	273
Kapitel 55: Erstellen Sie eine .ipa-Datei, die mit Apploadloader im Appstore hochgeladen w	276
Examples	276
Erstellen Sie eine .ipa-Datei, um die App mit dem Application Loader in den Appstore zu la	276
Kapitel 56: Erweiterung für umfassende Push-Benachrichtigung - iOS 10.....	282
Einführung	282
Examples	282
Benachrichtigungsinhalt-Erweiterung	282
Implementierung	282
Kapitel 57: EventKit	286
Examples	286
Erlaubnis beantragen	286
Schnell	286
Ziel c	286
Einen EKEventStore	286
Schnell	286
Ziel c	286
Hinweis	286
Verfügbarkeit überprüfen	286
Schnell	287
Ziel c	287
Erlaubnis beantragen	287
Schnell	287
Zugriff auf verschiedene Kalendertypen	287
Zugriff auf die Kalenderreihe	287
Schnell	288
Iteration durch Kalender	288
Schnell	288
Zugriff auf den Kalendertitel und die Farbe	288

Schnell.....	288
Ziel c.....	288
Ereignis hinzufügen.....	288
Ereignisobjekt erstellen.....	288
Schnell.....	288
Ziel c.....	289
Einstellen des zugehörigen Kalenders, Titels und Datums.....	289
Schnell.....	289
Ereignis zum Kalender hinzufügen.....	289
Schnell.....	289
Ziel c.....	289
Kapitel 58: FacebookSDK.....	290
Examples.....	290
FacebookSDK-Integration.....	290
Erstellen Sie Ihren eigenen benutzerdefinierten Button "Mit Facebook anmelden".....	292
Facebook-Benutzerdaten abrufen.....	293
Kapitel 59: Farbe der Statusleiste ändern.....	295
Examples.....	295
Für Nicht-UINavigationController-Statusleisten.....	295
Für UINavigationController-Statusleisten.....	295
Wenn Sie den Code von ViewController nicht ändern können.....	296
Zur ViewController-Eindämmung.....	296
Statusleistenstil für die gesamte Anwendung ändern.....	297
SCHNELL:.....	297
Schritt 1:.....	297
Schritt 2:.....	297
ZIEL C:.....	298
Kapitel 60: FCM Messaging in Swift.....	299
Bemerkungen.....	299
Examples.....	299
FCM in Swift initialisieren.....	299

Kapitel 61: FileHandle	301
Einführung	301
Examples	301
Lesen Sie die Datei aus dem Dokumentverzeichnis in Stücken	301
Kapitel 62: GameCenter-Bestenlisten	303
Examples	303
GameCenter-Bestenlisten	303
Kapitel 63: GameplayKit	306
Examples	306
Zufallszahlen generieren	306
Generation	306
Schnell	306
Ziel c	306
Schnell	306
Ziel c	306
Hinweis	307
Eine Zahl von 0 bis n erzeugen	307
Schnell	307
Ziel c	307
Eine Zahl von m bis n generieren	307
Schnell	307
Objective-C veraltet	307
Schnell	308
Objective-C veraltet	308
GKEntity und GKComponent	308
GKEntity	308
GKComponent	309
GKComponentSystem	309
Kapitel 64: GCD (Grand Central Dispatch)	311
Einführung	311
Examples	311

Erstellen Sie eine Versandwarteschlange.....	311
Die Hauptwarteschlange abrufen.....	311
Versandgruppe.....	312
Dispatch Semaphore.....	313
Serielle versus gleichzeitige Dispatch-Warteschlangen.....	314
Kapitel 65: Gesichtserkennung mit CoreImage / OpenCV.....	316
Examples.....	316
Gesichts- und Funktionserkennung.....	316
Kapitel 66: Graph (Coreplot).....	319
Examples.....	319
Erstellen von Diagrammen mit CorePlot.....	319
Kapitel 67: Größenklassen und Adaptivität.....	322
Bemerkungen.....	322
Examples.....	322
Trait-Sammlungen.....	322
Automatisches Layout mit Änderungen der Merkmalerfassung aktualisieren.....	323
Unterstützung für iOS Multitasking auf dem iPad.....	324
Kapitel 68: Größenklassen und Adaptivität.....	325
Bemerkungen.....	325
Examples.....	325
Größenklassen und Adaptivität durch Storyboard.....	325
Kapitel 69: Grundlegende Textdatei-E / A.....	328
Examples.....	328
Lesen und schreiben Sie aus dem Ordner Dokumente.....	328
Kapitel 70: Healthkit.....	330
Examples.....	330
HealthKit.....	330
Kapitel 71: Hintergrund festlegen.....	333
Examples.....	333
Hintergrund festlegen.....	333
Füllen Sie das Hintergrundbild einer UIView.....	333
Setze Hintergrund mit Bild.....	333

Erstellen einer Verlaufshintergrundansicht.....	333
Kapitel 72: Hintergrundmodi.....	335
Einführung.....	335
Examples.....	335
Aktivieren der Hintergrundmodus-Funktion.....	335
Hintergrundabruf.....	336
Schnell.....	336
Ziel c.....	337
Schnell.....	337
Testen des Hintergrundabrufs.....	337
Hintergrund-Audio.....	338
Kapitel 73: Hintergrundmodi und Ereignisse.....	339
Examples.....	339
Audio im Hintergrund abspielen.....	339
Kapitel 74: HINZUFÜGEN EINES SWIFT BRIDGING HEADER.....	341
Examples.....	341
So erstellen Sie einen Swift Bridging Header manuell.....	341
Xcode automatisch erstellen.....	341
Kapitel 75: iBeacon.....	343
Parameter.....	343
Bemerkungen.....	343
Examples.....	343
iBeacon-Grundbedienung.....	343
Scannen bestimmter Beacons.....	344
iBeacons in Reichweite.....	344
Kapitel 76: IBOutlets.....	345
Bemerkungen.....	345
Examples.....	345
Verwenden eines IBOutlets in einem UI-Element.....	345
Kapitel 77: In-App-Kauf.....	347
Examples.....	347

Einzelne IAP in Swift 2.....	347
In iTunesConnect einrichten.....	349
Die meisten grundlegenden Schritte zum Kaufen / Abonnieren eines Benutzers bei einem IAP.....	351
Kapitel 78: Initialisierungs-Idiome.....	352
Examples.....	352
Auf Tupel setzen, um Code-Wiederholung zu vermeiden.....	352
Initialisieren Sie mit Positionskonstanten.....	352
Attribute in didSet initialisieren.....	352
Gruppieren Sie Auslässe in einem benutzerdefinierten NSObject.....	353
Dann mit initialisieren.....	353
Werksmethode mit Block.....	354
Kapitel 79: Integration von SqlCipher.....	355
Einführung.....	355
Bemerkungen.....	355
Examples.....	358
Integration von Code:.....	358
Kapitel 80: iOS - Implementierung von XMPP mit dem Robbie Hanson-Framework.....	360
Examples.....	360
Robbie Hanson-Beispiel für iOS XMPP mit Openfire.....	360
SRXMPPDemo.....	360
Laden Sie das Beispiel und alle Klassen hier herunter - https://github.com/SahebRoy92/SRXM	360
Schritte zum folgen.....	360
Kapitel 81: iOS 10-Spracherkennungs-API.....	364
Examples.....	364
Sprache zu Text: Erkennen Sie Sprache aus einem Bündel mit Audioaufzeichnung.....	364
Kapitel 82: iOS Google Places-API.....	366
Examples.....	366
Orte in der Nähe vom aktuellen Standort abrufen.....	366
Kapitel 83: iOS TTS.....	368
Einführung.....	368
Examples.....	368

Text zu Sprache	368
Ziel c	368
Schnell	368
Hilfreiche Methoden	368
Kapitel 84: IOS-Version überprüfen	370
Examples	370
iOS 8 und höher	370
Versionen vergleichen	370
Ziel c	370
Swift 2.0 und höher	370
Geräte-iOS-Version	371
Ziel c	371
Schnell	371
Swift 3	371
Kapitel 85: Kategorien	372
Bemerkungen	372
Examples	372
Erstellen Sie eine Kategorie	372
Kapitel 86: Kernbewegung	376
Examples	376
Auf Barometer zugreifen, um relative Höhe zu erhalten	376
Kapitel 87: Kerndatei	377
Einführung	377
Examples	377
Operationen mit Kerndaten	377
Kapitel 88: Kernstandort	378
Syntax	378
Bemerkungen	378
Simulieren Sie einen Ort zur Laufzeit	378
Examples	379
Link CoreLocation Framework	379

Berechtigung zur Verwendung von Standortdiensten anfordern.....	380
Standortdienst-Berechtigung abrufen, während die App verwendet wird.....	380
Immer Standortdienst-Erlaubnis abrufen.....	381
Fügen Sie mithilfe der GPX-Datei einen eigenen benutzerdefinierten Speicherort hinzu.....	383
Standortdienste im Hintergrund.....	384
Kapitel 89: Kettenblöcke in einer Warteschlange (mit MKBlockQueue).....	386
Einführung.....	386
Examples.....	386
Beispielcode.....	386
Kapitel 90: Kontakte-Framework.....	388
Bemerkungen.....	388
Nützliche Links.....	388
Examples.....	388
Kontaktzugriff autorisieren.....	388
Framework importieren.....	388
Schnell.....	388
Ziel c.....	388
Zugänglichkeit prüfen.....	388
Schnell.....	388
Ziel c.....	388
Erlaubnis beantragen.....	389
Schnell.....	389
Zugriff auf Kontakte.....	389
Filter anwenden.....	389
Schnell.....	389
Ziel c.....	389
Schlüssel zum Abrufen angeben.....	389
Schnell.....	390
Kontakte abrufen.....	390
Schnell.....	390
Zugriff auf Kontaktdaten.....	390

Schnell	390
Kontakt hinzufügen	390
Schnell	390
Kapitel 91: Konvertieren Sie HTML in NSAttributedString-Zeichenfolge und umgekehrt	392
Examples	392
Ziel-C-Code zum Konvertieren einer HTML-Zeichenfolge in NSAttributedString und umgekehrt	392
Kapitel 92: Konvertieren Sie NSAttributedString in UIImage	393
Examples	393
NSAttributedString in UIImage-Konvertierung	393
Kapitel 93: Laufzeit in Objective-C	394
Examples	394
Zugeordnete Objekte verwenden	394
Kapitel 94: Leitfaden zur Auswahl der besten iOS-Architekturmuster	396
Einführung	396
Examples	396
MVC-Muster	396
MVP-Muster	396
MVVM-Muster	397
VIPER-Muster	398
Kapitel 95: Lokalisierung	401
Einführung	401
Examples	401
Lokalisierung in iOS	401
Kapitel 96: MeinLayout	402
Einführung	402
Examples	402
Eine einfache Demo zur Verwendung von MyLayout	402
Kapitel 97: Mitteilungen	404
Syntax	404
Parameter	404
Examples	404

Gerät für Push-Benachrichtigungen registrieren.....	404
Schnell.....	404
Ziel c.....	405
Schnell.....	406
Ziel c.....	407
Schnell.....	407
Ziel c.....	407
Schnell.....	408
Ziel c.....	408
Hinweis.....	408
Überprüfen Sie, ob Ihre App bereits für die Push-Benachrichtigung registriert ist.....	408
Schnell.....	408
Registrieren für die (nicht interaktive) Push-Benachrichtigung.....	408
Umgang mit Push-Benachrichtigung.....	409
Registrieren der App-ID für die Verwendung mit Push-Benachrichtigungen.....	411
Dinge, die du brauchst.....	411
Aktivieren des APNs-Zugriffs für die App-ID im Apple Developer Center.....	411
Aktivieren des APNs-Zugriffs in Xcode.....	412
Registrierung von Push-Benachrichtigungen aufheben.....	413
Ziel c.....	413
Schnell.....	413
Einstellen der Anwendungssymbol-Ausweisnummer.....	413
Testen von Push-Benachrichtigungen.....	414
Generierung eines .pem-Zertifikats aus Ihrer .cer-Datei, um sie an den Server-Entwickler w.....	415
Kapitel 98: MKDistanceFormatter.....	417
Examples.....	417
String aus der Ferne.....	417
Entfernungseinheiten.....	417
Einheitenstil.....	417
Kapitel 99: MKMapView.....	419
Examples.....	419

MKMapView hinzufügen.....	419
Kartentyp ändern.....	419
.Standard.....	419
Schnell 2.....	419
Swift 3.....	419
Ziel c.....	419
.Satellit.....	420
Schnell 2.....	420
Swift 3.....	420
Ziel c.....	421
.satelliteFlyover.....	421
Schnell 2.....	422
Swift 3.....	422
Ziel c.....	422
.hybrid.....	422
Schnell 2.....	422
Swift 3.....	422
Ziel c.....	422
.hybridFlyover.....	423
Schnell 2.....	423
Swift 3.....	423
Ziel c.....	424
Stellen Sie Zoom / Region für die Karte ein.....	424
Lokale Suchimplementierung mit MKLocalSearch.....	424
OpenStreetMap Tile-Overlay.....	424
UserLocation und UserTracking-Beispiel anzeigen.....	427
Ziel c.....	427
Schnell.....	427
Ziel c.....	428
Schnell.....	428
Pin / Punkt-Anmerkung auf der Karte hinzufügen.....	428

Simulieren Sie einen benutzerdefinierten Standort.....	428
Blättern Sie zu Koordinate und Zoom-Ebene.....	428
Mit Anmerkungen arbeiten.....	430
Passen Sie die Sichtbarkeit der Kartenansicht an, um alle Anmerkungen anzuzeigen.....	431
Kapitel 100: ModelPresentationStyles.....	432
Einführung.....	432
Bemerkungen.....	432
Examples.....	432
Erkundung von ModalPresentationStyle mit dem Interface Builder.....	432
Kapitel 101: Momentaufnahme von UIView.....	443
Examples.....	443
Schnappschuss abrufen.....	443
Schnappschuss mit Unteransicht mit anderem Markup und Text.....	443
Kapitel 102: MPMediaPickerDelegate.....	445
Bemerkungen.....	445
Examples.....	445
Laden Sie Musik mit MPMediaPickerControllerDelegate und spielen Sie sie mit AVAudioPlayer.....	445
Kapitel 103: MPVolumeView.....	447
Einführung.....	447
Bemerkungen.....	447
Examples.....	447
MPVolumeView hinzufügen.....	447
Kapitel 104: Multicast-Delegierte.....	448
Einführung.....	448
Examples.....	448
Multicast-Delegaten für alle Steuerelemente.....	448
Kapitel 105: MVP-Architektur.....	453
Einführung.....	453
Bemerkungen.....	453
Examples.....	454
Hundewechsel.....	454

DoggyView.swift	454
DoggyService.swift	455
DoggyPresenter.swift	455
DoggyListViewController.swift	456
Kapitel 106: MVVM	458
Examples	458
MVVM ohne reaktive Programmierung	458
Kapitel 107: Navigationsleiste	462
Examples	462
Anpassen der Standardansicht der Navigationsleiste	462
SWIFT-Beispiel	462
Kapitel 108: NSArray	463
Einführung	463
Bemerkungen	463
Examples	463
Array in Json-String umwandeln	463
Kapitel 109: NSAttributedString	464
Bemerkungen	464
Examples	464
Erstellen einer Zeichenfolge mit benutzerdefiniertem Kerning (Buchstabenabstand)	464
Erstellen Sie eine Zeichenfolge mit durchgestrichenem Text	464
Attributierte Zeichenfolgen und fetten Text in Swift anhängen	465
Ändern Sie die Farbe eines Wortes oder einer Zeichenfolge	465
Alle Attribute entfernen	466
Kapitel 110: NSBundle	467
Examples	467
Holen Sie sich das Hauptpaket	467
Bundle per Pfad erhalten	467
Kapitel 111: NSData	469
Bemerkungen	469
Nützliche Ressourcen	469
Examples	469

NSData-Objekte erstellen.....	469
Datei verwenden.....	469
Schnell.....	469
Ziel c.....	469
Verwenden eines String-Objekts.....	469
Schnell.....	469
Ziel c.....	469
Konvertierung von NSData in andere Typen.....	470
Zu String.....	470
Schnell.....	470
Ziel c.....	470
Zum Array.....	470
Schnell.....	470
Ziel c.....	470
In Byte-Array.....	470
Schnell.....	470
Ziel c.....	470
Konvertierung von NSData in HEX-String.....	470
Schnell.....	471
Ziel c.....	471
Kapitel 112: NSDate.....	472
Syntax.....	472
Bemerkungen.....	472
Examples.....	473
Aktuelles Datum abrufen.....	473
Schnell.....	474
Swift 3.....	474
Ziel c.....	474
NSDate-Objekt N Sekunden ab dem aktuellen Datum abrufen.....	474
Schnell.....	474
Swift 3.....	474

Ziel c.....	475
Datumsvergleich.....	475
Schnell.....	475
Ziel c.....	475
Schnell.....	475
Ziel c.....	475
Schnell.....	475
Ziel c.....	476
Swift 3.....	476
Holen Sie sich Unix-Epoche-Zeit.....	477
Schnell.....	477
Ziel c.....	477
NSDateFormatter.....	477
1. Erstellen Sie ein NSDateFormatter Objekt.....	478
Schnell.....	478
Swift 3.....	478
Ziel c.....	478
2. Legen Sie das Datumsformat fest, in dem Sie Ihre Zeichenfolge möchten.....	478
Schnell.....	478
Ziel c.....	478
3. Holen Sie sich die formatierte Zeichenfolge.....	478
Schnell.....	478
Swift 3.....	478
Ziel c.....	479
Hinweis.....	479
Nützliche Erweiterung zum Konvertieren des Datums in einen String.....	479
Konvertieren Sie NSDate, das (nur) aus Stunde und Minute besteht, in ein vollständiges NSD.....	479
Ziel c.....	479
UTC Zeitversatz von NSDate mit TimeZone.....	480
Zeitzyklusart abrufen (12 Stunden oder 24 Stunden).....	480
Prüfen, ob das aktuelle Datum das Symbol für AM oder PM enthält.....	480

Ziel c.....	480
Anforderung des Zeitzyklus-Typs von NSDateFormatter.....	480
Ziel c.....	481
Referenz.....	481
NSDate von JSON-Datumsformat abrufen "/ Date (1268123281843) /".....	481
Ziel c.....	481
Holen Sie sich historische Zeit von NSDate (zB: vor 5s, 2m, 3h).....	482
Ziel c.....	482
Kapitel 113: NSHTTPCookieStorage.....	483
Examples.....	483
Speichern und lesen Sie die Cookies von UserDefaults.....	483
Kapitel 114: NSInvocation.....	485
Examples.....	485
NSInvocation Objective-C.....	485
Kapitel 115: NSNotificationCenter.....	487
Einführung.....	487
Parameter.....	487
Bemerkungen.....	487
Examples.....	488
Beobachter hinzufügen.....	488
Namenskonvention.....	488
Schnell 2.3.....	488
Swift 3.....	488
Ziel c.....	488
Beobachter entfernen.....	489
Schnell 2.3.....	489
Swift 3.....	489
Ziel c.....	489
Benachrichtigung versenden.....	489
Schnell.....	489

Ziel c	489
Benachrichtigung mit Daten buchen.....	490
Schnell	490
Ziel c	490
Benachrichtigung beobachten.....	490
Schnell	490
Ziel c	490
Beobachter mit Block hinzufügen / entfernen.....	490
Beobachter für Namen hinzufügen und entfernen.....	491
Kapitel 116: NSPredicate	492
Syntax.....	492
Examples.....	492
Erstellen eines NSPredicate mit predicateWithBlock.....	492
Ziel c.....	492
Schnell.....	493
Erstellen eines NSPredicate mit predicateWithFormat.....	493
Ziel c.....	493
Schnell.....	493
Ein NSPredicate mit Substitutionsvariablen erstellen.....	493
Ziel c.....	493
Schnell.....	493
Verwenden von NSPredicate zum Filtern eines Arrays.....	494
Ziel c.....	494
Schnell.....	494
Formularvalidierung mit NSPredicate.....	494
NSPredicate mit "AND", "OR" und "NOT" -Bedingung.....	496
Ziel c	496
UND - Bedingung.....	496
ODER - Bedingung.....	496
NICHT - Bedingung.....	496
Kapitel 117: NSTimer	498

Parameter.....	498
Bemerkungen.....	498
Examples.....	498
Timer erstellen.....	498
Einen Timer manuell auslösen.....	499
Timer ungültig machen.....	499
Timer-Frequenzoptionen.....	500
Wiederholtes Timer-Ereignis.....	500
Nicht wiederholtes verzögertes Timer-Ereignis.....	500
Weitergabe von Daten mit Timer.....	501
Kapitel 118: NSURL	502
Examples.....	502
So erhalten Sie die letzte Zeichenfolgekomponente aus NSURL-Zeichenfolge.....	502
So erhalten Sie die letzte Zeichenfolgekomponente von der URL (NSURL) in Swift.....	502
Kapitel 119: NSURLConnection	503
Examples.....	503
Methoden delegieren.....	503
Synchrone Anforderung.....	503
Asynchrone Anforderung.....	504
Kapitel 120: NSURLSession	505
Bemerkungen.....	505
Examples.....	506
Einfache GET-Anfrage.....	506
Ziel-C Erstellen Sie eine Session- und Datenaufgabe.....	507
Hintergrundkonfiguration einrichten.....	507
Senden einer POST-Anforderung mit Argumenten mithilfe von NSURLSession in Objective-C.....	508
Kapitel 121: NSUserActivity	514
Einführung.....	514
Bemerkungen.....	514
Aktivitätsarten	514
Aktive Aktivität einstellen / kündigen	514

Suchindizierung	514
Examples	514
NSUserActivity erstellen	514
Kapitel 122: UserDefaults	516
Syntax	516
Bemerkungen	516
Examples	516
Einstellwerte	516
Schnell <3	516
Swift 3	516
Ziel c	516
Schnell <3	517
Swift 3	517
Ziel c	517
Benutzerdefinierte Objekte	517
Schnell	517
Ziel c	517
Standardwerte abrufen	518
Schnell	518
Ziel c	518
Schnell	518
Ziel c	518
Werte speichern	518
Schnell	519
Ziel c	519
Verwenden Sie Manager zum Speichern und Lesen von Daten	519
Schnell	519
Ziel c	520
Hinweis	520
NSUserDefaults löschen	520
Schnell	521

Ziel c.....	521
UserDefaults wird in Swift 3 verwendet.....	521
Kapitel 123: Objective-C Zugeordnete Objekte.....	523
Einführung.....	523
Syntax.....	523
Parameter.....	523
Bemerkungen.....	523
Examples.....	524
Beispiel für ein assoziiertes Basisobjekt.....	524
Kapitel 124: Online-Bilder zwischenspeichern.....	525
Examples.....	525
AlamofireImage.....	525
Kapitel 125: OpenGL.....	526
Einführung.....	526
Examples.....	526
Beispielprojekt.....	526
Kapitel 126: Parallelität.....	527
Einführung.....	527
Syntax.....	527
Parameter.....	527
Bemerkungen.....	528
Examples.....	528
Gleichzeitiger Ausführen von Code - Ausführen von Code, während anderer Code ausgeführt wi.....	528
Ausführung im Hauptthread.....	528
Versandgruppe - Warten auf andere Threads abgeschlossen.....	529
Kapitel 127: PDF-Erstellung in iOS.....	530
Examples.....	530
PDF erzeugen.....	530
PDF anzeigen.....	531
Mehrseitiges PDF.....	532
Erstellen Sie ein PDF aus einem beliebigen in UIWebView geladenen Microsoft-Dokument.....	532

Kapitel 128: plist iOS	534
Einführung	534
Examples	534
Beispiel:	534
Speichern und Bearbeiten / Löschen von Daten in Plist	539
Kapitel 129: Profil mit Instrumenten	541
Einführung	541
Examples	541
Zeitprofiler	541
Kapitel 130: QR Code Scanner	554
Einführung	554
Examples	554
UIViewController sucht nach QR und zeigt den Videoeingang an	554
QR-Code mit dem AVFoudation-Framework scannen	555
Schritt 1	555
Schritt 2	556
Schritt 3	556
Kapitel 131: Reich	558
Bemerkungen	558
Examples	558
RLMObject-Basismodellklasse mit Primärschlüssel - Objective-C	558
Kapitel 132: Rich Benachrichtigungen	559
Einführung	559
Examples	559
Erstellen einer einfachen UNNotificationContentExtension	559
Kapitel 133: Safari-Dienstleistungen	568
Examples	568
Implementieren Sie SFSafariViewControllerDelegate	568
Elemente zur Safari-Leseliste hinzufügen	568
Öffnen Sie eine URL mit SafariViewController	569
Kapitel 134: Schlüsselbund	570

Syntax.....	570
Bemerkungen.....	570
Examples.....	570
Passwort zum Schlüsselbund hinzufügen.....	570
Schnell	571
Schnell	571
Schnell	571
Schnell	571
Schnell	571
Schnell	571
Schnell	572
Ein Passwort im Schlüsselbund finden.....	572
Schnell	572
Schnell	572
Schnell	572
Schnell	573
Aktualisieren eines Passworts im Schlüsselbund.....	573
Schnell	573
Schnell	573
Schnell	574
Schnell	574
Passwort aus dem Schlüsselbund entfernen.....	574
Schnell	574
Schnell	574
Keychain Hinzufügen, Aktualisieren, Entfernen und Suchen mit einer Datei.....	575
Schlüsselbund-Zugriffskontrolle (TouchID mit Passwort-Fallback).....	577
Schnell	577
Schnell	578
Schnell	578
Schnell	578
Schnell	579

Kapitel 135: Schlüsselwert-Schlüsselwertbeobachtung	580
Bemerkungen	580
Examples	580
Verwendung des Kontextes für die KVO-Beobachtung	580
Beobachten einer Eigenschaft einer NSObject-Unterklasse	581
Kapitel 136: Schneiden Sie einen UIImage in einen Kreis	582
Examples	582
Schneiden Sie ein Bild in einen Kreis - Ziel C	582
SWIFT 3 Beispiel	583
Kapitel 137: Schnelle und Objective-C-Interoperabilität	585
Examples	585
Verwenden von Objective-C-Klassen in Swift	585
Schritt 1: Fügen Sie die Objective-C-Implementierung hinzu	585
Schritt 2: Fügen Sie den Bridging-Header hinzu	585
Schritt 3: Objective-C Header hinzufügen - .h	587
Schritt 4: Erstellen Sie Ihre Objective-C-Klasse	587
Schritt 5: Klasse zum Bridging-Header hinzufügen	587
Schritt 6: Verwenden Sie Ihr Objekt	587
Verwendung schneller Klassen in Objective-C	587
Schritt 1: Neue Swift-Klasse erstellen	587
Schritt 2: Importieren Sie Swift-Dateien in die ObjC-Klasse	588
Schritt 3: Verwenden Sie Ihre Klasse	588
Hinweis:	588
Kapitel 138: Segues	590
Examples	590
Ein Überblick	590
Vorbereiten des View Controllers vor dem Auslösen eines Segue	590
PrepareForSegue :	590
Parameter	590
Beispiel in Swift	591
Entscheidung, ob eine aufgerufene Segue ausgeführt werden soll	591
ShouldPerformSegueWithIdentifier :	591

Parameter.....	591
Beispiel in Swift.....	591
Verwenden von Segues, um im Navigationsstapel rückwärts zu navigieren.....	591
Programmgesteuert auslösen.....	592
PerformSegueWithIdentifier:.....	592
Parameter.....	592
Beispiel in Swift.....	592
Kapitel 139: Selektive UIView-Ecken abrunden.....	593
Examples.....	593
Ziel-C-Code, um die ausgewählte Ecke eines UIView abzurunden.....	593
Kapitel 140: Sicherheit.....	594
Einführung.....	594
Examples.....	594
Transportsicherheit mit SSL.....	594
Sichern von Daten in iTunes-Backups.....	595
Kapitel 141: Simulator.....	597
Einführung.....	597
Bemerkungen.....	597
Verschiedene Arten von Simulatoren.....	597
Hilfe bekommen.....	597
Examples.....	598
Simulator starten.....	598
3D / Force Touch-Simulation.....	598
Gerätemodell ändern.....	598
Im Simulator navigieren.....	598
Home "Button".....	598
Sperrern.....	598
Drehung.....	598
Kapitel 142: Simulator-Builds.....	599
Einführung.....	599
Examples.....	599

Build manuell auf dem Simulator installieren.....	599
Kapitel 143: SiriKit.....	600
Bemerkungen.....	600
Verschiedene Arten von Siri-Anfragen.....	600
Examples.....	600
Hinzufügen der Siri-Erweiterung zur App.....	600
Fähigkeit hinzufügen.....	600
Erweiterung hinzufügen.....	600
Laut Apple:.....	601
Hinweis.....	601
Hinweis.....	601
Kapitel 144: SLComposeViewController.....	603
Examples.....	603
SLComposeViewController für Twitter, Facebook, SinaWeibo und TencentWeibo.....	603
Kapitel 145: Standort mit GPX-Dateien simulieren iOS.....	605
Examples.....	605
Ihre .gpx-Datei: MPS_HQ.gpx.....	605
Um diesen Ort festzulegen:.....	605
Kapitel 146: StoreKit.....	607
Examples.....	607
Erhalten Sie lokalisierte Produktinformationen aus dem App Store.....	607
Kapitel 147: Storyboard.....	608
Einführung.....	608
Examples.....	608
Initialisieren.....	608
Initial ViewController abrufen.....	608
ViewController abrufen.....	608
Kapitel 148: Swift: Ändern des rootViewControllers in AppDelegate, um den Haupt- oder Logi.	609
Einführung.....	609
Bemerkungen.....	609
Ansätze:.....	609

Examples.....	610
Option 1: Den Root-View-Controller austauschen (gut).....	610
Option 2: Alternativen Fluss modal (besser) präsentieren.....	610
Kapitel 149: SWRevealViewController.....	612
Bemerkungen.....	612
Examples.....	612
Einrichten einer Basis-App mit SWRevealViewController.....	612
Kapitel 150: Tastatur verwalten.....	617
Examples.....	617
Scrollen einer UIScrollView / UITableView bei Anzeige der Tastatur.....	617
Schließen Sie eine Tastatur mit Tipp auf Anzeige ab.....	618
Erstellen Sie eine benutzerdefinierte In-App-Tastatur.....	619
Erstellen Sie die .xib-Tastaturlayoutdatei.....	619
Erstellen Sie die .swift UIView-Unterklassen-Tastaturdatei.....	620
Richten Sie den View Controller ein.....	622
Häufiger Fehler.....	623
Anmerkungen.....	624
Verwalten der Tastatur mit einem Singleton + -Delegierten.....	624
Ansicht nach oben oder unten verschieben, wenn Tastatur vorhanden ist.....	627
Hinweis: Dies funktioniert nur für die von iOS bereitgestellte integrierte Tastatur.....	627
SCHNELL:.....	627
ZIEL C:.....	628
Kapitel 151: Überholspur.....	629
Examples.....	629
Fastlane-Werkzeuge.....	629
Installieren Sie Fastlane.....	629
iOS-Tools.....	629
iOS TestFlight-Tools.....	629
Android-Tools.....	630
Kapitel 152: UIA Auftritt.....	631
Examples.....	631

Legt das Aussehen aller Instanzen der Klasse fest	631
Aussehen für die Klasse, wenn sie in der Containerklasse enthalten ist	632
Kapitel 153: UINavigationController	634
Parameter	634
Examples	634
Activity View Controller initialisieren	634
Ziel c	634
Schnell	634
Kapitel 154: UIAlertController	635
Bemerkungen	635
Examples	635
AlertViews mit UIAlertController	635
Vorübergehendes toastähnliches Popup	637
Schnell	637
Textfeld in UIAlertController wie eine Eingabeaufforderung hinzufügen	637
Schnell	637
Ziel c	637
Aktionsblätter mit UIAlertController	638
Einfaches Aktionsblatt mit zwei Tasten	638
Schnell	638
Ziel c	638
Schnell	638
Ziel c	639
Schnell	639
Ziel c	639
Schnell	639
Ziel c	639
Aktionsblatt mit zerstörerischem Button	640
Schnell	640
Ziel c	641
Anzeigen und Verarbeiten von Warnungen	641

Ein Knopf	641
Schnell.....	641
Zwei Knöpfe	642
Schnell.....	642
Drei Knöpfe	643
Schnell.....	643
Handhabung von Tastenanschlägen	644
Schnell.....	644
Anmerkungen	644
Eine Aktionsschaltfläche hervorheben.....	644
Kapitel 155: UIBarButtonItem	646
Parameter.....	646
Bemerkungen.....	646
Examples.....	646
Erstellen eines UIBarButtonItem.....	646
Erstellen eines UIBarButtonItem im Interface Builder.....	646
Fügen Sie Ihrem Storyboard einen Navigations-Controller hinzu	646
Fügen Sie ein Bar-Schaltflächenelement hinzu	647
Legen Sie die Attribute fest	648
Fügen Sie eine IB-Aktion hinzu	649
Anmerkungen	650
Bar Button Element Originalbild ohne Farbton.....	650
Kapitel 156: UIBezierPath	651
Examples.....	651
Wie wird ein Eckenradius auf von UIBezierPath gezeichnete Rechtecke angewendet?.....	651
So erstellen Sie einfache Formen mit UIBezierPath.....	653
UIBezierPath + AutoLayout.....	655
So wenden Sie Schatten auf UIBezierPath an.....	656
Entwerfen und Zeichnen eines Bezierpfads.....	657
So zeichnen Sie einen Bézier-Pfad in einer benutzerdefinierten Ansicht	657
Umriss der Form gestalten	658

Teilen Sie den Pfad in Segmente	658
Bauen Sie den Pfad programmgesteuert auf	659
Zeichne den Pfad	661
Weitere Studie	663
Anmerkungen	664
Kreisansicht und Spaltenansicht mit UIBezierPath	664
Kapitel 157: UIButton	667
Einführung	667
Bemerkungen	667
Schaltflächentypen	667
Examples	668
Erstellen eines UIButton	668
Titel einstellen	668
Titelfarbe einstellen	669
Inhalte horizontal ausrichten	669
Titelbezeichnung abrufen	670
Deaktivieren eines UIButton	670
Hinzufügen einer Aktion zu einem UIButton über Code (programmgesteuert)	671
Schriftart einstellen	671
Anhängen einer Methode an eine Schaltfläche	671
Ermitteln Sie die Größe von UIButton streng nach Text und Schriftart	672
Bild einstellen	672
Schnell	672
Ziel c	673
Mehrere Kontrollzustände	673
Schnell	673
Ziel c	673
Kapitel 158: UICollectionView	674
Examples	674
Erstellen Sie eine Sammlungsansicht programmgesteuert	674
Swift - UICollectionViewDelegateFlowLayout	674

Erstellen Sie eine UICollectionView	674
UICollectionView - Datenquelle	675
Einfaches Swift-Beispiel für eine Sammlungsansicht	676
Erstellen Sie ein neues Projekt	676
Fügen Sie den Code hinzu	676
Richten Sie das Storyboard ein	677
Schließen Sie die Auslässe an	679
Fertig	680
Verbesserungen vornehmen	681
Weitere Studie	682
Stapelaktualisierungen durchführen	682
UICollectionViewDelegate-Setup und Elementauswahl	683
Verwalten Sie die Ansicht für mehrere Sammlungen mit DataSource und Flowlayout	684
Kapitel 159: UIColor	687
Examples	687
Erstellen eines UIColor	687
Undokumentierte Methoden	688
styleTypeString	688
_systemDestructiveTintColor()	689
Farbe mit Alpha-Komponente	690
Schnell	690
Swift 3	690
Ziel c	690
Benutzerdefinierte Attribute auf den UIColor-Datentyp anwenden lassen	690
Das neue benutzerdefinierte Attribut (borderUIColor) wird problemlos erkannt und angewende	690
Erstellen eines UIColor aus Hexadezimalzahl oder Zeichenfolge	691
Angepasste Helligkeit der Farbe von UIColor	693
UIColor aus einem Bildmuster	694
Hellerer und dunklerer Farbton einer bestimmten UIColor	695
Kapitel 160: UIControl - Ereignisbehandlung mit Blöcken	697
Examples	697

Einführung.....	697
Kapitel 161: UIDatePicker.....	701
Bemerkungen.....	701
Examples.....	701
Erstellen Sie eine Datumsauswahl.....	701
Schnell.....	701
Ziel c.....	701
Minimum-Maximum-Datum einstellen.....	701
Mindestdatum.....	701
Maximales Datum.....	701
Modi.....	701
Minutenintervall einstellen.....	702
Countdown-Dauer.....	702
Kapitel 162: UIDevice.....	703
Parameter.....	703
Bemerkungen.....	703
Examples.....	703
Abrufen des Modellnamens des iOS-Geräts.....	703
Akkustatus und Akkuladestand abrufen.....	705
Gerät identifizieren und bedienen.....	705
Die Ausrichtung des Geräts ermitteln.....	706
Den Batteriezustand des Geräts ermitteln.....	707
Verwendung des Näherungssensors.....	708
Kapitel 163: UIFeedbackGenerator.....	709
Einführung.....	709
Examples.....	709
Trigger Impact Haptic.....	709
Schnell.....	709
Ziel c.....	710
Kapitel 164: UIFont.....	711
Einführung.....	711
Examples.....	711

UIFont deklarieren und initialisieren	711
Ändern der Schriftart eines Labels	711
Kapitel 165: UIGestureRecognizer	712
Examples	712
UITapGestureRecognizer	712
UIPanGestureRecognizer	713
UITapGestureRecognizer (Doppeltipp)	714
Anmerkungen	714
UILongPressGestureRecognizer	714
Anmerkungen	715
UISwipeGestureRecognizer	715
Anmerkungen	716
UIPinchGestureRecognizer	716
Anmerkungen	717
UIRotationGestureRecognizer	717
Anmerkungen	717
Hinzufügen einer Gestenerkennung im Interface Builder	717
Anmerkungen	719
Kapitel 166: UIImage	720
Bemerkungen	720
Examples	720
UIImage erstellen	720
Mit lokalem Image	720
Schnell	720
Ziel c.	720
Hinweis	720
Mit NSData	720
Schnell	720
Mit UIColor	720
Schnell	721
Ziel c.	721

Mit Dateiinhalt	721
Ziel c	721
Bildobjekte mit Dateiinhalt erstellen und initialisieren.....	722
Veränderbare Bild mit Kappen.....	722
Bilder vergleichen.....	723
Schnell.....	723
Ziel c.....	723
Erstellen Sie UIImage mit UIColor.....	723
Schnell	723
Swift 3	724
Ziel c:	724
Farbverlauf mit Farben.....	724
Hintergrundfarbe für Farbverläufe.....	725
Konvertieren Sie UIImage in / aus der base64-Kodierung.....	725
Machen Sie eine Momentaufnahme einer UIView.....	725
Wenden Sie UIColor auf UIImage an.....	726
Ändern Sie die UIImage-Farbe.....	726
Kapitel 167: UIImagePickerController	728
Einführung.....	728
Examples.....	728
Allgemeine Verwendung von UIImagePickerController.....	728
Kapitel 168: UIImageView	730
Examples.....	730
Erstellen Sie eine UIImageView.....	730
Zuweisen eines Bildes zu einer UIImageView.....	730
Animieren einer UIImageView.....	731
Bild zu einem Kreis oder abgerundet machen.....	731
Ziel c.....	732
Schnell.....	732
UIImage mit Label maskiert.....	733
Ziel c.....	733

Swift 3.....	733
Ändern Sie die Farbe eines Bildes.....	733
Wie wirkt sich die Mode-Eigenschaft auf ein Bild aus?.....	733
Zum Füllen skalieren.....	734
Aspect Fit.....	735
Aspekt füllen.....	735
Neu zeichnen.....	736
Center.....	736
oben.....	737
Unterseite.....	737
Links.....	738
Recht.....	738
Oben links.....	739
Oben rechts.....	739
Unten links.....	739
Unten rechts.....	740
Anmerkungen.....	740
Kapitel 169: UIKit Dynamics.....	742
Einführung.....	742
Bemerkungen.....	742
Schnell.....	742
Ziel c.....	742
Examples.....	743
Das fallende Quadrat.....	743
Schnelle Ansicht basierend auf Gestengeschwindigkeit.....	744
Schnell.....	744
Ziel c.....	746
"Sticky Corners" -Effekt mit UIFieldBehaviors.....	748
Schnell.....	748
Ziel c.....	750

UIDynamicBehavior-gesteuerte benutzerdefinierte Überblendung	753
Schnell	753
Ziel c	754
Schnell	755
Ziel c	755
Schnell	757
Ziel c	760
Schattenübergang mit realer Physik unter Verwendung von UIDynamicBehaviors	764
Schnell	765
Ziel c	766
Schnell	767
Ziel c	768
Schnell	768
Ziel c	772
Positionieren Sie dynamische Animationspositionen in Grenzen	777
Schnell	778
Ziel c	778
Schnell	778
Ziel c	779
Schnell	780
Ziel c	781
Kapitel 170: UIKit Dynamics mit UICollectionView	783
Einführung	783
Examples	783
Erstellen eines benutzerdefinierten Ziehverhaltens mit UIDynamicAnimator	783
Schnell	784
Ziel c	785
Schnell	786
Ziel c	787
Schnell	787
Ziel c	789

Schnell.....	790
Ziel c.....	792
Kapitel 171: UILabel.....	795
Einführung.....	795
Syntax.....	795
Bemerkungen.....	795
Examples.....	795
Text in einem vorhandenen Etikett ändern.....	795
Festlegen des Textes mit String Literalen.....	796
Den Text mit einer Variablen einstellen.....	796
Textfarbe.....	796
Textfarbe auf einen Teil des Textes anwenden.....	797
Textausrichtung.....	797
Erstellen Sie ein UILabel.....	798
Mit einem Rahmen.....	798
Schnell.....	798
Ziel c.....	798
Mit automatischem Layout.....	798
Schnell.....	798
Ziel c.....	799
Mit Objective-c + Visual Format Language (VFL).....	799
Mit dem Interface Builder.....	799
Verknüpfung zwischen Interface Builder und View Controller.....	800
Schnell.....	800
Ziel c.....	801
Schriftart einstellen.....	801
Schnell.....	801
Ziel c.....	801
Ändern Sie die Größe der Standardschriftart.....	801
Schnell.....	801
Swift 3.....	801

Ziel c.....	801
Verwenden Sie eine bestimmte Schriftstärke.....	801
Schnell.....	801
Swift3.....	802
Ziel c.....	802
Schnell.....	802
Swift3.....	802
Ziel c.....	802
Verwenden Sie einen dynamischen Textstil.....	802
Schnell.....	802
Swift 3.....	802
Ziel c.....	802
Verwenden Sie eine andere Schriftart.....	803
Schnell.....	803
Ziel c.....	803
Überschreibt die Schriftgröße.....	803
Schnell.....	803
Swift 3.....	803
Ziel c.....	803
Verwenden Sie Custom Font Swift.....	803
Anzahl der Zeilen.....	803
Den Wert programmgesteuert einstellen.....	804
Schnell.....	804
Ziel c.....	804
Hinweis.....	804
Schnell.....	804
Ziel c.....	804
Hinweis.....	804
Hinweis.....	804
Wert im Interface Builder einstellen.....	805

Größe passend.....	805
Hintergrundfarbe.....	807
Fügen Sie dem Text Schatten hinzu.....	808
Variable Höhe mit Einschränkungen.....	808
Schnell.....	809
Schnell.....	809
LineBreakMode.....	809
Code verwenden.....	809
Schnell.....	809
Swift 3.....	809
Ziel c.....	810
Storyboard verwenden.....	810
Konstanten.....	810
Content Bounds berechnen (zB für dynamische Zellenhöhen).....	811
Klickbare Beschriftung.....	813
Schnell.....	813
Ziel c.....	813
Festlegen von "userInteractionEnabled" im Attribute-Inspector des Storyboards.....	813
Dynamischer Etikettenrahmen aus unbekannter Textlänge.....	814
Ziel c.....	814
Schnell.....	814
Beschriftung mit Text.....	815
Begründen Sie den Text.....	822
Beschriftung mit automatischer Größe für den Text.....	823
Befestigen Sie die linke und obere Kante.....	823
Anmerkungen.....	823
Ermitteln Sie die Größe von UILabel streng nach Text und Schriftart.....	824
Hervorgehobene und hervorgehobene Textfarbe.....	825
Kapitel 172: UILabel-Text unterstrichen.....	826
Examples.....	826
Unterstrichen eines Textes in einem UILabel mit Objective C.....	826

Einen Text in UILabel mit Swift unterstreichen.....	826
Kapitel 173: UILocalNotification.....	827
Einführung.....	827
Bemerkungen.....	827
Examples.....	827
Lokale Benachrichtigung planen.....	827
Registrierung für lokale Benachrichtigungen.....	828
Antwort auf empfangene lokale Benachrichtigung.....	829
Lokale Benachrichtigungen mit UUID verwalten.....	829
Eine Benachrichtigung verfolgen.....	829
Eine Benachrichtigung abbrechen.....	830
Eine lokale Benachrichtigung wird sofort angezeigt.....	830
Benachrichtigungston.....	831
Registrieren und Planen der lokalen Benachrichtigung in Swift 3.0 (iOS 10).....	831
Was ist neu in UILocalNotification mit iOS10?.....	832
Kapitel 174: UINavigationController.....	836
Bemerkungen.....	836
Examples.....	836
In einem Navigations-Controller einblenden.....	836
Zum vorherigen Ansichtscontroller.....	836
Zum Root-View-Controller.....	836
Einen Navigationscontroller erstellen.....	837
Betten Sie einen View-Controller programmgesteuert in einen Navigations-Controller ein.....	837
Einen View Controller auf den Navigationsstapel schieben.....	837
Zweck.....	838
Kapitel 175: UIPageViewController.....	839
Einführung.....	839
Syntax.....	839
Bemerkungen.....	839
Examples.....	839
Erstellen Sie programmgesteuert einen horizontalen Paging-UIPageViewController.....	839

Eine einfache Möglichkeit zum Erstellen horizontaler Seitenansichts-Controller (unendliche	841
Kapitel 176: UIPheonix - einfaches, flexibles, dynamisches und hoch skalierbares UI-Framew	845
Einführung	845
Bemerkungen	845
Examples	845
Beispiel-UI-Komponenten	845
Verwendungsbeispiel	846
Kapitel 177: UIPickerView	848
Examples	848
Grundlegendes Beispiel	848
Schnell	848
Ziel c	848
PickerView-Hintergrundfarbe und Textfarbe ändern	849
Kapitel 178: UIRefreshControl TableView	850
Einführung	850
Examples	850
Ziel-C-Beispiel	850
RefreshControl für tableView einrichten:	851
Kapitel 179: UIScrollView	852
Examples	852
Erstellen Sie eine UIScrollView	852
Blättern Sie in die Inhaltsgröße	852
ScrollView mit AutoLayout	852
Bildlauf mit aktiviertem automatischen Layout	858
Schlüssel Konzepte	859
Starten Sie ein neues Projekt	859
Storyboard	859
Fertig	862
Weitere Studie	862
Bildlauf aktivieren / deaktivieren	862
Vergrößern / Verkleinern von UIImageView	863

Erstellen Sie nun die UIImageView-Instanz	863
Ermitteln, wann der Bildlauf von UIScrollView mit Delegat-Methoden abgeschlossen ist.....	864
Ziel c:.....	864
Schnell:.....	864
Bildlaufrichtung einschränken.....	865
Kapitel 180: UIScrollView AutoLayout	866
Examples.....	866
ScrollableController.....	866
UIScrollView dynamische Inhaltsgröße über das Storyboard.....	869
Kapitel 181: UIScrollView mit StackView-Kind	872
Examples.....	872
Eine komplexe StackView in Scrollview Example.....	872
Mehrdeutiges Layout vermeiden.....	873
In verschachtelten StackViews zum Inhalt blättern.....	874
Kapitel 182: UISearchController	875
Syntax.....	875
Parameter.....	875
Bemerkungen.....	876
Examples.....	876
Suchleiste im Titel der Navigationsleiste.....	876
Suchleiste im Header der Tabellenansicht.....	879
Implementierung.....	880
UISerachController in Objective-C.....	881
Kapitel 183: UISegmentedControl	882
Einführung.....	882
Examples.....	882
UISegmentedControl über Code erstellen.....	882
Kapitel 184: UISlider	883
Examples.....	883
UISlider.....	883
SWIFT-Beispiel.....	883
Hinzufügen eines benutzerdefinierten Daumens.....	884

Kapitel 185: UISplitViewController	885
Bemerkungen.....	885
Examples.....	885
Master- und Detailansicht-Interaktion mit Delegates in Ziel C.....	885
Kapitel 186: UISplitViewController	894
Bemerkungen.....	894
Examples.....	894
Interaktion zwischen Master- und Detailansicht mithilfe von Delegierten in Ziel C.....	894
Kapitel 187: UIStackView	898
Examples.....	898
Erstellen Sie eine horizontale Stapelansicht programmgesteuert.....	898
Erstellen Sie eine vertikale Stapelansicht programmgesteuert.....	898
Zentriertasten mit UIStackview.....	899
Kapitel 188: UIStackView dynamisch aktualisieren	907
Examples.....	907
Verbinden Sie den UISwitch mit einer Aktion, mit der Sie zwischen horizontalem oder vertikal.....	907
Kapitel 189: UIStoryboard	909
Einführung.....	909
Examples.....	909
Eine programmgesteuerte Instanz von UIStoryboard abrufen.....	909
SCHNELL:.....	909
ZIEL C:.....	909
Öffne ein anderes Storyboard.....	910
Kapitel 190: UISwitch	911
Syntax.....	911
Bemerkungen.....	911
1. UISwitch-Referenz: Apple-Dokumentation.....	911
2. Eine weitere Referenz von: Enoch Huang.....	911
Examples.....	911
Ein / Aus einstellen.....	911
Hintergrundfarbe einstellen.....	912

Tönungsfarbe einstellen.....	912
Bild für Ein / Aus einstellen.....	912
Kapitel 191: UITabBarController.....	914
Examples.....	914
Erstellen Sie eine Instanz.....	914
Ändern des Titeltitels und Symbols der Tab-Leiste.....	914
Ziel c:.....	915
Schnell:.....	915
Navigationscontroller mit TabBar.....	915
Farbe der Tab-Leistenfarbe.....	917
UITabBarController mit benutzerdefinierter Farbauswahl.....	917
Bild für die Tab-Leiste auswählen und hier den Tab-Titel einstellen.....	918
Auswahl einer anderen Registerkarte.....	919
Erstellen Sie programmgesteuert einen Tab-Leisten-Controller ohne Storyboard.....	919
Kapitel 192: UITableView.....	921
Einführung.....	921
Syntax.....	921
Bemerkungen.....	923
Examples.....	923
Zellen, die die Größe selbst bestimmen.....	923
Erstellen einer UITableView.....	924
Fügen Sie Ihrem Storyboard eine UITableView hinzu.....	924
Füllen Sie Ihre Tabelle mit Daten.....	925
Erstellen einer einfachen Datenquelle.....	925
Einrichten Ihrer Datenquelle in Ihrem View Controller.....	925
Verbinden der Datenquelle der Tabellensicht mit Ihrem Ansichtscontroller.....	926
Zeilenauswahlen behandeln.....	927
Das finale Resultat.....	927
Schnell.....	928
Ziel c.....	928
Delegieren und Datenquelle.....	929

UITableViewDataSource.....	930
UITableViewDelegate.....	933
Kundenspezifische Zellen.....	936
Erstellen Sie Ihre eigene Zelle.....	937
UITableViewCells erweitern und reduzieren.....	939
Zum Löschen von Zeilen streichen.....	941
Fügen Sie den Code hinzu.....	942
Storyboard.....	943
Fertig.....	943
Anmerkungen.....	944
Lesen Sie weiter.....	944
Trennlinien.....	944
Breite der Trennlinien bearbeiten.....	944
Ändern der Trennlinien für bestimmte Zellen.....	944
Entfernen Sie alle Trennlinien.....	945
Überschüssige Trennlinien ausblenden.....	945
Kapitel 193: UITableViewCell.....	947
Einführung.....	947
Examples.....	947
Xib-Datei von UITableViewCell.....	947
Kapitel 194: UITableViewController.....	949
Einführung.....	949
Examples.....	949
TableView mit dynamischen Eigenschaften mit tableViewCellStyle basic.....	949
TableView mit benutzerdefinierter Zelle.....	950
Kapitel 195: UI-Test.....	952
Syntax.....	952
Examples.....	952
Testdateien zu Xcode Project hinzufügen.....	952
Beim Erstellen des Projekts.....	952

Nachdem Sie das Projekt erstellt haben.....	952
Eingabehilfe.....	952
Wenn Eingabehilfen in Dienstprogrammen aktiviert ist.....	953
Wenn Eingabehilfen in Dienstprogrammen deaktiviert ist.....	953
Einrichten in UITest-Datei.....	954
UIView, UIImageView, UIScrollView.....	954
UILabel.....	955
UIStackView.....	955
UITableView.....	955
UITableViewCell.....	955
UITableViewCell-Elemente.....	955
UICollectionView.....	955
UIButton, UIBarButtonItem.....	955
UITextField.....	955
UITextView.....	956
UISwitch.....	956
Alarmer.....	956
Deaktivieren Sie Animationen während des UI-Tests.....	956
Mittagsessen und Anwendung während der Ausführung beenden.....	956
Mittagsanwendung zum Testen.....	956
Anwendung beenden.....	956
Geräte drehen.....	956
Kapitel 196: UITextField.....	958
Einführung.....	958
Syntax.....	958
Examples.....	959
Textfeld initialisieren.....	959
Schnell.....	959
Ziel c.....	959
Interface Builder.....	959
Eingabe-Zubehöransicht (Symbolleiste).....	959

Schnell.....	959
Ziel c.....	960
Automatische Kapitalisierung.....	960
Schnell.....	960
Ziel c.....	960
Tastatur schließen.....	960
Schnell.....	961
Ziel c.....	963
Ausrichtung einstellen.....	963
Schnell.....	963
Ziel c.....	963
KeyboardType.....	963
Verschieben des Bildlaufs, wenn UITextView zum Ersthelfer wird.....	964
Tastaturfokus erhalten und Tastatur ausblenden.....	966
Schnell.....	966
Ziel c.....	966
Schnell.....	966
Ziel c.....	967
Ersetzen Sie die Tastatur durch UIPickerView.....	967
Tastatur schließen, wenn der Benutzer die Rückkehrtaste drückt.....	970
Cursorposition abrufen und einstellen.....	971
Nützliche Informationen.....	971
Cursorposition abrufen.....	971
Cursorposition einstellen.....	971
verbunden.....	972
Anmerkungen.....	973
verbunden.....	973
Blinkende Einfügemarke ausblenden.....	973
Swift 2,3 <.....	973
Swift 3.....	973
Ziel c.....	974

Ändern Sie die Farbe und die Schriftart des Platzhalters	974
Erstellen Sie ein UITextField	974
Schnell	974
Ziel c	974
Kapitel 197: UITextField-Delegat	976
Examples	976
UITextField - Beschränkt das Textfeld auf bestimmte Zeichen	976
Nächstes Tag suchen und Tastatur verwalten	977
Aktionen, wenn ein Benutzer die Interaktion mit einem Textfeld gestartet / beendet hat	977
Kapitel 198: UITextView	979
Examples	979
Text ändern	979
Attributierten Text festlegen	979
Textausrichtung ändern	979
UITextViewDelegate-Methoden	979
Schriftart ändern	980
Textfarbe ändern	980
UITextView mit HTML-Text	980
Automatische Erkennung von Links, Adressen, Datumsangaben und mehr	981
Aktivieren der automatischen Erkennung	981
Anklickbare Daten	981
Überprüfen Sie, ob leer oder null ist	981
Cursorposition abrufen und einstellen	982
Nützliche Informationen	982
Cursorposition abrufen	982
Cursorposition einstellen	982
verbunden	983
Anmerkungen	984
verbunden	984
Entfernen Sie zusätzliche Füllungen, um einen genau gemessenen Text zu erhalten	984
Kapitel 199: UIView	985

Syntax.....	985
Bemerkungen.....	985
Examples.....	985
Erstellen Sie eine UIView.....	985
Machen Sie die Ansicht gerundet.....	986
Programmatisch.....	986
Storyboard-Konfiguration.....	987
Schnelle Erweiterung.....	988
Momentaufnahme machen.....	988
Verwendung von IBInspectable und IBDesignable.....	988
UIView animieren.....	991
UIView-Erweiterung für Größen- und Rahmenattribute.....	992
Programmgesteuertes Verwalten des Einfügens und Löschens von UIView in und aus einem ander.....	993
Erstellen Sie UIView mit Autolayout.....	994
Intrinsische Inhaltsgröße verwenden.....	996
Schüttle einen Blick.....	999
Kapitel 200: UIViewController.....	1001
Examples.....	1001
Unterklasse.....	1001
Erstellen Sie eine Instanz.....	1003
Stellen Sie die Ansicht programmgesteuert ein.....	1003
Instanzieren Sie aus einem Storyboard.....	1003
Rufen Sie den Container-View-Controller auf.....	1004
Hinzufügen / Entfernen eines untergeordneten Ansichtscrollers.....	1005
Kapitel 201: UIWebView.....	1006
Bemerkungen.....	1006
Examples.....	1006
Erstellen Sie eine UIWebView-Instanz.....	1006
URL-Anfrage stellen.....	1006
Laden Sie keine Webinhalte mehr.....	1007
Laden Sie den aktuellen Webinhalt neu.....	1007
Festlegen der Inhaltsgröße.....	1007

Laden Sie den HTML-String.....	1008
Laden Sie JavaScript.....	1008
Laden Sie Dokumentdateien wie .pdf, .txt, .doc usw.....	1009
Machen Sie Links, die in UIWebView anklickbar sind.....	1009
Lokale HTML-Datei in WebView laden.....	1010
Kapitel 202: Umgang mit URL-Schemata.....	1012
Syntax.....	1012
Parameter.....	1012
Bemerkungen.....	1012
Examples.....	1013
Verwenden des integrierten URL-Schemas zum Öffnen der Mail-App.....	1013
Schnell:.....	1013
Ziel c:.....	1013
Apple URL-Schemata.....	1013
Kapitel 203: Universelle Links.....	1017
Bemerkungen.....	1017
Examples.....	1017
Setup-Server.....	1017
Mehrere Domains unterstützen.....	1018
App-Site-Association-Datei signieren.....	1018
IOS-Anwendung einrichten (Universal Links aktivieren).....	1019
Ziel c.....	1022
Schnell:.....	1023
iOS-Anwendungscode.....	1023
Kapitel 204: UUID (Universally Unique Identifier).....	1024
Bemerkungen.....	1024
Examples.....	1024
UUID generieren.....	1024
Zufällige UUID.....	1024
Schnell.....	1024
Ziel c.....	1024

Kennung für Verkäufer.....	1024
Schnell.....	1024
Ziel c.....	1025
Apples IFA vs. IFV (Apple Identifier für Advertiser vs. Identifier für Anbieter).....	1025
Erstellen Sie eine UUID-Zeichenfolge für iOS-Geräte.....	1025
Swift 3,0.....	1025
Kapitel 205: Verwenden von Image Aseets.....	1027
Einführung.....	1027
Examples.....	1027
App-Symbol mit Bildelementen.....	1027
LaunchImage mit Image Assets.....	1030
Anmerkungen:.....	1034
Kapitel 206: WCSessionDelegate.....	1035
Einführung.....	1035
Examples.....	1035
Kit-Controller beobachten (WKInterfaceController).....	1035
Kapitel 207: WKWebView.....	1036
Einführung.....	1036
Examples.....	1036
Einen einfachen WebBrowser erstellen.....	1036
Hinzufügen eines benutzerdefinierten Benutzerskripts, das aus dem App-Paket geladen wurde.....	1042
Senden Sie Nachrichten aus JavaScript und behandeln Sie sie auf der nativen Seite.....	1042
Kapitel 208: Xcode Build & Archive von der Kommandozeile aus.....	1044
Syntax.....	1044
Parameter.....	1044
Bemerkungen.....	1044
Examples.....	1044
Bauen & Archivieren.....	1044
Kapitel 209: XCTest Framework - Unit Testing.....	1046
Examples.....	1046
Testdateien zu Xcode Project hinzufügen.....	1046

Beim Erstellen des Projekts	1046
Nachdem Sie das Projekt erstellt haben	1046
Schnell.....	1046
Ziel c.....	1047
Storyboard und View Controller als Instanzen zur Testdatei hinzufügen.....	1048
Definieren des View Controllers	1048
Schnell.....	1048
Vorstellung des Storyboards und Initialisieren des View Controllers	1048
Schnell.....	1048
Ziel c.....	1048
Testmethoden hinzufügen.....	1048
Testmethoden	1048
Schnell.....	1049
Ziel c.....	1049
Schnell.....	1049
Ziel c.....	1049
Hinweis	1049
Starten Sie den Test.....	1050
Testen einer bestimmten Methode	1050
Alle Methoden testen	1050
Sehen Sie das Testergebnis	1050
Alle Tests werden ausgeführt	1050
Importieren Sie ein Modul, das getestet werden kann.....	1051
Triggeransicht laden und Aussehen.....	1051
Laden anzeigen	1051
Aussehen anzeigen	1051
Testklasse schreiben.....	1051
Kapitel 210: Zugänglichkeit	1053
Einführung.....	1053
Examples.....	1053

Machen Sie eine Ansicht zugänglich.....	1053
Eingabehilfen.....	1053
Bildschirmwechsel.....	1053
Layout ändern.....	1054
Ankündigung.....	1054
Elemente bestellen.....	1054
Container für Barrierefreiheit.....	1055
Modale Ansicht.....	1055
Elemente ausblenden.....	1056
Credits.....	1057



You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [ios](#)

It is an unofficial and free iOS ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official iOS.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit iOS

Bemerkungen

Anmerkungen

1- Sie benötigen kein Apple Developer Account, um iOS Apps zu entwickeln. Die Dokumentation und die Tools können kostenlos mit Ihrer Apple ID heruntergeladen werden. Sie können Apps auch auf *Ihren persönlichen Geräten* mit derselben Apple-ID signieren und installieren. Wenn Sie Apps im [App Store](#) vertreiben oder verkaufen möchten, müssen Sie das Apple Developer Program ab 99 USD registrieren (Dies ist der Preis zum Zeitpunkt des Schreibens und kann sich ändern). Dies wird auch Support-Vorfälle auf Code-Ebene und Betatests für Ihre Apps über TestFlight hinzufügen.

2- Das Erstellen einer Apple ID ohne Kreditkarte [erfordert einen kurzen Vorgang](#) . Wenn Sie nichts dagegen haben, eine Zahlungsmethode als Teil der Anmeldung zu verknüpfen, gehen Sie zu <https://appleid.apple.com/>

- [Beginnen Sie mit der Entwicklung von iOS-Apps \(Swift\)](#)
- [Xcode-Hilfe \(einschließlich Erste Schritte\)](#)
- [Downloads \(einschließlich Xcode, wenn Sie nicht durch den AppStore gehen möchten\)](#)

Verwandte Stack Overflow-Tags

- [xcode](#) Apples IDE (Integrated Development Environment) für die Entwicklung von iOS- und MacOS-Apps
- [swift-language](#) Eine der Hauptsprachen, die Sie für die Entwicklung in iOS verwenden können.
- [object-c-language](#) Eine der Hauptsprachen, die Sie für die Entwicklung in iOS verwenden können.
- [cocoa](#) Eine Apple-API für die Entwicklung in iOS und macOS.
- [Sprite-Kit](#) Für animierte 2D-Grafiken.
- [Kerndaten](#) Zum Speichern und Abrufen relationaler Daten.

Versionen

Ausführung	Veröffentlichungsdatum
iPhone OS 2	2008-07-11
iPhone OS 3	2009-06-17

Ausführung	Veröffentlichungsdatum
iOS 4	2010-06-08
iOS 5	2011-10-12
iOS 6	2012-09-19
iOS 7	2013-09-18
iOS 8	2014-09-17
iOS 8.1	2014-10-20
iOS 8.2	2015-03-09
iOS 8.3	2015-04-08
iOS 8.4	2015-06-30
iOS 9	2015-09-16
iOS 9.1	2015-10-22
iOS 9.2	2015-12-08
iOS 9.3	2016-03-21
iOS 10.0.1	2016-09-13
iOS 10.1	2016-10-24
iOS 10.2	2016-12-12
iOS 10.2.1	2017-01-23
iOS 10.3	2017-03-27
iOS 10.3.3	2017-07-19

Examples

Standard-Single-View-Anwendung erstellen

Um eine Anwendung für iOS zu entwickeln, sollten Sie mit einer Anwendung namens Xcode beginnen. Es gibt andere alternative Tools, die Sie verwenden können, aber Xcode ist das offizielle Tool von Apple. Beachten Sie jedoch, dass es nur unter macOS läuft. Die neueste offizielle Version ist Xcode 8.3.3 mit Xcode 9 (derzeit in der Beta-Phase), die später in diesem Jahr veröffentlicht werden soll.

1. Starten Sie Ihren Mac und installieren Sie [Xcode aus dem App Store](#), falls er noch nicht installiert ist.

(Wenn Sie den App Store nicht verwenden möchten oder Probleme haben, können Sie [Xcode auch von der Apple Developer-Website herunterladen](#). Stellen Sie jedoch sicher, dass Sie die neueste Release-Version und **keine** Betaversion auswählen.)



2. Öffnen Sie Xcode. Folgendes Fenster wird geöffnet:



Welcome to Xcode

Version 8.0 (8A218a)



Get started with a playground

Explore new ideas quickly and easily.



Create a new Xcode project

Create an app for iPhone, iPad, Mac, Apple TV, or watchOS.



Check out an existing project

Start working on something from an SCM repository.

Das Fenster bietet Ihnen folgende Optionen:

- **Erste Schritte mit einem Spielplatz:** Dies wurde mit der Sprache Swift und Xcode 6 eingeführt. Es ist ein interaktiver Bereich, in dem kleine Code-Teile geschrieben werden können, um Laufzeitänderungen zu überprüfen. Es ist eine großartige Möglichkeit für Swift-Lernende, die neuen Swift-Funktionen kennenzulernen. Dies wurde mit der Sprache Swift und Xcode 6 eingeführt. Es ist ein interaktiver Bereich, in dem kleine Code-Teile geschrieben werden können, um Laufzeitänderungen zu überprüfen. Es ist eine großartige Möglichkeit für Swift-Lernende, die neuen Swift-Funktionen kennenzulernen.
- **Erstellen Sie ein neues Xcode-Projekt:** Wählen Sie diese Option , um ein neues Projekt mit Standardkonfiguration zu **erstellen** .
- **Ein vorhandenes Projekt auschecken:** Hiermit wird ein Projekt von einem Repository-Speicherort ausgecheckt, z. B. ein Projekt aus dem SVN.

3. Wählen Sie die zweite Option **Erstellen Sie ein neues Xcode-Projekt aus**, und Xcode fordert Sie auf, eine anfängliche Projekteinrichtung vorzunehmen:

Choose a template for your new project:

ios

watchOS

tvOS

macOS

Cross-platform

Application



Single View
Application



Game



Master
App



Sticker Pack
Application



iMessage
Application

Framework & Library



Cocoa Touch
Framework



Cocoa Touch
Static Library



Metal

Cancel

Dieser Assistent wird verwendet, um Ihre Projektvorlage auszuwählen. Es gibt 5 Möglichkeiten:

- **iOS:** Zum Erstellen von iOS-Apps, -Bibliotheken und -Frameworks
- **watchOS:** Zum Erstellen von WatchOS-Apps, -Bibliotheken und -Frameworks
- **tvOS:** Zum Erstellen von tvOS-Apps, -Bibliotheken und -Frameworks
- **macOS:** Zum Erstellen von MacOS-Apps, -Bibliotheken, Frameworks, Paketen, AppleScripts usw.
Zum Erstellen von MacOS-Apps, -Bibliotheken, Frameworks, Paketen, AppleScripts usw.
- **Plattformübergreifend:** Zum Erstellen von plattformübergreifenden Apps, Vorlagen und In-App-Kaufinhalten

Sie sehen, dass es für Ihre Anwendung viele verschiedene Vorlagen gibt. Diese Vorlagen sind hilfreich, um Ihre Entwicklung voranzutreiben. Sie sind mit einigen grundlegenden Projekteinstellungen wie UI-Schnittstellen und Klassendateien vorgefertigt.

Hier verwenden wir die erste Option, **iOS** .

1. Master-Detail-Anwendung:

Diese Vorlage enthält eine kombinierte Master- und Detailschnittstelle: Der Master enthält Objekte, die sich auf die Detailschnittstelle beziehen. Durch die Auswahl von Objekten im Master wird die Detailschnittstelle geändert. Sie können diese Art Benutzeroberfläche in den Anwendungen Einstellungen, Notizen und Kontakte auf dem iPad sehen.

2. Seitenbasierte Anwendung:

Diese Vorlage wird zum Erstellen der seitenbasierten Anwendung verwendet. Seiten sind verschiedene Ansichten, die von einem Container gehalten werden.

3. Einzelansicht-Anwendung:

Dies ist eine normale Anwendungsentwicklungsvorlage. Dies ist für Anfänger gut, um den Anwendungsfluss zu erlernen.

4. Tabbed-Anwendung:

Diese Vorlage erstellt Registerkarten im unteren Teil einer Anwendung. Jede Registerkarte hat eine andere Benutzeroberfläche und einen anderen Navigationsfluss. Diese Vorlage wird in Apps wie Clock, iTunes Store, iBooks und App Store verwendet.

5. Spiel:

Dies ist ein Ausgangspunkt für die Spielentwicklung. Mit Spieltechnologien wie SceneKit, SpriteKit, OpenGL ES und Metal können Sie noch weiter gehen.

4. In diesem Beispiel beginnen wir mit der **Einzelansicht-Anwendung**

Choose options for your new project:

Product Name:

Team:

None

Organization Name:

StackOver

Organization Identifier:

com.stacko

Bundle Identifier:

com.stacko

Language:

Swift

Devices:

Universal

Use Core

Include U

Include U

Cancel

Der Assistent hilft Ihnen, Projekteigenschaften zu definieren:

- **Produktname:** Der Name des Projekts / der Anwendung
- **Organisationsname:** Der Name der Organisation, an der Sie beteiligt sind
- **Organisationskennung:** Die eindeutige Organisationskennung, die in der Bündelkennung verwendet wird. Es wird empfohlen, die Notation des Reverse Domain Name Service einzuhalten.
Die eindeutige Organisationskennung, die in der Bündelkennung verwendet wird. Es wird empfohlen, die Notation des Reverse Domain Name Service einzuhalten.
- **Bundle Identifier:** *Dieses Feld ist sehr wichtig.* Sie basiert auf Ihrem Projektnamen und Ihrer Organisationskennung. Wählen Sie die Option mit Bedacht aus. Die Paketkennung wird in Zukunft verwendet, um die Anwendung auf einem Gerät zu installieren und die App zu iTunes Connect (dem Ort, an dem wir Apps hochladen, die im App Store veröffentlicht werden sollen) hochzuladen. Es ist ein eindeutiger Schlüssel, um Ihre Anwendung zu identifizieren.
- **Sprache:** Die Programmiersprache, die Sie verwenden möchten. Hier können Sie Objective-C in Swift ändern, wenn es nicht ausgewählt ist.
- **Geräte:** Unterstützte Geräte für Ihre Anwendung, die später geändert werden können. Es zeigt iPhone, iPad und Universal. Universelle Anwendungen unterstützen iPhone- und iPad-Geräte. Es wird empfohlen, diese Option zu wählen, wenn die App nicht auf nur einem Gerätetyp ausgeführt werden muss.
- **Core Data verwenden:** Wenn Sie Core Data Model in Ihrem Projekt verwenden möchten, markieren Sie es als ausgewählt, und es wird eine Datei für das `.xcdatamodel`. Sie können diese Datei auch später hinzufügen, wenn Sie dies nicht im Voraus wissen.
- **Unit-Tests einschließen:** Hiermit wird das Unit-Testziel konfiguriert und Klassen für Unit-Tests erstellt
- **UI-Test einschließen:** Hiermit wird das UI-Testziel konfiguriert und Klassen für den UI-Test erstellt

Klicken Sie auf **Weiter**, und Sie werden nach einem Ort gefragt, an dem Sie das Projektverzeichnis erstellen möchten.

Klicken Sie auf **Erstellen**, und Sie sehen die Xcode-Benutzeroberfläche mit einem bereits definierten Projekt-Setup. Sie können einige Klassen und Storyboard-Dateien sehen.

Dies ist eine grundlegende Vorlage für eine Single View-Anwendung.

Prüfen Sie oben links im Fenster, ob ein Simulator ausgewählt ist (z. B. "iPhone 6" wie hier gezeigt), und drücken Sie dann die dreieckige RUN-Taste.



5. Eine neue Anwendung wird geöffnet - Simulator (dies kann bei der ersten Ausführung einige Zeit dauern, und Sie müssen möglicherweise zwei Mal versuchen, wenn beim ersten Mal ein Fehler angezeigt wird). Diese Anwendung bietet uns die Gerätesimulation für erstellte Anwendungen. Es sieht fast aus wie ein echtes Gerät! Es enthält einige Anwendungen wie

ein echtes Gerät. Sie können Orientierungen, Positionen, Schüttelbewegungen, Erinnerungswarnungen, Statusanzeigen während des Anrufs, Fingerberührungen, Sperren, Neustart usw.

Sie sehen eine einfache weiße Anwendung, da wir an der Vorlage noch keine Änderungen vorgenommen haben.

Beginnen Sie also selbst. Es ist ein langer Weg und es gibt viele neue Möglichkeiten für Sie!

Wenn Sie nicht sicher sind, wohin Sie als Nächstes gehen sollen, probieren Sie das "[Jump Right In](#)" -Tutorial von Apple aus. Sie haben bereits die ersten Schritte ausgeführt und sind damit auf dem richtigen Weg.

Hallo Welt

Nach dem Einrichten von Xcode ist es nicht schwierig, das erste iOS-Gerät in Betrieb zu nehmen. Im folgenden Beispiel werden wir:

- Starten Sie ein neues Projekt
- Fügen Sie eine Beschriftung hinzu
- Nachricht an Konsole ausgeben.
- Führen Sie im Simulator aus

Neues Projekt starten

Wenn der Xcode-Begrüßungsbildschirm angezeigt wird, wählen **Sie Neues Xcode-Projekt erstellen** . Alternativ können Sie auch **Datei> Neu> Projekt ...** über das Xcode-Menü ausführen, wenn Sie es bereits geöffnet haben.



Welcome to Xcode

Version ...



Get started with a playground

Explore new ideas quickly and easily.



Create a new Xcode project

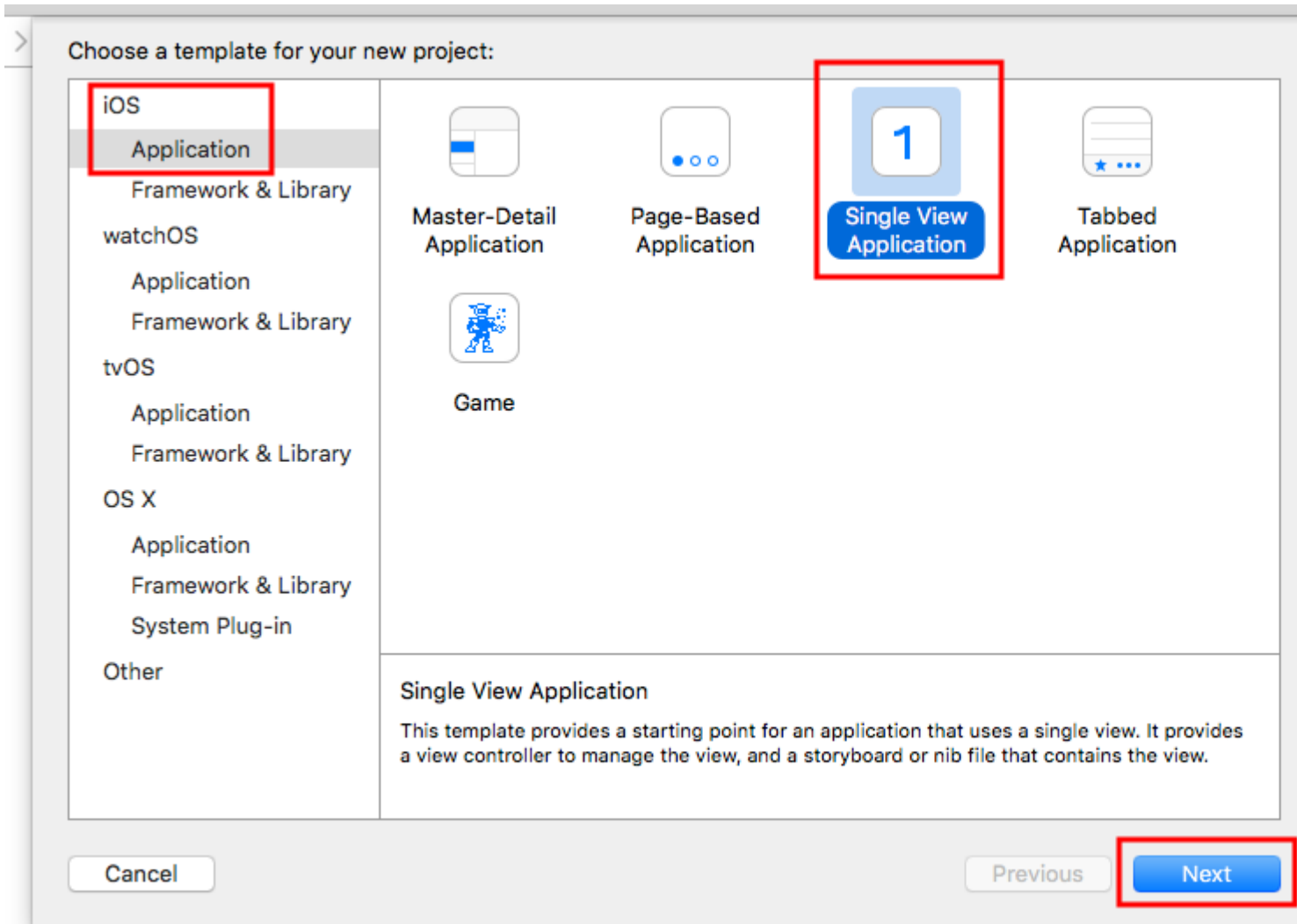
Start building a new iPhone, iPad or Mac application.



Check out an existing project

Start working on something from an SCM repository.

Wählen Sie eine **Einzelansicht aus** und klicken Sie auf **Weiter** .



Schreiben Sie "HelloWorld" für den **Produktnamen** (oder was auch immer Sie möchten) und stellen Sie unter **Sprache** sicher, dass **Swift** ausgewählt ist.

- **Universal** bedeutet, dass Ihre App sowohl auf dem iPhone als auch auf dem iPad ausgeführt werden kann.
- **Use Core Data (Kerndaten verwenden)** bezieht sich auf persistente Datenspeicherung, die in unserer Hello World-App nicht benötigt wird.
- Wir werden in diesem Beispiel keine **Unit-Tests** oder **UI-Tests durchführen** , aber es schadet nicht, wenn Sie sich daran gewöhnen, sie hinzuzufügen.

> Choose options for your new project:

Product Name: HelloWorld

Organization Name: Me

Organization Identifier: com.example

Bundle Identifier: com.example.HelloWorld

Language: Swift

Devices: Universal

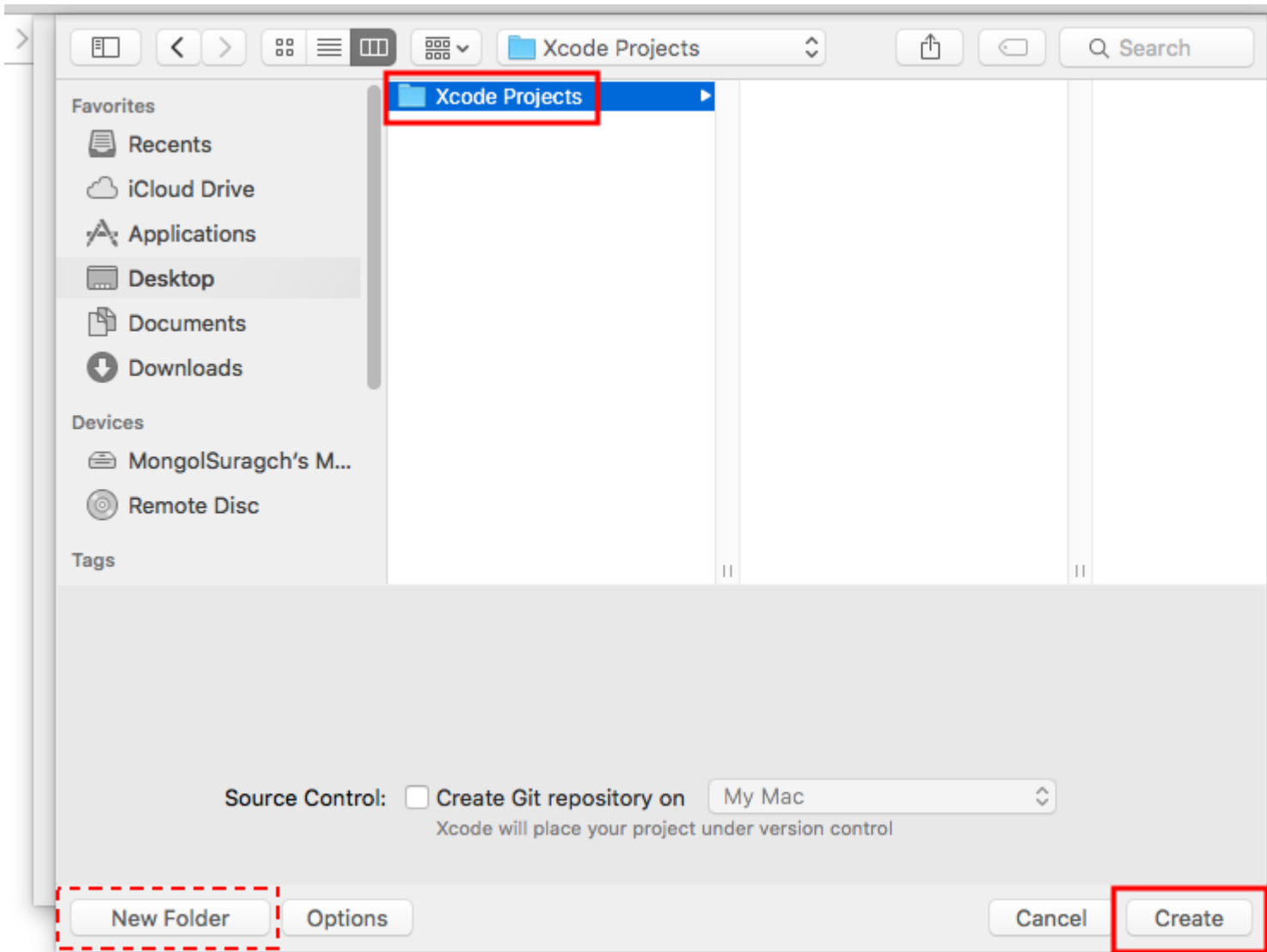
Use Core Data

Include Unit Tests

Include UI Tests

Cancel Previous Next

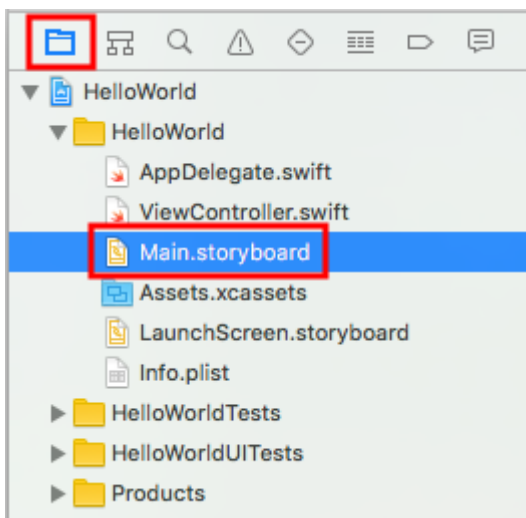
Wählen Sie einen vorhandenen Ordner aus oder erstellen Sie einen neuen, in dem Sie Ihre Xcode-Projekte speichern. Dies wird in Zukunft der Standard sein. Wir haben hier eines namens "Xcode-Projekte" erstellt. Klicken **Sie** dann auf **Erstellen** . Wenn Sie möchten, können Sie Quellcodeverwaltung auswählen (wird bei der Synchronisierung mit Sites wie [GitHub verwendet](#)), aber in diesem Beispiel werden wir sie nicht benötigen.



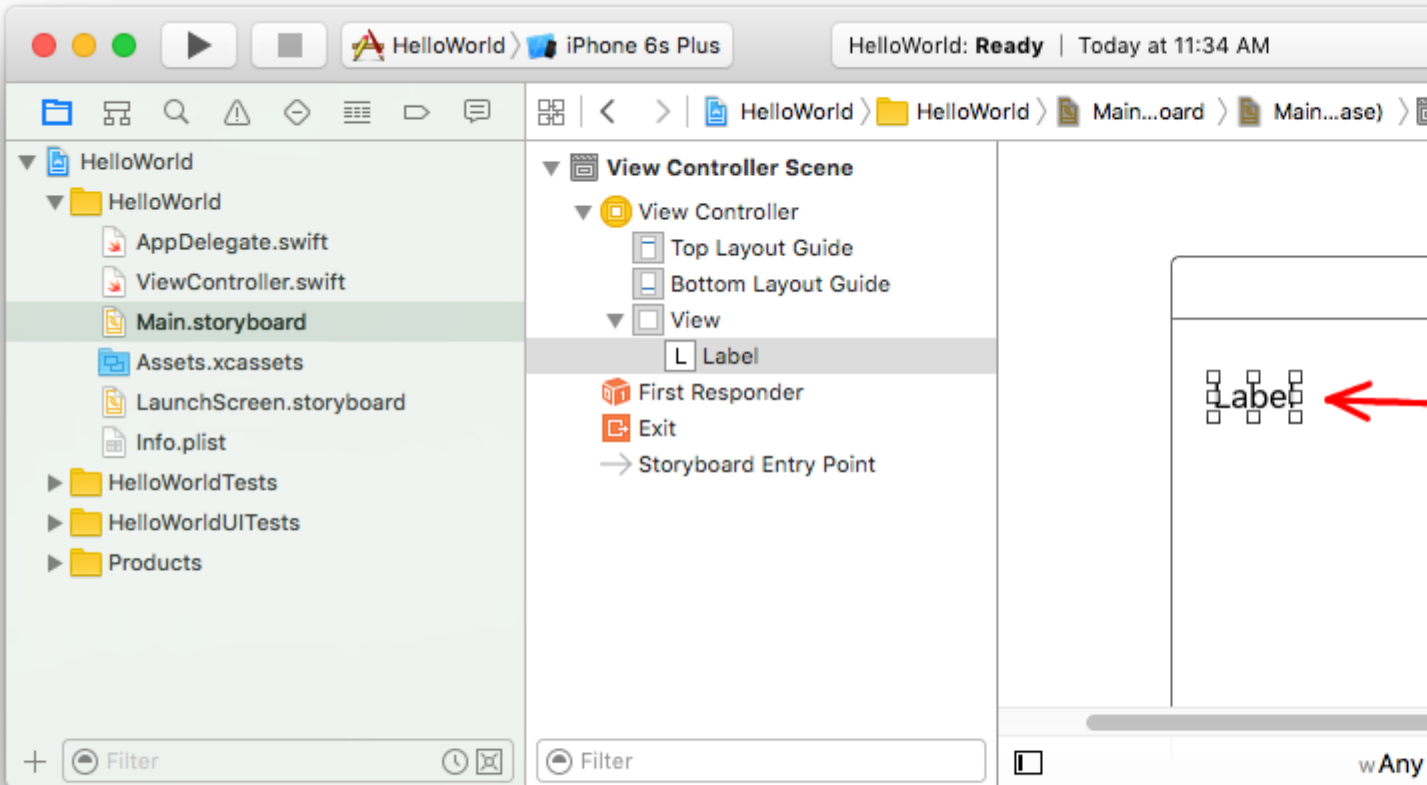
Beschriftung hinzufügen

Dies ist die Dateistruktur eines Xcode-Projekts.

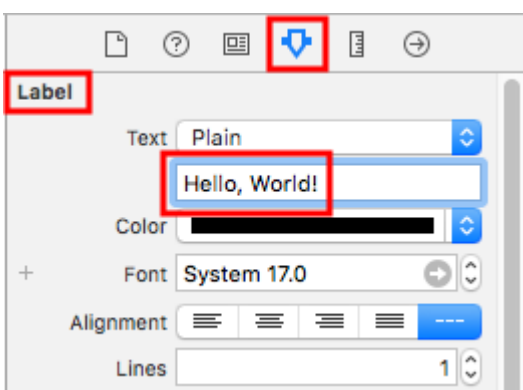
Wählen Sie im Projektnavigator *Main.storyboard* aus.



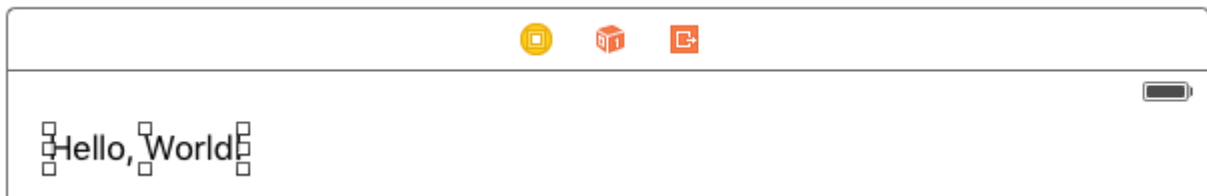
Geben Sie "label" in das Suchfeld der Objektbibliothek rechts unten in Xcode ein. Ziehen Sie dann das UILabel auf den Storyboard View Controller. Platzieren Sie es generell im Bereich der oberen linken Ecke.



Vergewissern Sie sich, dass das Label im Storyboard ausgewählt ist, und ändern Sie im **Eigenschafteninspektor** den Text in "Hallo, Welt!". Sie müssen dann die Größe des Labels im Storyboard ändern und neu positionieren, da die Textlänge jetzt länger ist.

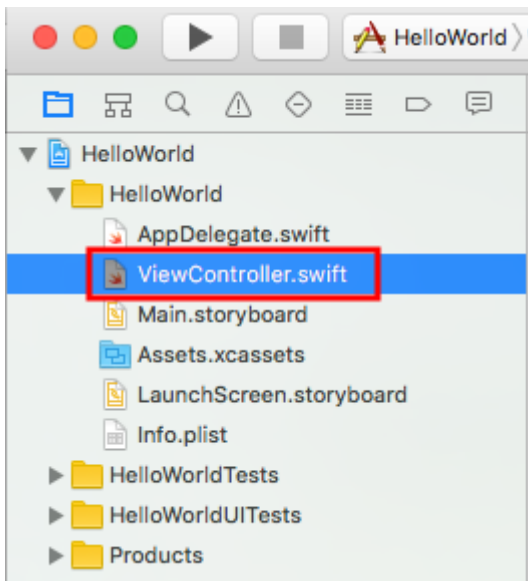


Alternativ können Sie auch auf das Label im Storyboard doppelklicken, um es als "Hallo, Welt!" zu bearbeiten. Auf jeden Fall sollte das Storyboard so aussehen:



Code hinzufügen

Wählen Sie im Projektnavigator *ViewController.swift* aus.



Fügen Sie der `viewDidLoad()` Methode den `print("Successfully created my first iOS application.")` `viewDidLoad()` . Es sollte ungefähr so aussehen.

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // print to the console when app is run
        print("Successfully created my first iOS application.")
    }

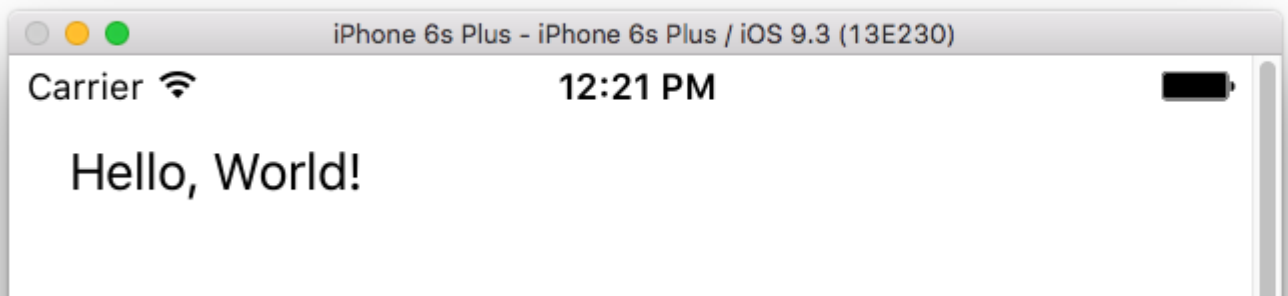
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

App im Simulator ausführen



Klicken Sie auf die Schaltfläche Ausführen, um die App zu erstellen und auszuführen. In diesem Beispiel ist das aktuelle Simulatorgerät (als "Schema" bezeichnet) standardmäßig das iPhone 6s Plus. Neuere Versionen von Xcode verwenden standardmäßig neuere Schemata. Sie können auch andere Schemata auswählen, indem Sie auf den Namen klicken. Wir werden einfach beim Standard bleiben.

Der Simulator benötigt beim ersten Durchlauf einige Zeit, um zu starten. Sobald es läuft, sollte es so aussehen:



Im Simulormenü können Sie „ **Fenster**“> „ Skalieren“ auswählen, um es zu verkleinern, oder cmd + 1/2/3/4/5 für eine 100% / 75% / 50% / 33% / 25% -Skala drücken.

Der Xcode-Debug-Bereich (unten) sollte auch gedruckt haben "Meine erste iOS-Anwendung wurde erfolgreich erstellt." zur Konsole. "Meine erste iOS-Anwendung wurde erfolgreich erstellt." message ist die Zeichenfolge, die Sie programmgesteuert im **Code hinzufügen**- Bereich gedruckt haben.



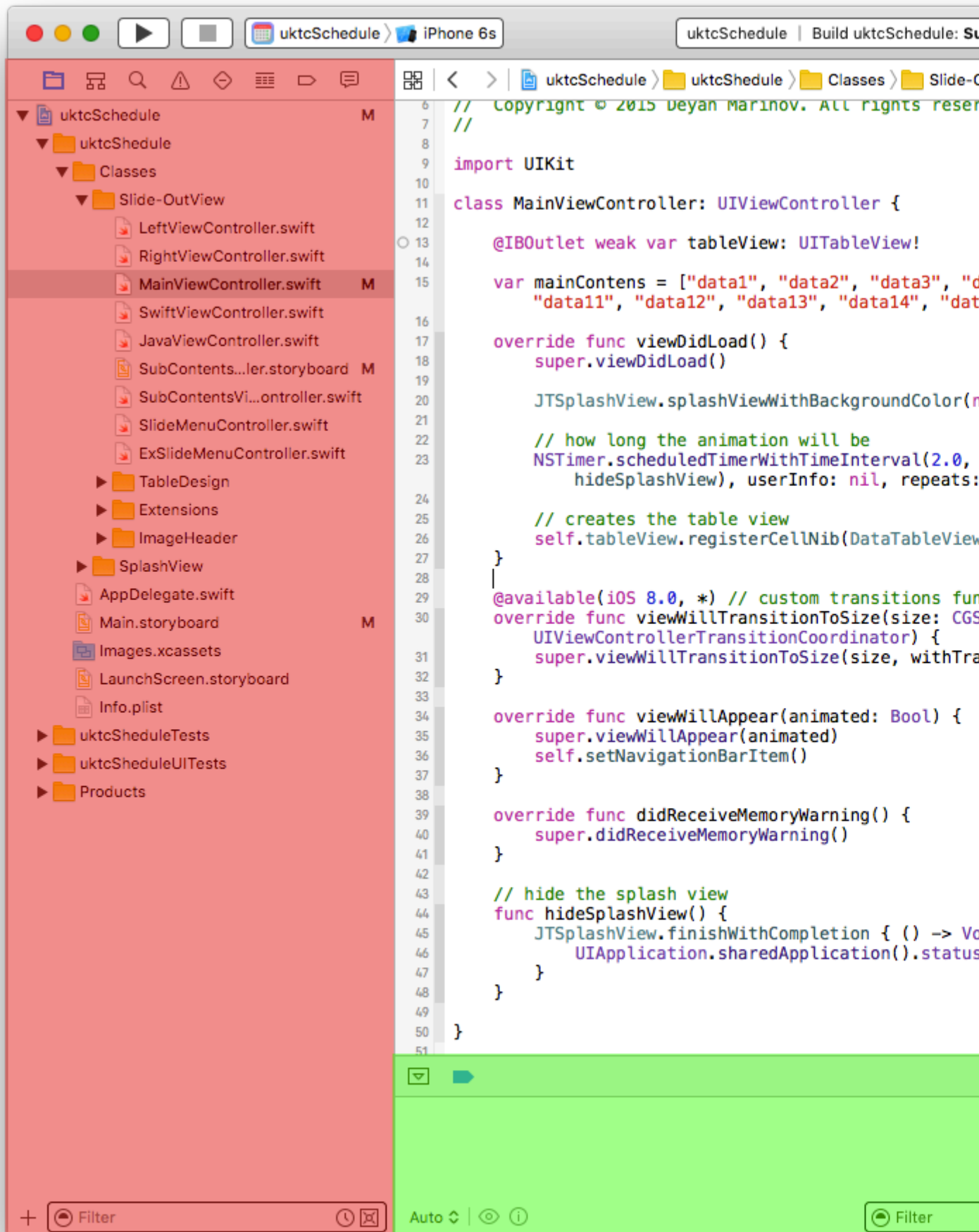
Weitermachen

Als Nächstes sollten Sie etwas über die Einschränkungen des automatischen Layouts lernen. Diese helfen Ihnen, Ihre Steuerelemente auf dem Storyboard so zu positionieren, dass sie bei jeder Gerätegröße und -ausrichtung gut aussehen.

Xcode-Schnittstelle

Im Xcode haben Sie drei separate Arbeitsbereiche - Navigatoren (rot), Debug-Bereich (grün) und

Dienstprogramme (blau).



Das Arbeitsbereichsfenster enthält immer den Editorbereich. Wenn Sie eine Datei in Ihrem Projekt auswählen, wird ihr Inhalt im Editorbereich angezeigt, wo Xcode die Datei in einem geeigneten Editor öffnet. Im obigen Bild ist beispielsweise der Editorbereich `MainViewController.swift` eine Swift-Code-Datei, die im Navigatorbereich links im Arbeitsbereichsfenster ausgewählt wird.

Navigator-Bereich



Das Navigatorfenster enthält die folgenden acht Optionen:

- **Projektnavigator.** Dateien in Ihrem Projekt hinzufügen, löschen, gruppieren und anderweitig verwalten, oder wählen Sie eine Datei aus, um deren Inhalt im Editorbereich anzuzeigen oder zu bearbeiten.
- **Symbolnavigator.** Durchsuchen Sie die Symbole in Ihrem Projekt als Liste oder Hierarchie. Mit den Schaltflächen auf der linken Seite der Filterleiste können Sie die angezeigten Symbole auf eine Kombination aus nur Klassen und Protokollen, nur Symbolen in Ihrem Projekt oder nur Containern beschränken.
- **Navigator** suchen Verwenden Sie Suchoptionen und Filter, um alle Zeichenfolgen in Ihrem Projekt schnell zu finden.
- **Ausgabennavigator.** Anzeigen von Problemen wie Diagnosen, Warnungen und Fehlern beim Öffnen, Analysieren und Erstellen Ihres Projekts.
- **Navigator testen.** Erstellen, Verwalten, Ausführen und Überprüfen von Komponententests.
- **Debug-Navigator** Untersuchen Sie die laufenden Threads und die zugehörigen Stack-Informationen zu einem bestimmten Zeitpunkt während der Programmausführung.
- **Haltepunkt-Navigator.** Optimieren Sie Haltepunkte, indem Sie Eigenschaften wie Auslösebedingungen festlegen.
- **Berichtsnavigator.** Zeigen Sie den Verlauf Ihrer Build-, Ausführungs-, Debugging-, kontinuierlichen Integrations- und Quellcodeverwaltungsaufgaben an.

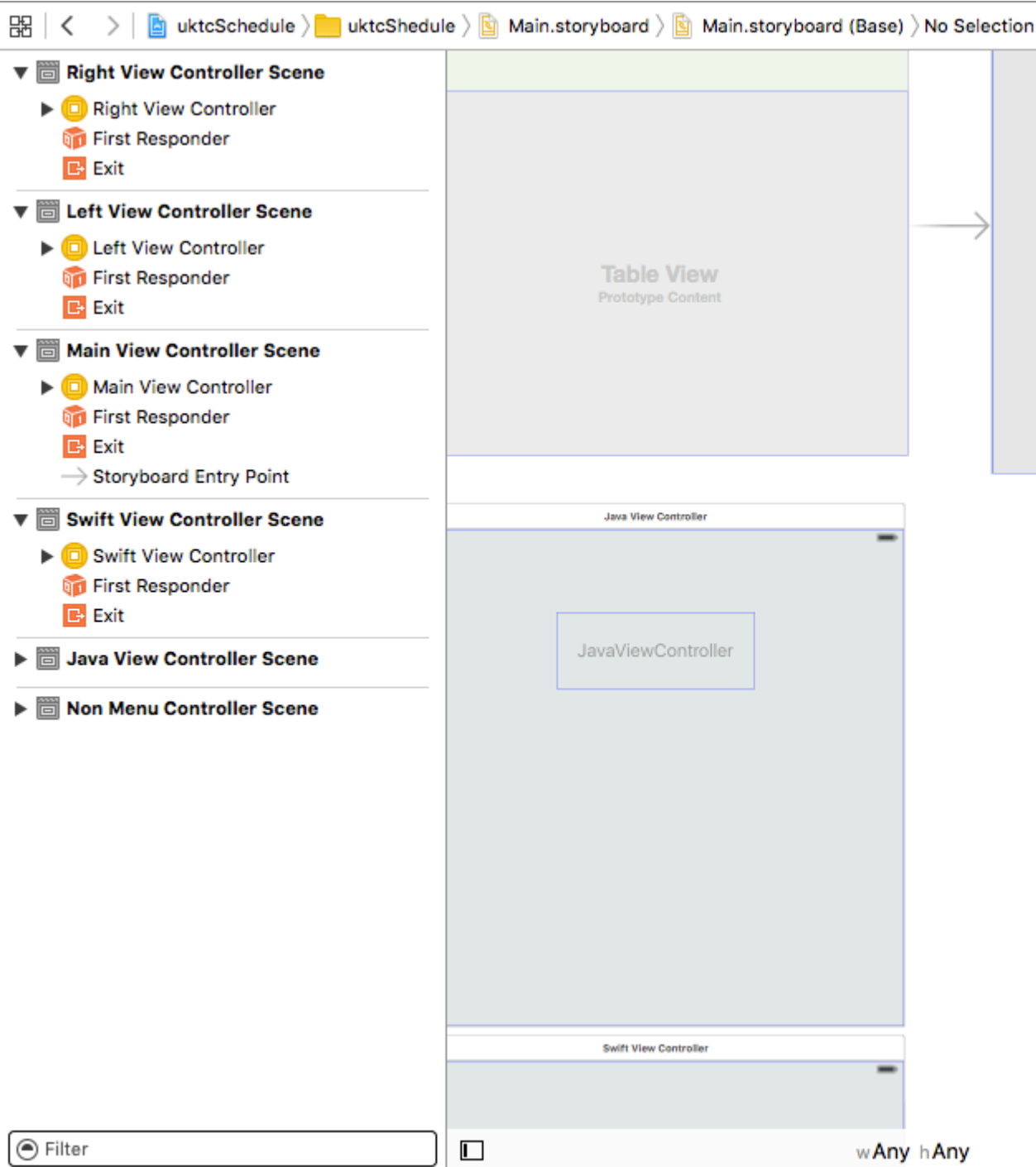
Die Herausgeber

Die meisten Entwicklungsarbeiten in Xcode erfolgen im Editorbereich, dem Hauptbereich, der immer innerhalb des Arbeitsbereichsfensters sichtbar ist. Die Editoren, die Sie am häufigsten verwenden, sind:

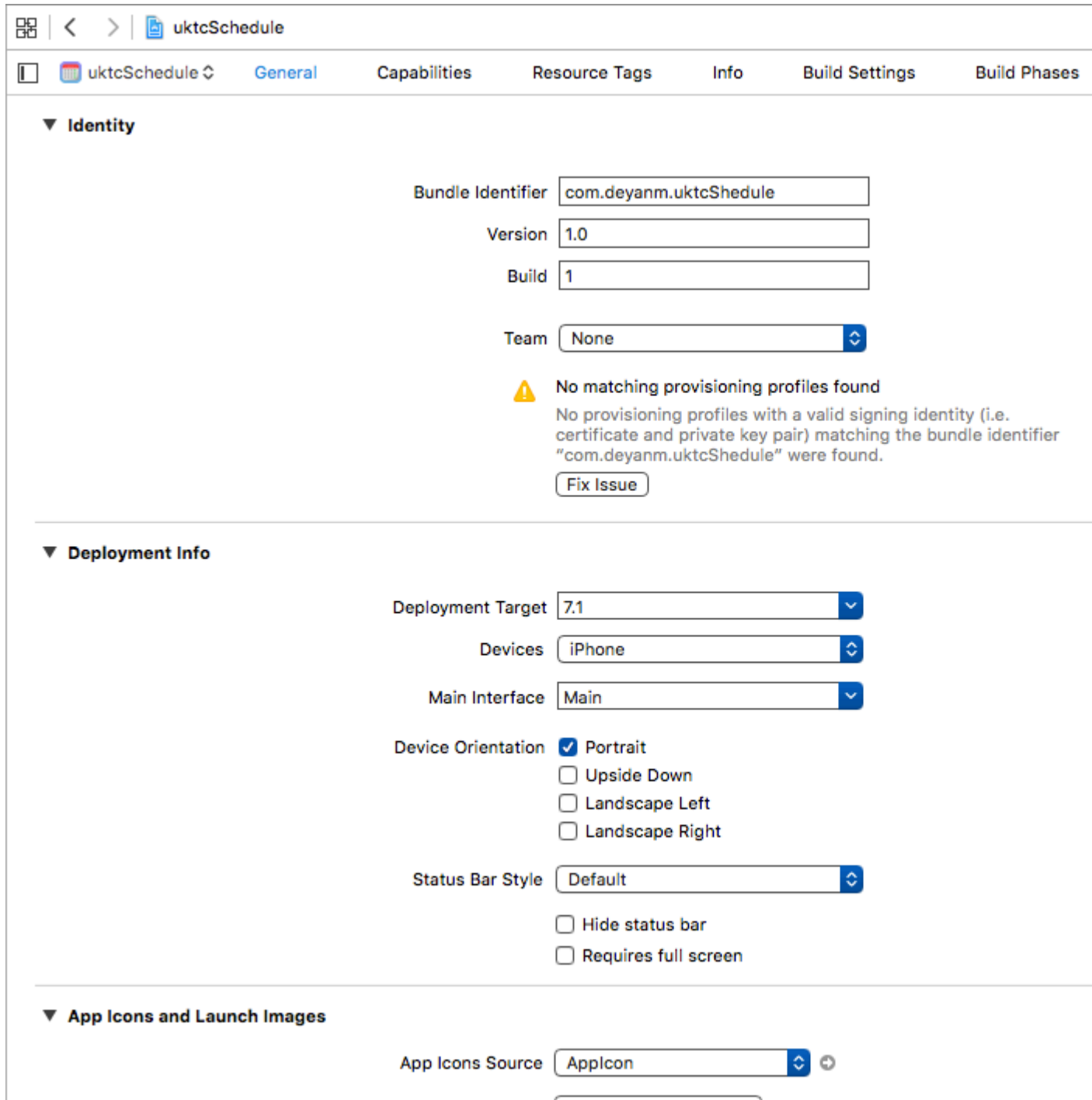
- **Quelledgeitor** Quellcode schreiben und bearbeiten.

```
uktcSchedule > uktcShedule > Classes > Slide-OutView > LeftViewController.swift > No Selection
1 //
2 // LeftViewController.swift
3 // uktcShedule
4 //
5 // Created by Deyan Marinov on 10/9/15.
6 // Copyright © 2015 Deyan Marinov. All rights reserved.
7 //
8
9 import UIKit
10
11 enum LeftMenu: Int {
12     case Main = 0
13     case Swift
14     case Java
15 }
16
17 protocol LeftMenuProtocol : class {
18     func changeViewController(menu: LeftMenu)
19 }
20
21 class LeftViewController : UIViewController, LeftMenuProtocol {
22
23     @IBOutlet weak var tableView: UITableView!
24     var menus = ["Main", "Swift", "Java"]
25     var mainViewController: UIViewController!
26     var swiftViewController: UIViewController!
27     var javaViewController: UIViewController!
28     var goViewController: UIViewController!
29     var nonMenuViewController: UIViewController!
30     var imageHeaderView: ImageHeaderView!
31
32     required init?(coder aDecoder: NSCoder) {
33         super.init(coder: aDecoder)
34     }
35
36     override func viewDidLoad() {
37         super.viewDidLoad()
38         self.tableView.separatorColor = UIColor(red: 224/255, green: 224/255, blue: 224/255,
39
40         let storyboard = UIStoryboard(name: "Main", bundle: nil)
41         let swiftViewController = storyboard.instantiateViewControllerWithIdentifier("SwiftV
42         self.swiftViewController = UINavigationController(rootViewController: swiftViewContr
43
44         let javaViewController = storyboard.instantiateViewControllerWithIdentifier("JavaVie
45         self.javaViewController = UINavigationController(rootViewController: javaViewControl
46
47         self.tableView.registerClass(DownTableViewCell.self)
48     }
49 }
```

- **Interface Builder.** Benutzeroberflächendateien grafisch erstellen und bearbeiten.



- **Projekteditor** Zeigen Sie an und bearbeiten Sie, wie Ihre Apps erstellt werden sollen, z. B. durch Angabe von Erstellungsoptionen, Zielarchitekturen und App-Berechtigungen.



Konfigurieren Sie den Editorbereich für eine bestimmte Aufgabe mit den

Editorkonfigurationsschaltflächen auf der rechten Seite der Symbolleiste:

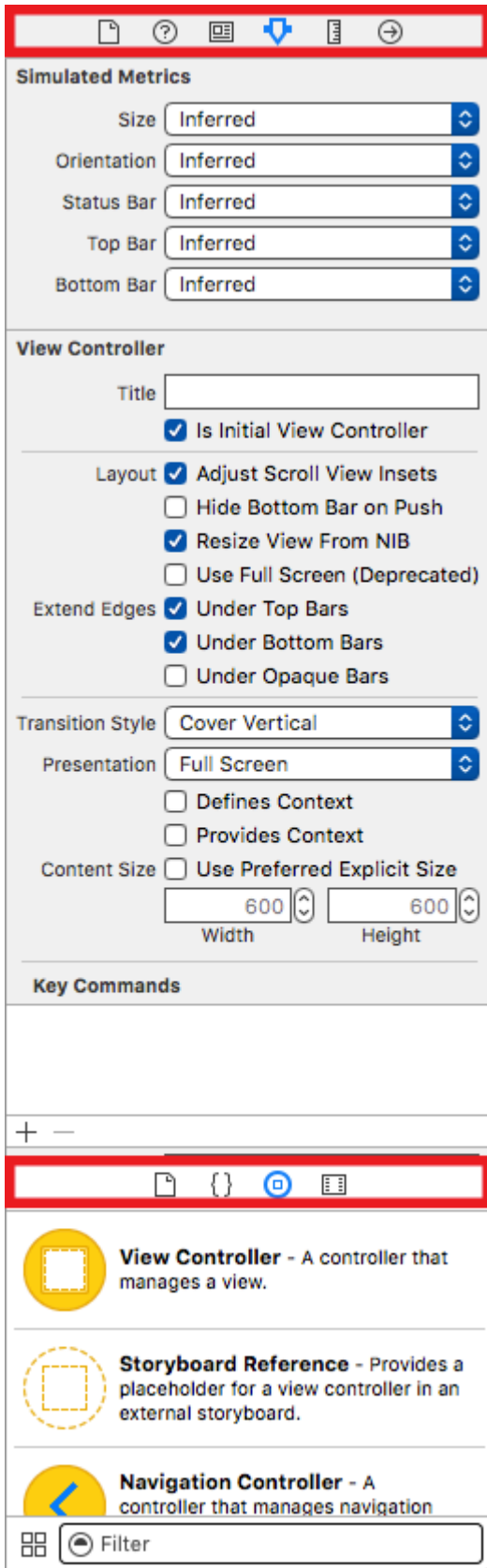


- **Standardeditor.** Füllt den Editorbereich mit dem Inhalt der ausgewählten Datei.
- **Schnittassistentz.** Zeigt einen separaten Editorbereich an, dessen Inhalt logisch mit dem Inhalt des Standard-Editorfensters zusammenhängt. Sie können den Inhalt auch ändern.
- **Versions-Editor** Zeigt die Unterschiede zwischen der ausgewählten Datei in einem Bereich und einer anderen Version derselben Datei in einem zweiten Bereich. Dieser Editor funktioniert nur, wenn sich Ihr Projekt unter Quellcodeverwaltung befindet.

Ressourcen und Elemente im Versorgungsbereich

Über den Dienstprogrammbereich ganz rechts im Arbeitsbereichsfenster haben Sie schnellen Zugriff auf diese Ressourcen: Inspektoren zum Anzeigen und Ändern der Eigenschaften der in einem Editor geöffneten Datei. Bibliotheken mit vorgefertigten Ressourcen für die Verwendung in Ihrem Projekt

Im oberen Bereich des Dienstprogramms werden Inspektoren angezeigt. Im unteren Bereich haben Sie Zugriff auf Bibliotheken.



Das erste Feld (rot hervorgehoben) ist die **Inspektorleiste**. Wählen Sie damit den Inspector aus, der für Ihre aktuelle Aufgabe am besten geeignet ist. In der Inspektorleiste sind immer zwei Inspektoren sichtbar (in einigen Editoren sind zusätzliche Inspektoren verfügbar):

- **Dateiinspektor** Anzeigen und Verwalten von Metadaten für die ausgewählte Datei. Normalerweise lokalisieren Sie Storyboards und andere Mediendateien und ändern

Einstellungen für Benutzeroberflächendateien.

- **Schnelle Hilfe.** Zeigen Sie Details zu einem Symbol, einem Schnittstellenelement oder einer Build-Einstellung in der Datei an. In der Schnellhilfe wird beispielsweise eine kurze Beschreibung der Methode angezeigt, wo und wie die Methode deklariert wird, deren Gültigkeitsbereich, die verwendeten Parameter sowie die Verfügbarkeit der Plattform und Architektur.

Verwenden Sie die **Bibliotheksleiste** (die zweite rot hervorgehobene), um auf betriebsbereite Ressourcenbibliotheken für Ihr Projekt zuzugreifen:

- **Dateivorlagen** Vorlagen für allgemeine Dateitypen und Code-Konstrukte.
- **Code Ausschnitte.** Kurze Quellcodes für die Verwendung in Ihrer Software, wie Klassendeklarationen, Steuerabläufe, Blockdeklarationen und Vorlagen für häufig verwendete Apple-Technologien.
- **Objekte.** Elemente für die Benutzeroberfläche Ihrer App.
- **Medien.** Dateien mit Grafiken, Symbolen, Sounddateien und dergleichen.

Um eine Bibliothek zu verwenden, ziehen Sie sie direkt in den entsprechenden Bereich. Um beispielsweise ein Code-Snippet zu verwenden, ziehen Sie es aus der Bibliothek in den Quelleditor. Um eine Quelldatei aus einer Dateivorlage zu erstellen, ziehen Sie die Vorlage in den Projektnavigator.

Um die in einer ausgewählten Bibliothek angezeigten Elemente einzuschränken, geben Sie den relevanten Text in das Textfeld in der **Filterleiste** (unterer Bereich) ein. Geben Sie beispielsweise "button" in das Textfeld ein, um alle Schaltflächen in der Objektbibliothek anzuzeigen.

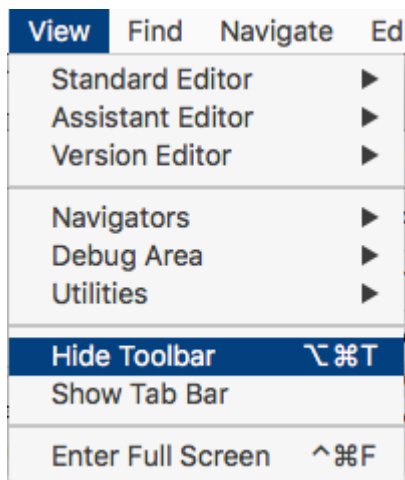
Verwalten Sie Aufgaben mit der Arbeitsbereich-Symboleiste

Die Symboleiste oben im Arbeitsbereich bietet schnellen Zugriff auf häufig verwendete Befehle. Mit der **Schaltfläche Ausführen können Sie** Ihre Produkte erstellen und ausführen. Die **Schaltfläche Stopp** beendet Ihren Laufcode. Mit dem **Menü Schema** können Sie die Produkte konfigurieren, die Sie erstellen und ausführen möchten. Die **Aktivitätsanzeige** zeigt den Fortschritt der aktuell ausgeführten Aufgaben an, indem Statusmeldungen, Build-Fortschritt und andere Informationen zu Ihrem Projekt angezeigt werden.

Mit den **Konfigurationsschaltflächen des Editors** (die erste Gruppe der drei Schaltflächen) können Sie den **Editorbereich konfigurieren**. Mit den **Konfigurationsschaltflächen des Arbeitsbereichs** (die zweite Gruppe der drei Schaltflächen) können Sie die optionalen Bereiche Navigator, Debug und Dienstprogramme ein- oder ausblenden.



Das **Menü Ansicht** enthält Befehle zum Ein- oder Ausblenden der Symbolleiste.



Erstellen Sie Ihr erstes Programm in Swift 3

Hier stelle ich vor, wie Sie ein erstes Basisprogramm in Swift 3 erstellen. Zuerst müssen Sie über grundlegende Programmiersprachenkenntnisse verfügen oder nicht bereit sein, sie von Anfang an zu lernen.

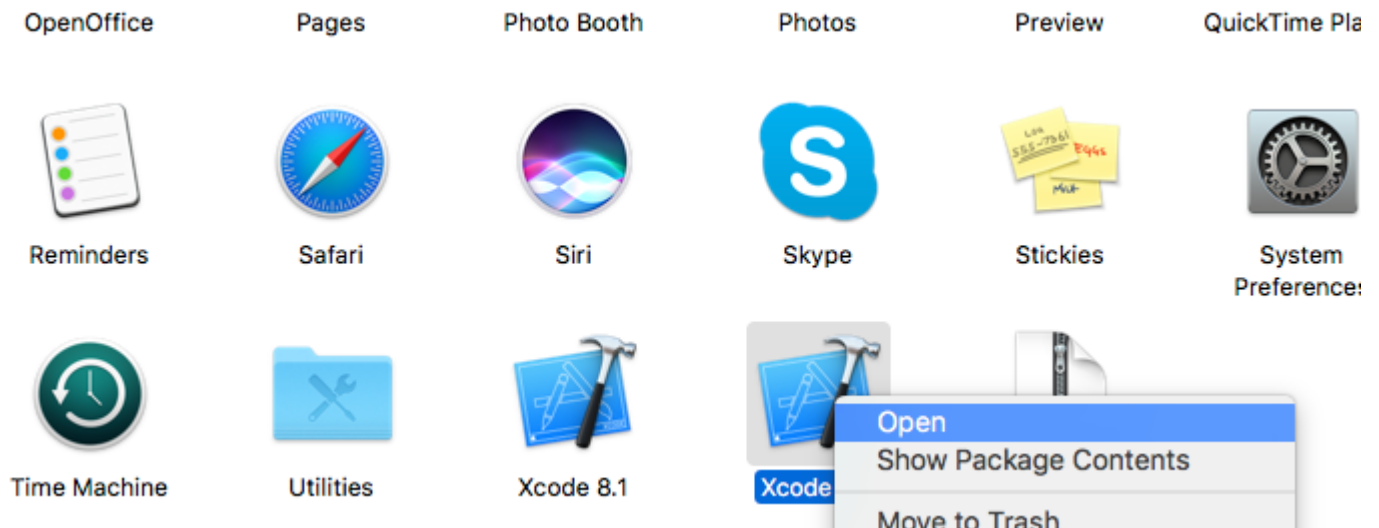
Anforderungen an Entwicklungen:

1. MAC OS - Version 10.11.6 oder höher für neuen Xcode 8.2
2. Xcode - Version 8.2 [Apple Document for Xcode Einführung](#).

Xcode 8.2 verfügt über neue Swift 3-Sprachfunktionen mit neuen iOS 10-kompatiblen APIs.

Erstellen Sie Ihr erstes Programm

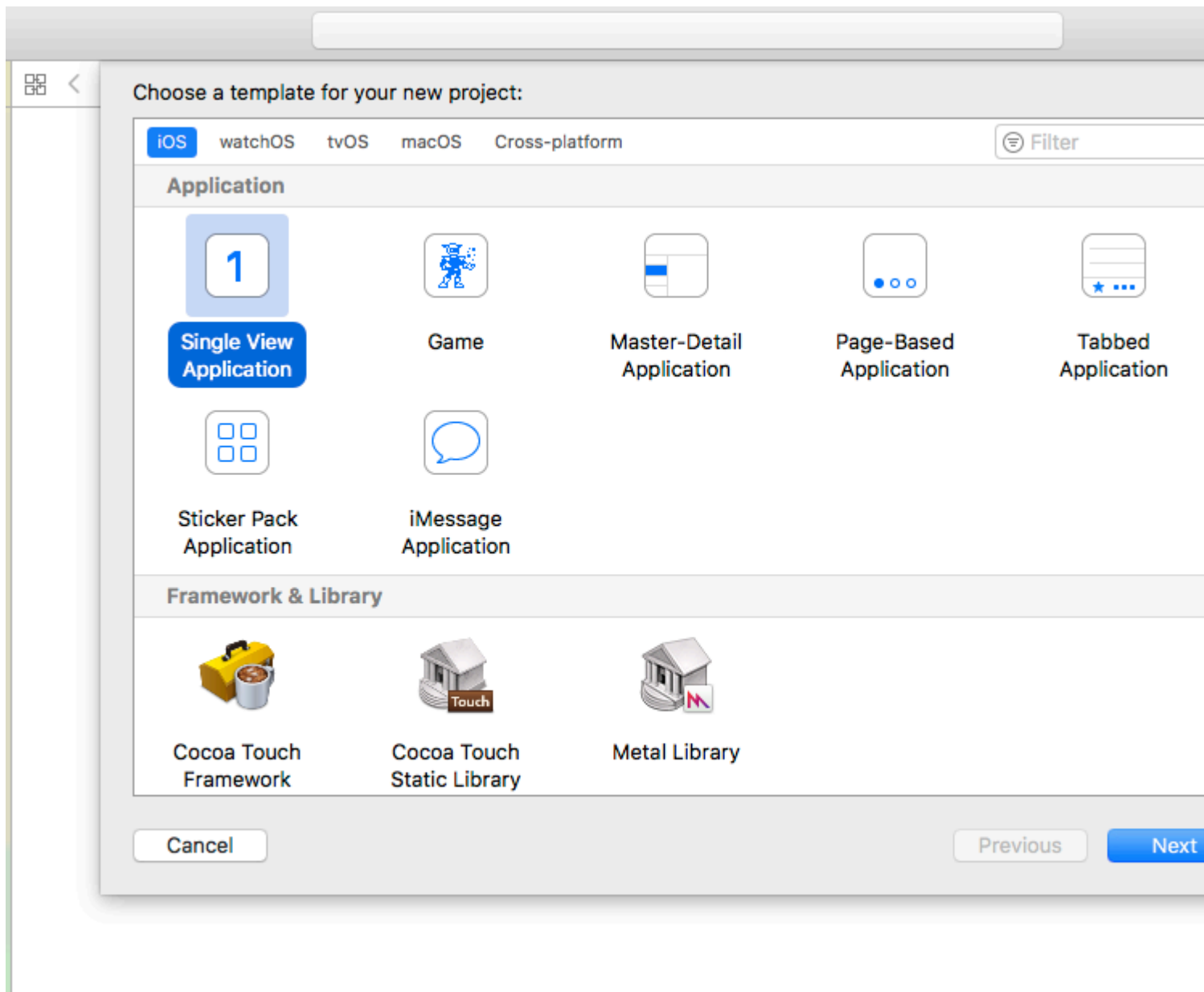
Gehen Sie zuerst zur Anwendung und öffnen Sie Ihren Xcode 8.2.



Danach sehen Sie den Bildschirm



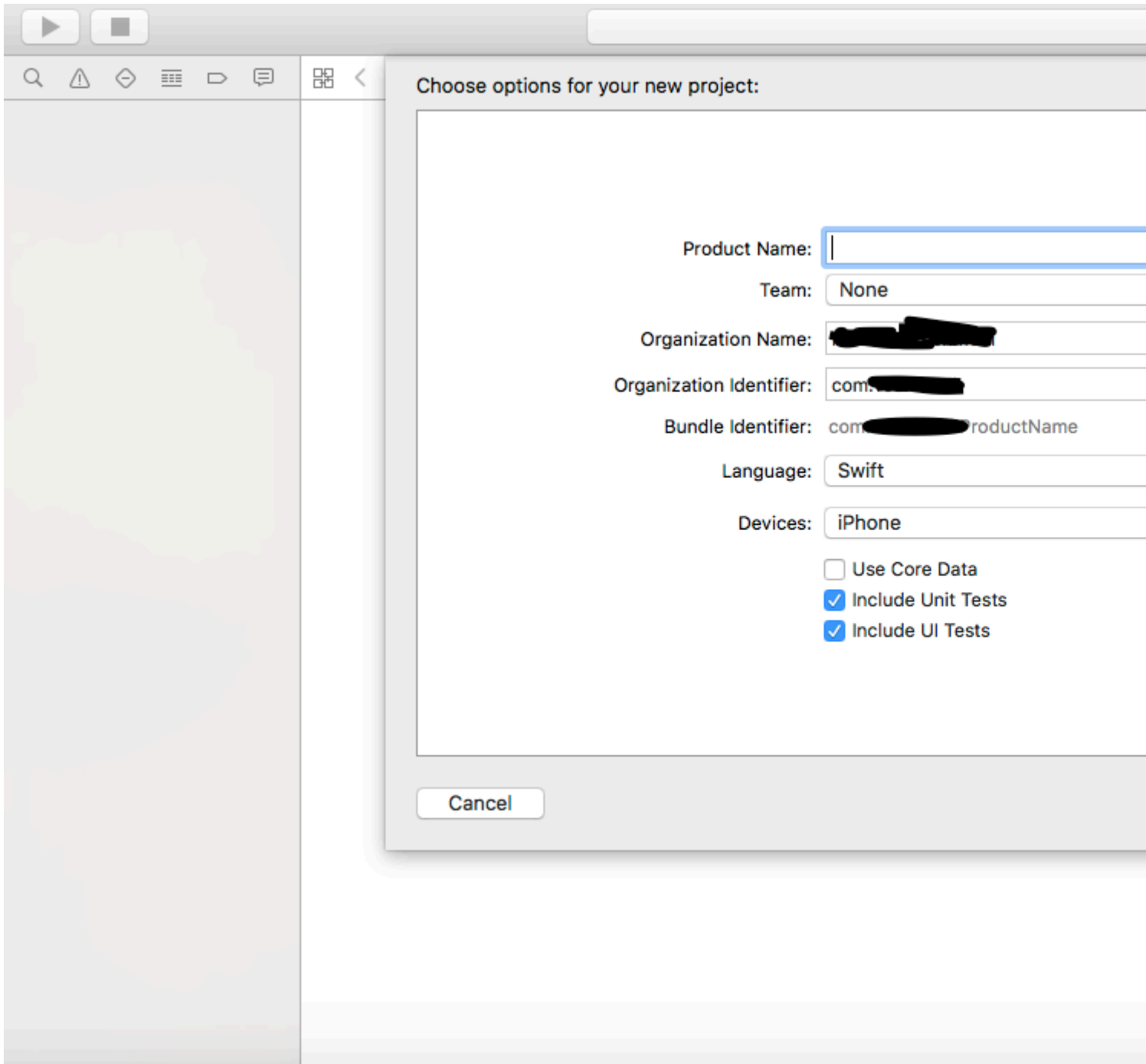
Wählen Sie dann Neues Projekt erstellen und danach das nächste Bild



Dies ist auch ein sehr wichtiger Teil von Xcode für die Auswahl unseres Projekttyps. Wir müssen unser Projekt nach OS-Typen auswählen. Es gibt fünf Arten von Optionen auf der Oberseite:

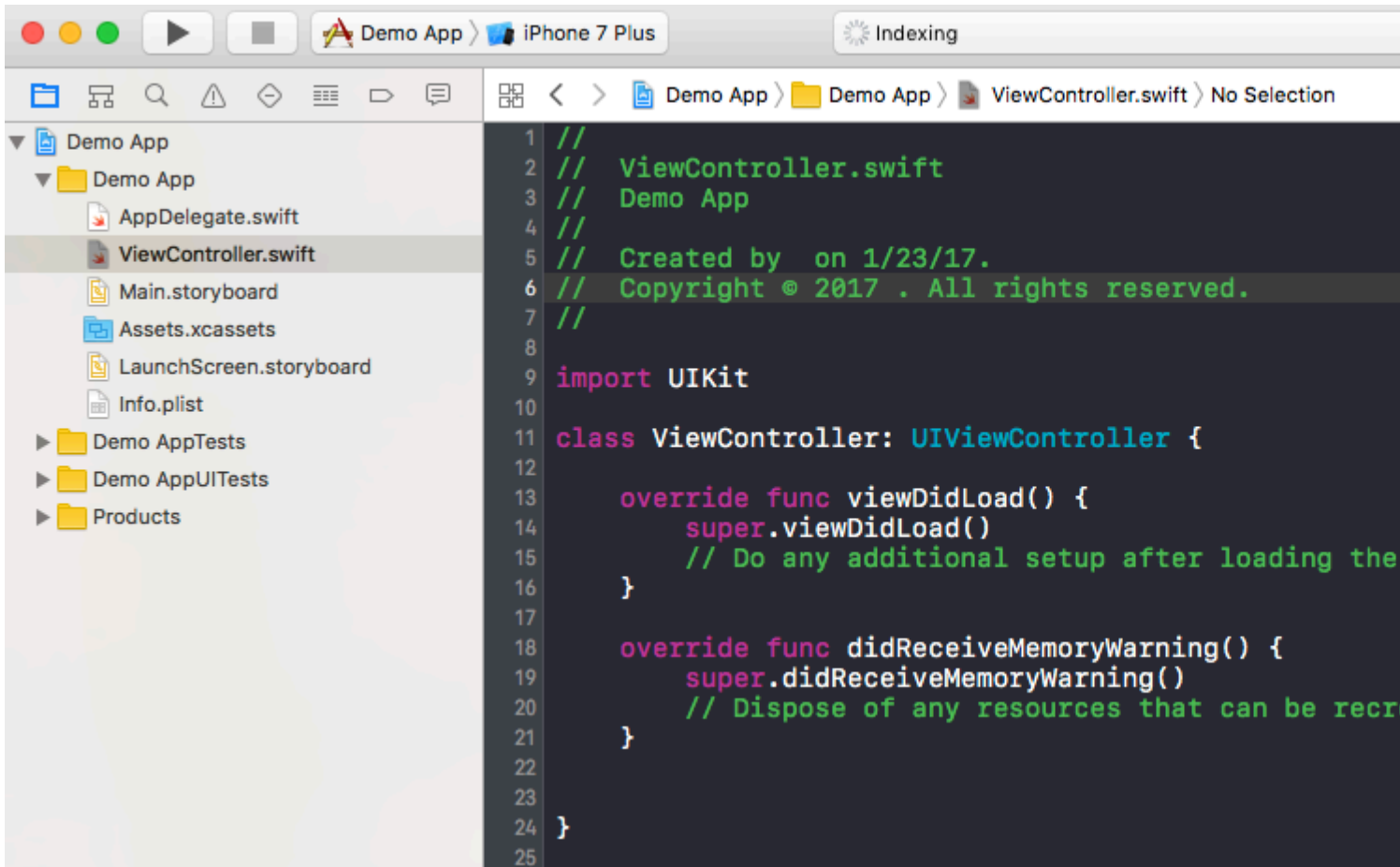
1. [iOS](#)
2. [watchOS](#)
3. [Mac OS](#)
4. Plattformübergreifend

Jetzt wählen wir die iOS-Plattform für die Entwicklung und das Erstellen eines sehr einfachen Projekts mit der Einzelansicht-Anwendungsoption:



Dann müssen wir den Produktnamen angeben. Dies stellt den Namen des Pakets und den Namen der Anwendung dar.

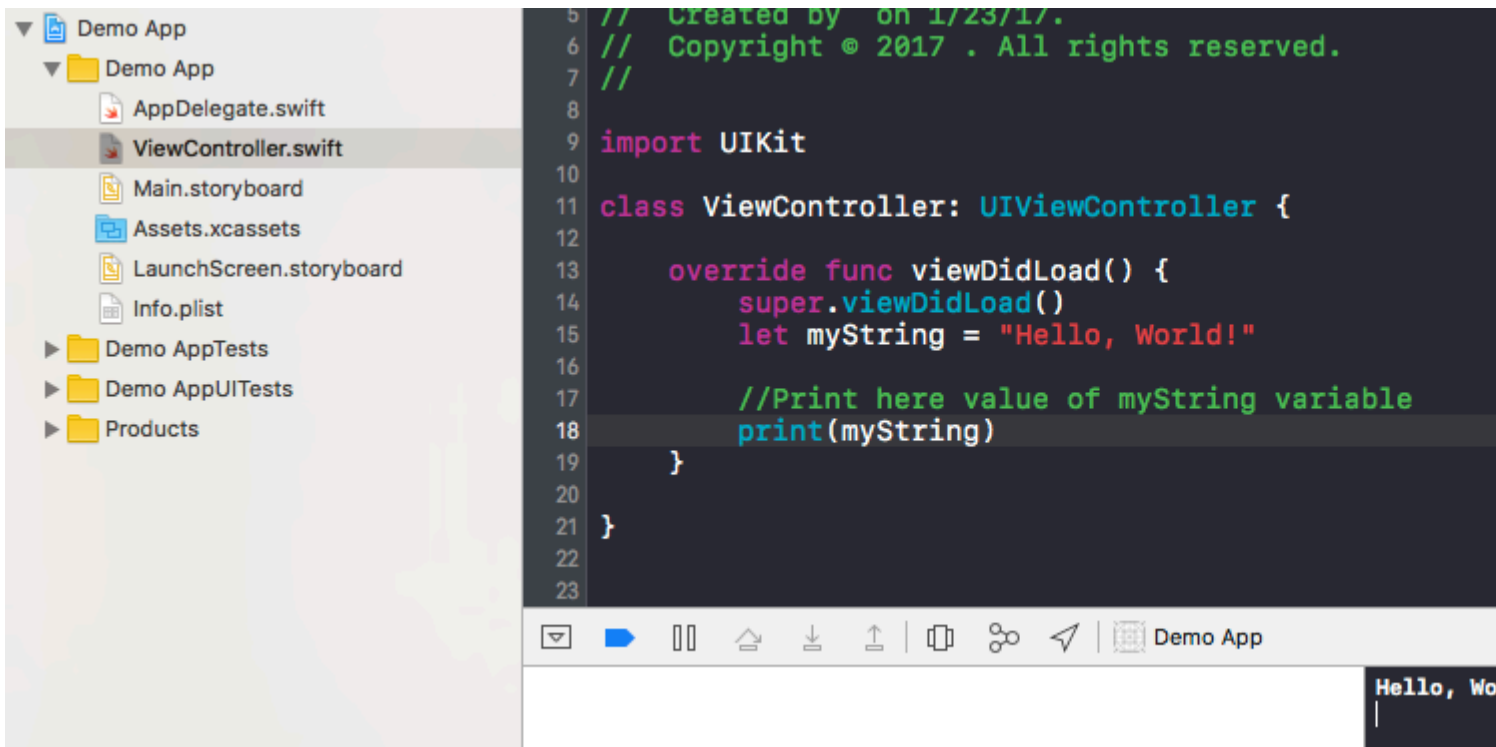
Anwendungsname können Sie später nach Ihren Anforderungen ändern. Dann müssen wir auf "Erstellen" klicken und danach sieht Ihr Bildschirm wie folgt aus:



In dieser Klasse sehen Sie, dass der Dateiname ViewController.swift und in der Klasse auch der Name ViewController lautet, der von der Superklasse UIViewController übernommen wird. Schließlich erstellen wir unsere erste Variable mit dem Namen **myString** vom Typ 'String'. Fügen Sie Folgendes unter 'super.viewDidLoad ()' hinzu

```
let myString = "Hello, World!"
```

Wir werden den Inhalt dieser Variablen ausdrucken. Wählen Sie zunächst Ihren Simulatortyp oben links im Bildschirm aus und klicken Sie auf die Schaltfläche "Ausführen".



Danach wird Ihre Ausgabe auf dem Terminal angezeigt, das sich rechts unten befindet. Herzlichen Glückwunsch, dies ist Ihr erstes Hello World-Programm in Xcode.

Erste Schritte mit iOS online lesen: <https://riptutorial.com/de/ios/topic/191/erste-schritte-mit-ios>

Kapitel 2: 3D Touch

Examples

3D Touch mit Swift

3D-Touch wurde mit dem iPhone 6s Plus eingeführt. Mit dieser neuen Schnittstellenschicht werden zwei Verhalten hinzugefügt: Peek und Pop.

Peek and Pop in aller Kürze

Peek - Drücken Sie fest

Pop - Press wirklich hart

Überprüfung auf 3D-Unterstützung

Sie sollten prüfen, ob das Gerät eine 3D-Touch-Unterstützung hat. Sie können dies tun, indem Sie den Wert der *forceTouchCapability*- Eigenschaft eines *UITraitCollection*- Objekts *überprüfen* . *UITraitCollection* beschreibt die iOS-Schnittstellenumgebung für Ihre App.

```
if (traitCollection.forceTouchCapability == .Available) {
    registerForPreviewingWithDelegate(self, sourceView: view)
}
```

Umsetzung des Delegierten

Sie müssen die beiden Methoden von *UIViewControllerPreviewingDelegate* in Ihrer Klasse implementieren. Eines der Verfahren ist für *Peek* und das andere ist für *Pop* - Verhalten.

Die für den *Peek* zu implementierende Methode ist *previewingContext* .

```
func previewingContext(previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController? {

    guard let indexPath = self.tableView.indexPathForRowAtPoint(location), cell =
self.tableView.cellForRowAtIndex(indexPath) as? <YourTableViewCell> else {
        return nil
    }

    guard let detailVC =
storyboard?.instantiateViewControllerWithIdentifier("<YourViewControllerIdentifier>") as?
<YourViewController> else {
        return nil
    }

    detailVC.peekActive = true
    previewingContext.sourceRect = cell.frame

    // Do the stuff
```

```

return detailVC
}

```

Die für den *Popup* zu implementierende Methode ist *previewingContext* . :)

```

func previewingContext (previewingContext: UIViewControllerPreviewing, commitViewController
viewControllerToCommit: UIViewController) {

    let balanceViewController = viewControllerToCommit as! <YourViewController>

    // Do the stuff

    navigationController?.pushViewController(balanceViewController, animated: true)

}

```

Wie Sie sehen, handelt es sich dabei um überladene Methoden. Sie können den 3D-Touch auf beliebige Weise verwenden, um diese Methoden zu implementieren.

Ziel c

```

//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
    (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
{
    [self registerForPreviewingWithDelegate:self sourceView:self.view];
}
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}

```

3 D Touch Objective-C-Beispiel

Ziel c

```

//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
    (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
{
    [self registerForPreviewingWithDelegate:self sourceView:self.view];
}
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
    viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        CountryCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navigationController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
    commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}

```

3D Touch online lesen: <https://riptutorial.com/de/ios/topic/6705/3d-touch>

Kapitel 3: AFNetworking

Examples

Blockieren des Beendigungsblocks für einen benutzerdefinierten Thread

Wenn AFNetworking verwendet wird, wird der Anruf in einem von AFNetworking bereitgestellten benutzerdefinierten Thread abgesetzt. Wenn der Aufruf zum Beendigungsblock zurückkehrt, wird er im Hauptthread ausgeführt.

In diesem Beispiel wird ein benutzerdefinierter Thread festgelegt, der an den Beendigungsblock gesendet wird:

AFNetworking 2.xx:

```
// Create dispatch_queue_t with your name and DISPATCH_QUEUE_SERIAL as for the flag
dispatch_queue_t myQueue = dispatch_queue_create("com.CompanyName.AppName.methodTest",
        DISPATCH_QUEUE_SERIAL);

// init AFHTTPRequestOperation of AFNetworking
operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];

// Set the FMDB property to run off the main thread
[operation setCompletionQueue:myQueue];
```

AFNetworking 3.xx:

```
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc] init];
[self setCompletionQueue:myQueue];
```

AFNetworking online lesen: <https://riptutorial.com/de/ios/topic/3002/afnetworking>

Kapitel 4: AirDrop

Examples

AirDrop

Ziel c

Airdrop kann von `UIActivityViewController` . Die `UIActivityViewController` Klasse ist ein Standard-View-Controller, der verschiedene Standarddienste bereitstellt, z. B. das Kopieren von Elementen in die Zwischenablage, das Teilen von Inhalten auf Social-Media-Sites, das Senden von Elementen über Messages, AirDrop und einige Anwendungen von Drittanbietern.

In diesem Fall senden wir ein Bild über `UIActivityViewController`

```
UIImage *hatImage = [UIImage imageNamed:@"logo.png"];
if (hatImage)//checks if the image file is not nil
{
//Initialise a UIActivityViewController
UIActivityViewController *controller = [[UIActivityViewController alloc]
initWithActivityItems:@[hatImage] applicationActivities:nil];
//Excludes following options from the UIActivityViewController menu
NSArray *excludeActivities = @[UIActivityTypePostToWeibo,UIActivityTypePrint,
UIActivityTypeMail,UIActivityTypeMessage,UIActivityTypePostToTwitter,UIActivityTypePostToFacebook,
UIActivityTypeCopyToPasteboard,UIActivityTypeAssignToContact,
UIActivityTypeSaveToCameraRoll,UIActivityTypeAddToReadingList,
UIActivityTypePostToFlickr,UIActivityTypePostToVimeo,
UIActivityTypePostToTencentWeibo];
controller.excludedActivityTypes = excludeActivities;
[self presentViewController:controller animated:YES completion:nil];
}
```

Schnell

```
if ((newImage) != nil)
{
    let activityVC = UIActivityViewController(activityItems: [newImage],
applicationActivities: nil)
    activityVC.excludedActivityTypes =[UIActivityTypeAddToReadingList]
    self.presentViewController(activityVC, animated: true, completion: nil)
}
```

AirDrop online lesen: <https://riptutorial.com/de/ios/topic/7360/airdrop>

Kapitel 5: AirPrint-Tutorial in iOS

Examples

AirPrint-Druck Banner-Text

Ziel c

Fügen Sie der Datei `ViewController.h` den Delegaten und einen Textformatierer `ViewController.h`

```
@interface ViewController : UIViewController <UIPrintInteractionControllerDelegate> {
    UISimpleTextPrintFormatter *_textFormatter;
}
```

`ViewController.m` Datei `ViewController.m` die folgenden Konstanten

```
#define DefaultFontSize 48
#define PaddingFactor 0.1f
```

Die Funktion, mit der der Text gedruckt wird, lautet wie folgt:

```
-(IBAction)print:(id)sender;
{
    /* Get the UIPrintInteractionController, which is a shared object */
    UIPrintInteractionController *controller = [UIPrintInteractionController
sharedPrintController];
    if(!controller){
        NSLog(@"Couldn't get shared UIPrintInteractionController!");
        return;
    }

    /* Set this object as delegate so you can use the
printInteractionController:cutLengthForPaper: delegate */
    controller.delegate = self;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;

    /* Use landscape orientation for a banner so the text print along the long side of the
paper. */
    printInfo.orientation = UIPrintInfoOrientationLandscape;

    printInfo.jobName = self.textField.text;
    controller.printInfo = printInfo;

    /* Create the UISimpleTextPrintFormatter with the text supplied by the user in the text
field */
    _textFormatter = [[UISimpleTextPrintFormatter alloc] initWithText:self.textField.text];

    /* Set the text formatter's color and font properties based on what the user chose */
    _textFormatter.color = [self chosenColor];
    _textFormatter.font = [self chosenFontWithSize:DefaultFontSize];
}
```



```

/* Set this UISimpleTextPrintFormatter on the controller */
controller.printFormatter = _textFormatter;

/* Set up a completion handler block. If the print job has an error before spooling, this
is where it's handled. */
void (^completionHandler)(UIPrintInteractionController *, BOOL, NSError *) =
^(UIPrintInteractionController *printController, BOOL completed, NSError *error) {
    if(completed && error)
        NSLog( @"Printing failed due to error in domain %@ with error code %lu. Localized
description: %@, and failure reason: %@", error.domain, (long)error.code,
error.localizedDescription, error.localizedFailureReason );
    };

    if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad)
        [controller presentFromRect:self.printButton.frame inView:self.view animated:YES
completionHandler:completionHandler];
    else
        [controller presentAnimated:YES completionHandler:completionHandler]; // iPhone
}

```

Die Delegierungsfunktion, die die Druckseite einrichtet: -

```

- (CGFloat)printInteractionController:(UIPrintInteractionController
*)printInteractionController cutLengthForPaper:(UIPrintPaper *)paper {

    /* Create a font with arbitrary size so that you can calculate the approximate
font points per screen point for the height of the text. */
    UIFont *font = _textFormatter.font;
    CGSize size = [self.textField.text sizeWithAttributes:@{NSFontAttributeName: font}];

    float approximateFontPointPerScreenPoint = font.pointSize / size.height;

    /* Create a new font using a size that will fill the width of the paper */
    font = [self chosenFontWithSize: paper.printableRect.size.width *
approximateFontPointPerScreenPoint];

    /* Calculate the height and width of the text with the final font size */
    CGSize finalTextSize = [self.textField.text sizeWithAttributes:@{NSFontAttributeName:
font}];

    /* Set the UISimpleTextFormatter font to the font with the size calculated */
    _textFormatter.font = font;

    /* Calculate the margins of the roll. Roll printers may have unprintable areas
before and after the cut. We must add this to our cut length to ensure the
printable area has enough room for our text. */
    CGFloat lengthOfMargins = paper.paperSize.height - paper.printableRect.size.height;

    /* The cut length is the width of the text, plus margins, plus some padding */
    return finalTextSize.width + lengthOfMargins + paper.printableRect.size.width *
PaddingFactor;
}

```

AirPrint-Tutorial in iOS online lesen: <https://riptutorial.com/de/ios/topic/7395/airprint-tutorial-in-ios>

Kapitel 6: Alamofire

Syntax

- Antwort()
- responseData ()
- responseString (Kodierung: NSStringEncoding)
- responseJSON (Optionen: NSJSONReadingOptions)
- responsePropertyList (Optionen: NSPropertyListReadOptions)

Parameter

Parameter	Einzelheiten
Methode	.OPTIONS, .GET, .HEAD, .POST, .PUT, .PATCH, .DELETE, .TRACE, .CONNECT
URLString	URLStringConvertible
Parameter	[String: AnyObject]?
Codierung	ParameterEncoding
Überschriften	[String: String]?

Examples

Eine Anfrage machen

```
import Alamofire

Alamofire.request(.GET, "https://httpbin.org/get")
```

Automatische Validierung

```
Alamofire.request("https://httpbin.org/get").validate().responseJSON { response in
switch response.result {
case .success:
    print("Validation Successful")
case .failure(let error):
    print(error)
}
}
```

Antwortbehandlung

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .responseJSON { response in
        print(response.request) // original URL request
        print(response.response) // URL response
        print(response.data) // server data
        print(response.result) // result of response serialization

        if let JSON = response.result.value {
            print("JSON: \(JSON)")
        }
    }

```

Manuelle Validierung

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .validate(statusCode: 200..<300)
    .validate(contentType: ["application/json"])
    .response { response in
        print(response)
    }

```

Antworthandler

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .validate()
    .response { request, response, data, error in
        print(request)
        print(response)
        print(data)
        print(error)
    }

```

Verkettete Antwort-Handler

```

Alamofire.request(.GET, "https://httpbin.org/get")
    .validate()
    .responseString { response in
        print("Response String: \(response.result.value)")
    }
    .responseJSON { response in
        print("Response JSON: \(response.result.value)")
    }

```

Alamofire online lesen: <https://riptutorial.com/de/ios/topic/1823/alamofire>

Kapitel 7: Ändern der Größe von UIImage

Parameter

CGInterpolationQuality	Interpolationsstufen für das Rendern eines Bildes.
Die Interpolationsqualität ist ein Grafikzustandsparameter	<pre>typedef enum CGInterpolationQuality CGInterpolationQuality;</pre>

Examples

Ändern Sie die Größe eines Bildes nach Größe und Qualität

```
- (UIImage *)drawImageBySize:(CGSize)size quality:(CGInterpolationQuality)quality  
{  
    UIGraphicsBeginImageContextWithOptions(size, NO, 0.0);  
    CGContextRef context = UIGraphicsGetCurrentContext();  
    CGContextSetInterpolationQuality(context, quality);  
    [self drawInRect: CGRectMake (0, 0, size.width, size.height)];  
    UIImage *resizedImage = UIGraphicsGetImageFromCurrentImageContext();  
    UIGraphicsEndImageContext();  
    return resizedImage;  
}
```

Ändern der Größe von UIImage online lesen: <https://riptutorial.com/de/ios/topic/6422/ändern-der-gro-e-von-uiimage>

Kapitel 8: Antrag auf Antragsbewertung / Überprüfung

Einführung

Mit iOS 10.3 müssen Sie jetzt nicht mehr zum Apple Store navigieren, um Bewertungen / Überprüfungen durchzuführen. Apple hat die `SKStoreReviewController`-Klasse im StoreKit-Framework eingeführt. In welchem Entwickler müssen Sie nur die `requestReview ()` - Klassenmethode der `SKStoreReviewController`-Klasse aufrufen, und das System übernimmt den gesamten Prozess für Sie.

Darüber hinaus können Sie weiterhin einen permanenten Link in die Einstellungen oder Konfigurationsbildschirme Ihrer App einfügen, der zu Ihrer App Store-Produktseite führt. Um automatisch zu arbeiten

Examples

Bewerten / Überprüfen Sie die iOS-Anwendung

Geben Sie einfach einen Zeilencode ein, von dem aus Sie die Anwendung bewerten oder prüfen sollen.

```
SKStoreReviewController.requestReview ()
```

Antrag auf Antragsbewertung / Überprüfung online lesen:

<https://riptutorial.com/de/ios/topic/9678/antrag-auf-antragsbewertung---uberprufung>

Kapitel 9: AppDelegate

Einführung

AppDelegate ist ein Protokoll, das Methoden definiert, die vom UIApplication-Einzelobjekt als Reaktion auf wichtige Ereignisse während der Lebensdauer einer App aufgerufen werden.

Wird normalerweise zum Ausführen von Aufgaben beim Start der Anwendung (Setup-App-Umgebung, Analysetools (z. B. Mixpanel / GoogleAnalytics / Crashlitics), DB-Stack usw.) und zum Herunterfahren (z. B. Speichern des DB-Kontextes), zum Bearbeiten von URL-Anforderungen und ähnlichen Anwendungen verwendet Aufgaben.

Examples

Alle Zustände der Anwendung durch AppDelegate-Methoden

Wenn Sie ein Update erhalten oder etwas tun möchten, bevor die App für den Benutzer verfügbar ist, können Sie die unten beschriebene Methode verwenden.

AppDidFinishLaunching

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {  
    // Write your code before app launch  
    return YES;  
}
```

Während App in den Vordergrund tritt:

```
- (void)applicationWillEnterForeground:(UIApplication *)application {  
    // Called as part of the transition from the background to the active state; here you can  
    undo many of the changes made on entering the background.  
}
```

Wenn App Launching und auch Hintergrund in den Vordergrund treten, treffen Sie die Methode unten:

```
- (void)applicationDidBecomeActive:(UIApplication *)application {  
    // Restart any tasks that were paused (or not yet started) while the application was  
    inactive. If the application was previously in the background, optionally refresh the user  
    interface.  
}
```

Während App im Hintergrund eingeben:

```
- (void)applicationDidEnterBackground:(UIApplication *)application {  
    // Use this method to release shared resources, save user data, invalidate timers, and  
    store enough application state information to restore your application to its current state in
```

```
case it is terminated later.
    // If your application supports background execution, this method is called instead of
    applicationWillTerminate: when the user quits.
}
```

Während App zurücktreten aktiv

```
- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can
    occur for certain types of temporary interruptions (such as an incoming phone call or SMS
    message) or when the user quits the application and it begins the transition to the background
    state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics
    rendering callbacks. Games should use this method to pause the game.
}
```

Während der App beenden:

```
- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if appropriate. See also
    applicationDidEnterBackground:.
}
```

AppDelegate-Rollen:

- AppDelegate enthält Ihre App - startup code .
- Es reagiert auf `key changes` des `state` Ihrer App. Insbesondere reagiert es sowohl auf temporäre Unterbrechungen als auch auf Änderungen im Ausführungsstatus Ihrer App, z. B. wenn Ihre App vom Vordergrund in den Hintergrund wechselt.
- Es `responds to notifications` die von außerhalb der App stammen, z. B. Remote-Benachrichtigungen (auch Push-Benachrichtigungen genannt), Warnmeldungen zu wenig Arbeitsspeicher, Benachrichtigungen zum Abschluss des Herunterladens usw.
- Es `determines ob state preservation` und `restoration state preservation` soll, und unterstützt bei Bedarf den Erhaltungs- und Wiederherstellungsprozess.
- Es `responds to events` , die auf die App selbst abzielen, und ist nicht spezifisch für die Ansichten Ihrer App oder die Ansichtssteuerungen. Sie können es verwenden, um die zentralen Datenobjekte Ihrer App oder Inhalte zu speichern, die nicht über einen eigenen View-Controller verfügen.

Eine URL-spezifische Ressource öffnen

Fordert den Delegaten auf, eine durch eine URL angegebene Ressource zu öffnen, und stellt ein Wörterbuch mit Startoptionen bereit.

Verwendungsbeispiel:

```
func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey : Any] = [:]) -> Bool {
    return SomeManager.shared.handle(
        url,
        sourceApplication: options[.sourceApplication] as? String,
```

```
        annotation: options[.annotation]
    )
}
```

Umgang mit lokalen und Remote-Benachrichtigungen

Verwendungsbeispiel:

```
/* Instance of your custom APNs/local notification manager */
private var pushManager: AppleNotificationManager!
```

Anmeldung:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    // Called to tell the delegate the types of notifications that can be used to get the
    user's attention
    pushManager.didRegisterSettings(notificationSettings)
}

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    // Tells the delegate that the app successfully registered with Apple Push Notification
    service (APNs)
    pushManager.didRegisterDeviceToken(deviceToken)
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    // Sent to the delegate when Apple Push Notification service cannot successfully complete
    the registration process.
    pushManager.didFailToRegisterDeviceToken(error)
}
```

Remote-Benachrichtigungen verwalten:

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
: AnyObject]) {
    // Remote notification arrived, there is data to be fetched
    // Handling it
    pushManager.handleNotification(userInfo,
                                   background: application.applicationState == .Background
    )
}
```

Umgang mit lokalen Benachrichtigungen:

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {
    pushManager.handleLocalNotification(notification, background: false)
}
```

Handlungsaktion (veraltet):


```
func application(application: UIApplication, handleActionWithIdentifier identifier: String?,
forRemoteNotification userInfo: [NSObject : AnyObject],
                    completionHandler: () -> Void) {
    pushManager.handleInteractiveRemoteNotification(userInfo, actionIdentifier: identifier,
completion: completionHandler)
}
```

AppDelegate online lesen: <https://riptutorial.com/de/ios/topic/8740/appdelegate>

Kapitel 10: App-Einreichungsprozess

Einführung

In diesem Lernprogramm werden alle Schritte beschrieben, die zum Hochladen einer iOS-App in den App Store erforderlich sind.

Examples

Einrichten von Bereitstellungsprofilen

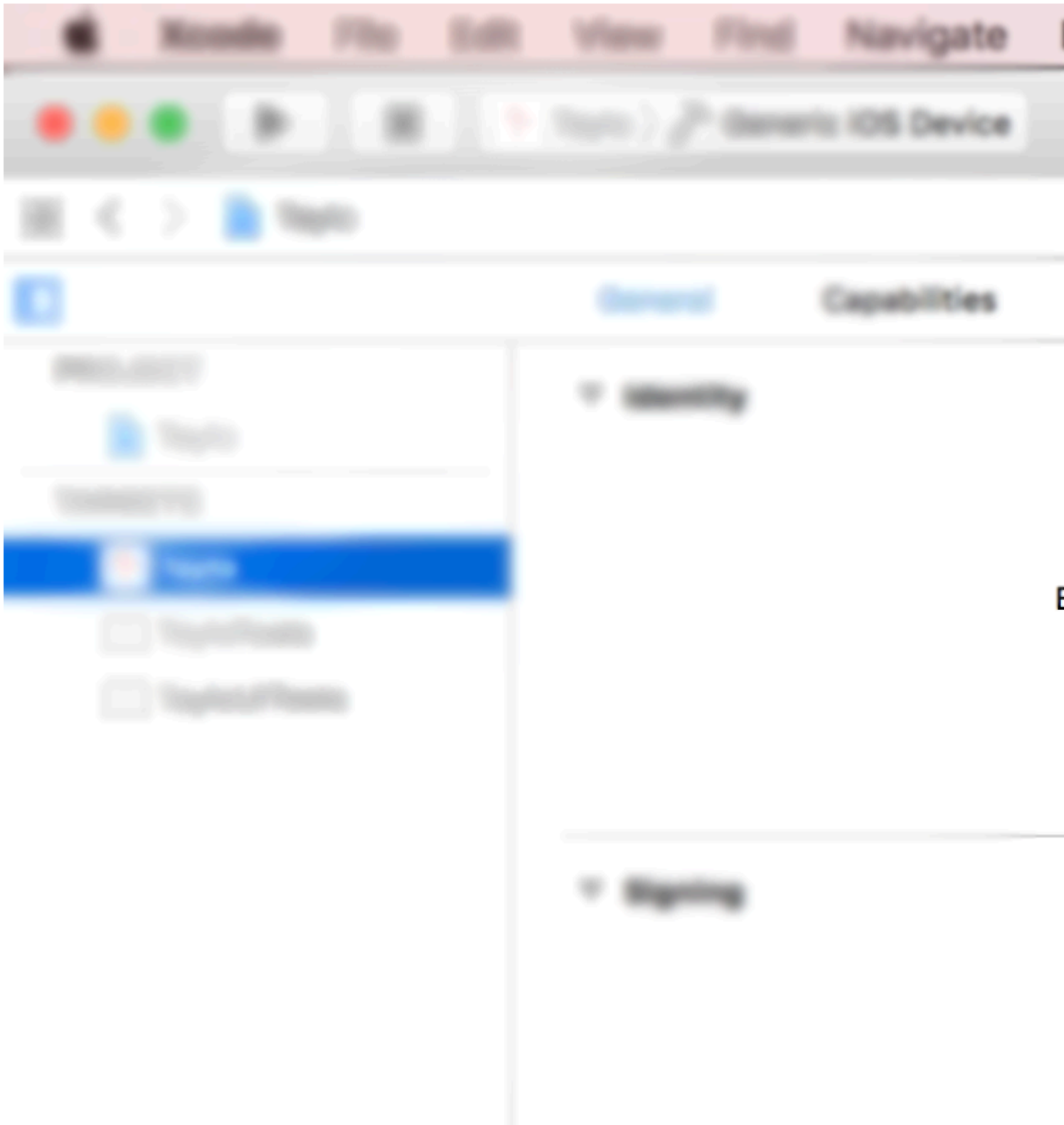
In früheren Versionen wurde das Einrichten von Bereitstellungsprofilen manuell durchgeführt. Sie generieren ein Distributionsprofil, laden es herunter und verteilen Ihre App. Dies musste für jede Entwicklungsmaschine durchgeführt werden, was extrem zeitaufwändig war. In den meisten Situationen erledigt Xcode 8 jedoch heutzutage die meiste Arbeit für Sie. Stellen Sie sicher, dass Sie sich mit dem Konto anmelden, das für die Verteilung der App verwendet wird. Wählen Sie dann unter Ziele -> Allgemein einfach "Codesignatur automatisch verwalten" aus.

▼ Signing

Automatically manage signing
Xcode will create and manage certificates for you.

Archivieren Sie den Code

Sobald alle Bereitstellungsprofile festgelegt sind, besteht der nächste Schritt beim Senden der App darin, Ihren Code zu archivieren. Wählen Sie aus der Dropdown-Liste der Geräte und Simulatoren die Option "Generisches iOS-Gerät" aus. Wählen Sie dann im Menü "Produkt" die Option "Archiv".



Wenn es sich bei der Einreichung um ein Update für die vorhandene App im Store handelt, stellen Sie sicher, dass die Build-Nummer höher als die aktuelle ist und die Versionsnummer unterschiedlich ist. Die aktuelle Anwendung hat beispielsweise die Build-Nummer 30 und das Versionskennzeichen 1.0. Das nächste Update sollte mindestens die Buildnummer 31 und das Versionskennzeichen 1.0.1 aufweisen. In den meisten Fällen sollten Sie Ihrer Version eine dritte Dezimalzahl hinzufügen, wenn Sie einige dringende Bugfixes oder kleine Patches verwenden. Die zweite Dezimalzahl ist meistens für Funktionsaktualisierungen reserviert, während die erste Dezimalzahl bei einer größeren App-Aktualisierung erhöht wird.

IPA-Datei exportieren

Sobald dies erledigt ist, können Sie Ihr Archiv im Xcode Organizer finden. Hier werden alle Ihre früheren Versionen und Archiv-Builds gespeichert und organisiert, falls Sie sie nicht löschen. Sie werden sofort einen großen blauen Knopf mit der Aufschrift "Upload to App Store ..." bemerken. In 9/10 Fällen wird dies jedoch aus verschiedenen Gründen nicht funktionieren (Xcode-Fehler meist). Umgehung besteht darin, Ihr Archiv zu exportieren und mit einem anderen Xcode-Tool namens Application Loader hochzuladen. Da das Anwendungsladeprogramm jedoch IPA-Dateien in den App Store lädt, muss das Archiv in das richtige Format exportiert werden. Dies ist eine triviale Aufgabe, die etwa eine halbe Stunde dauern kann. Klicken Sie im rechten Bereich auf die Schaltfläche "Exportieren".

Select a method for export:

- Save for iOS App Store Deployment**
Sign and package application for distribution in the iOS App Store
- Save for Ad Hoc Deployment**
Sign and package application for Ad Hoc distribution outside the App Store
- Save for Enterprise Deployment**
Sign and package application for enterprise distribution outside the App Store
- Save for Development Deployment**
Sign and package application for development distribution outside the App Store

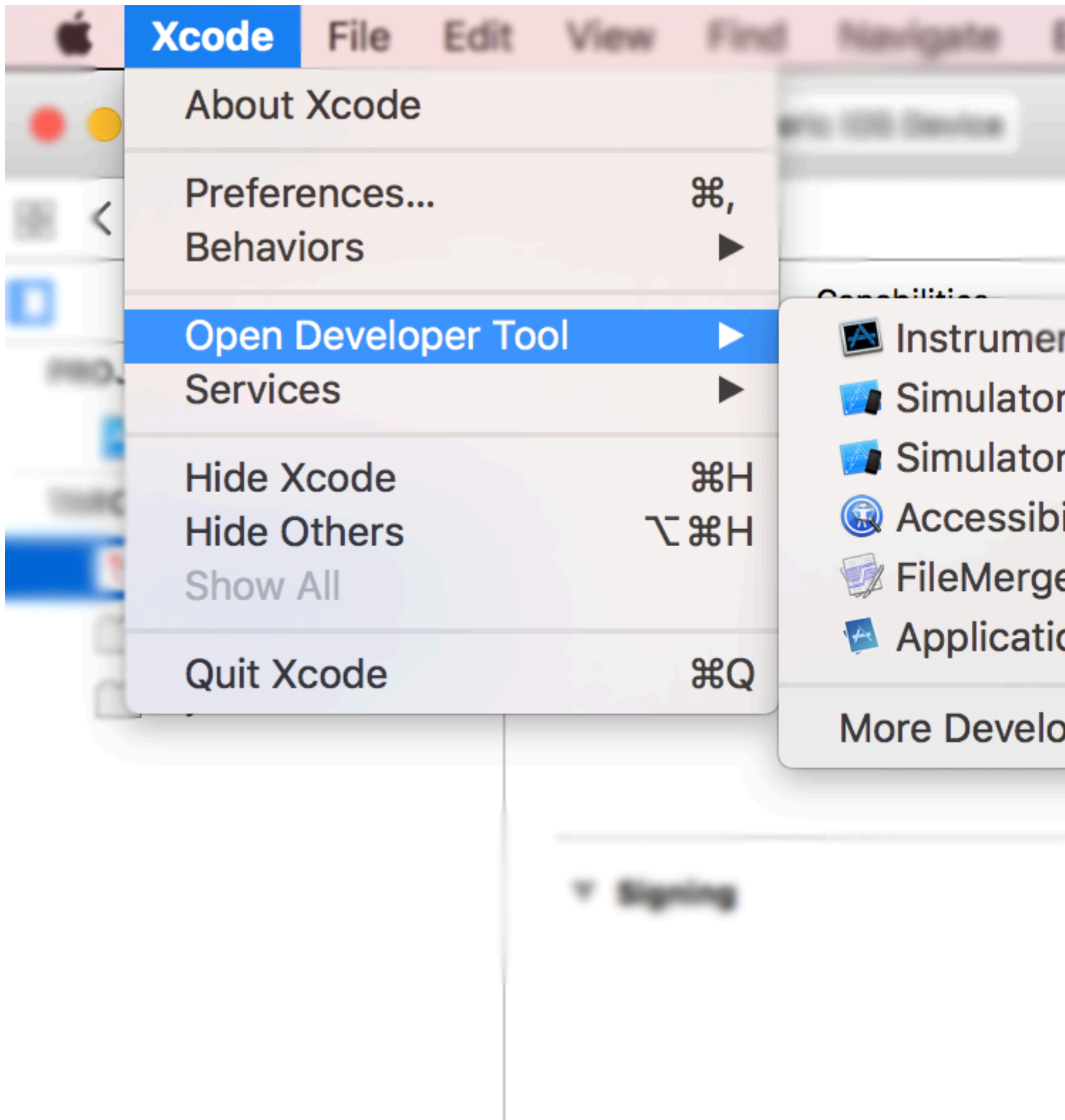
Cancel

Wenn Sie eine App in den App Store hochladen, wählen Sie die erste Option aus und klicken Sie

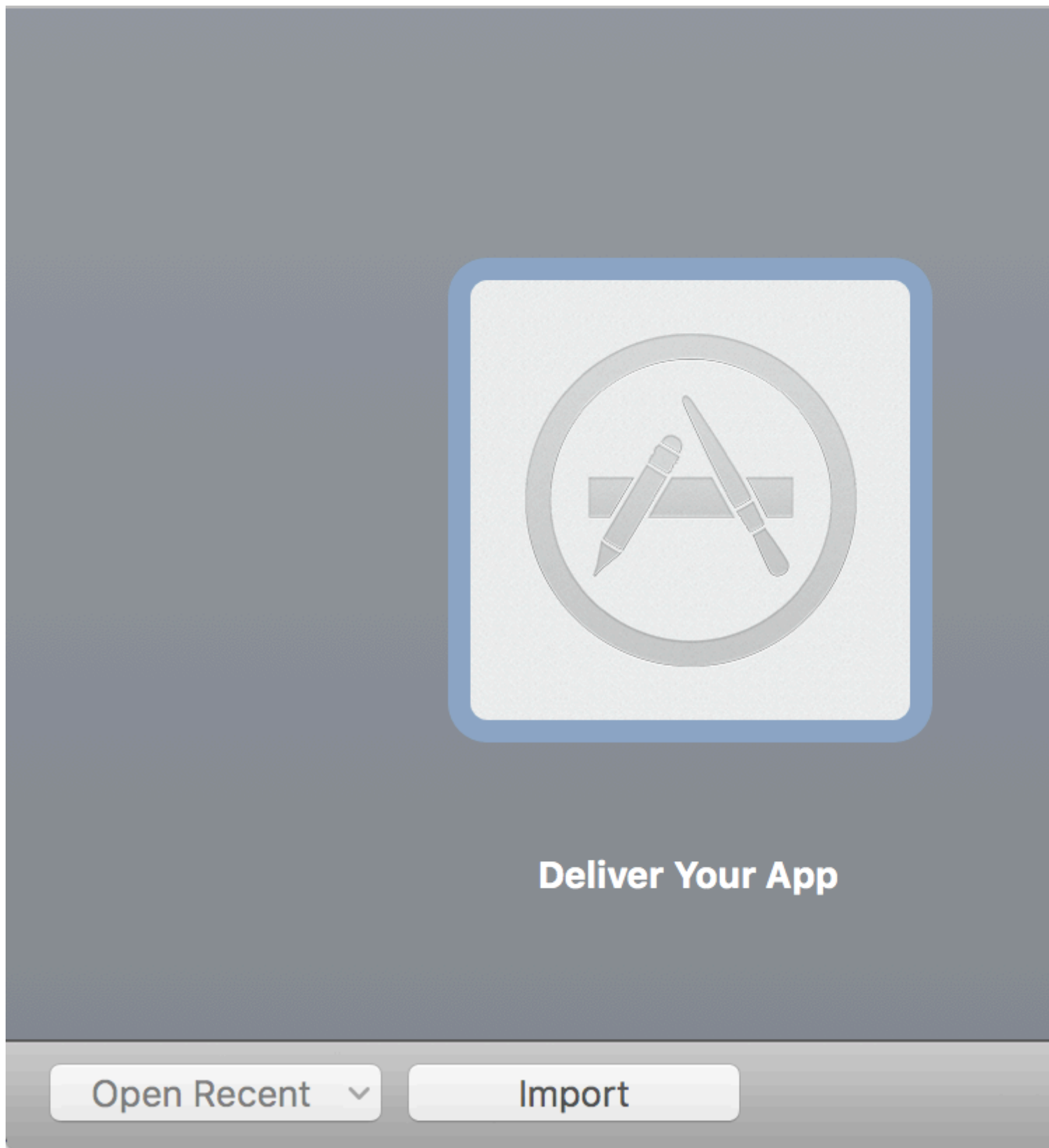
auf Weiter. Melden Sie sich an, bestätigen Sie Ihren Code erneut und holen Sie sich eine Tasse Kaffee. Sobald der Exportvorgang abgeschlossen ist, werden Sie gefragt, wo die generierte IPA-Datei gespeichert werden soll. Normalerweise ist der Desktop die bequemste Wahl.

Laden Sie die IPA-Datei mit dem Application Loader hoch

Sobald die IPA-Datei erstellt wurde, öffnen Sie Xcode, navigieren Sie zu den Entwicklerwerkzeugen und öffnen Sie Application Loader.



Wenn Sie mehrere Konten in Ihrem Xcode haben, werden Sie aufgefordert zu wählen. Wählen Sie im ersten Schritt natürlich den Code aus, den Sie zur Codesignierung verwendet haben. Wählen Sie "App ausliefern" und laden Sie den Code hoch. Nach dem Hochladen kann es bis zu einer Stunde dauern, bis Ihre Build-Liste in iTunes Connect angezeigt wird.



App-Einreichungsprozess online lesen: <https://riptutorial.com/de/ios/topic/8765/app-einreichungsprozess>

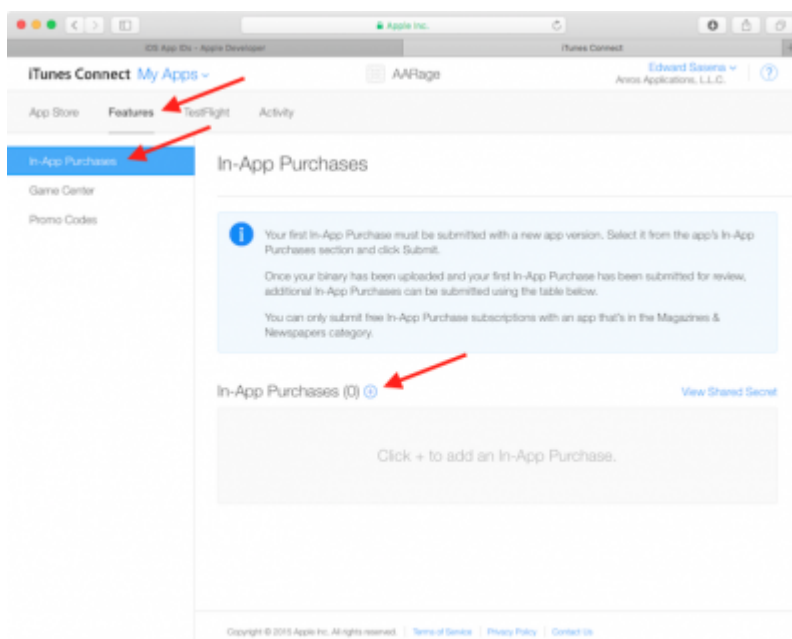
Kapitel 11: App-ID erstellen

Examples

In-App-Kaufprodukte erstellen

- Wenn Sie IAP innerhalb einer App anbieten, müssen Sie zunächst einen Eintrag für jeden Einzelkauf in iTunes Connect hinzufügen. Wenn Sie jemals eine App im Store zum Verkauf angeboten haben, ist dies ein ähnlicher Vorgang und umfasst beispielsweise die Auswahl einer Preisstufe für den Kauf. Wenn der Benutzer einen Kauf tätigt, übernimmt der App Store den komplexen Vorgang des Aufladens des iTunes-Kontos des Benutzers. Es gibt eine ganze Reihe verschiedener IAP-Typen, die Sie hinzufügen können:
 - **Verbrauchsmaterial** : Diese können mehrmals gekauft und verbraucht werden. Dies sind Dinge wie zusätzliche Leben, Spielwährung, vorübergehende Power-Ups und dergleichen.
 - **Nicht konsumierbar** : Etwas, das Sie einmal kaufen und dauerhaft voraussichtlich über zusätzliche Level und freischaltbare Inhalte verfügen.
 - **Nicht erneuerndes Abonnement** : Inhalte, die für einen bestimmten Zeitraum verfügbar sind.
 - **Auto-Renewing-Abonnement** : Ein wiederkehrendes Abonnement, z. B. ein monatliches Abonnement von raywenderlich.com.

Sie können In-App-Käufe nur für digitale Artikel und nicht für physische Waren oder Dienstleistungen anbieten. Weitere Informationen hierzu finden Sie in der vollständigen Dokumentation von Apple zum Erstellen von In-App-Kaufprodukten. Klicken Sie nun beim Anzeigen Ihres App-Eintrags in iTunes Connect auf die Registerkarte Funktionen und wählen Sie In-App-Käufe aus. Um ein neues IAP-Produkt hinzuzufügen, klicken Sie auf das + rechts von In-App-Käufen.



Es erscheint der folgende Dialog:

Select the In-App Purchase you want to create.

- Consumable**
A product that is used once, after which it becomes depleted and must be purchased again.
Example: Fish food for a fishing app.
-
- Non-Consumable**
A product that is purchased once and does not expire or decrease with use.
Example: Race track for a game app.
-
- Auto-Renewable Subscription**
A product that allows users to purchase dynamic content for a set period. This type of subscription renews automatically unless cancelled by the user.
Example: Monthly subscription for an app offering a streaming service.
-
- Non-Renewing Subscription**
A product that allows users to purchase a service with a limited duration. The content of this in-app purchase can be static. This type of subscription does not renew automatically.
Example: Annual subscription to a catalog of archived articles.

[Learn more about In-App Purchases.](#)

Cancel

Create

Wenn ein Benutzer in Ihrer App einen Raserei-Comic kauft, möchten Sie, dass er immer Zugriff darauf hat. Wählen Sie also Nicht-Verbrauchsmaterial und klicken Sie auf Erstellen. Füllen Sie als Nächstes die Details für den IAP aus:

- **Referenzname** : Ein Spitzname, der das IAP in iTunes Connect identifiziert. Dieser Name erscheint nirgendwo in der App. Der Titel des Comics, den Sie mit diesem Kauf freischalten werden, lautet „**Girlfriend of Drummer**“ . Geben Sie diesen hier ein.
- **Produkt-ID** : Dies ist eine eindeutige Zeichenfolge, die den IAP identifiziert. Normalerweise beginnen Sie am besten mit der Bundle-ID und fügen dann einen eindeutigen Namen hinzu, der für diesen käuflichen Artikel spezifisch ist. Für dieses Tutorial müssen Sie unbedingt „GirlfriendOfDrummerRage“ anhängen, da dies später in der App verwendet wird, um den Comic nachzuschalten. Zum Beispiel:
com.theNameYouPickedEarlier.Rage.GirlFriendOfDrummerRage.
- **Zum Verkauf** freigegeben: Aktiviert oder deaktiviert den Verkauf des IAP. Du möchtest es aktivieren!
- **Preisstufe** : Die Kosten des IAP. Wählen Sie Stufe 1.

Scrollen Sie nun zum Abschnitt "Lokalisierungen" und beachten Sie, dass ein Standardeintrag für Englisch (USA) vorhanden ist. Geben Sie als Anzeigenname und Beschreibung „Freundin des Schlagzeugers“ ein. Klicken Sie auf Speichern. Großartig! Sie haben Ihr erstes IAP-Produkt erstellt.

Localizations ⊕

English (U.S.)

Display Name ?

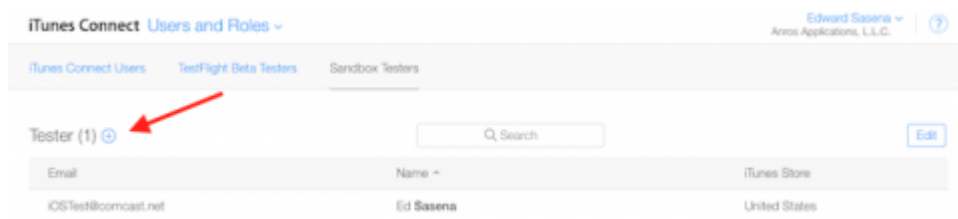
Description ?

234

Es ist noch ein weiterer Schritt erforderlich, bevor Sie in Code eintauchen können. Beim Testen von In-App-Käufen in einem Entwicklungs-Build einer App stellt Apple eine Testumgebung bereit, in der Sie Ihre IAP-Produkte "kaufen" können, ohne finanzielle Transaktionen zu erstellen.

Sandbox-Benutzer erstellen

Klicken Sie in iTunes Connect in der oberen linken Ecke des Fensters auf iTunes Connect, um zum Hauptmenü zurückzukehren. Wählen Sie Benutzer und Rollen aus und klicken Sie auf die Registerkarte Sandbox-Tester. Klicken Sie auf + neben dem Titel „Tester“.



Füllen Sie die Informationen aus und klicken Sie auf Speichern, wenn Sie fertig sind. Sie können sich einen Vor- und Nachnamen für Ihren Testbenutzer festlegen, aber die E-Mail-Adresse muss eine echte E-Mail-Adresse sein, da eine Bestätigung von Apple an die Adresse gesendet wird. Wenn Sie diese E-Mail erhalten, klicken Sie auf den Link, um Ihre Adresse zu bestätigen. Die E-Mail-Adresse, die Sie eingeben, sollte auch NICHT mit einem Apple ID-Konto verknüpft sein. Hinweis: Wenn Sie über ein Google Mail-Konto verfügen, können Sie einfach einen Adressalias verwenden, anstatt ein neues Konto erstellen zu müssen

App-ID erstellen online lesen: <https://riptutorial.com/de/ios/topic/10854/app-id-erstellen>

Kapitel 12: App-Transportsicherheit (ATS)

Parameter

Parameter	Einzelheiten
<code>NSAppTransportSecurity</code>	ATS konfigurieren
<code>NSAllowsArbitraryLoads</code>	Stellen Sie <code>YES</code> , um ATS überall zu deaktivieren. In iOS 10 und höher und MacOS 10.12 und höher wird der Wert dieses Schlüssels ignoriert, wenn in der Info.plist-Datei Ihrer App einer der folgenden Schlüssel vorhanden ist: <code>NSAllowsArbitraryLoadsInMedia</code> , <code>NSAllowsArbitraryLoadsInWebContent</code> , <code>NSAllowsLocalNetworking</code>
<code>NSAllowsArbitraryLoadsInMedia</code>	Stellen Sie <code>YES</code> , um den ATS für Medien zu deaktivieren, die mit APIs aus dem AV Foundation Framework geladen wurden. (iOS 10+, MacOS 10.12+)
<code>NSAllowsArbitraryLoadsInWebContent</code>	Setzen Sie diese Option auf <code>YES</code> , um ATS in den <code>WKWebView</code> Ihrer App (<code>WKWebView</code> , <code>UIWebView</code> , <code>WebView</code>) zu <code>UIWebView</code> , ohne dass Ihre <code>NSURLSession</code> -Verbindungen beeinträchtigt werden. (iOS 10+, MacOS 10.12+)
<code>NSAllowsLocalNetworking</code>	Setzen Sie diese <code>YES</code> auf <code>YES</code> , um Verbindungen zu nicht qualifizierten Domänen und zu lokalen Domänen zu deaktivieren. (iOS 10+, MacOS 10.12+)
<code>NSExceptionDomains</code>	Konfigurieren Sie Ausnahmen für bestimmte Domänen
<code>NSIncludesSubdomains</code>	Stellen Sie <code>YES</code> , um die Ausnahmen auf alle Unterdomänen der ausgewählten Domäne anzuwenden.
<code>NSRequiresCertificateTransparency</code>	Stellen Sie <code>YES</code> ein, um gültige, signierte CT-Zeitstempel (Certificate Transparency) aus bekannten CT-Protokollen für Serverzertifikate (X.509) in einer Domäne vorzulegen. (iOS 10+, MacOS 10.12+)

Parameter	Einzelheiten
<code>NSExceptionAllowsInsecureHTTPLoads</code>	Stellen Sie <code>YES</code> , um HTTP für die ausgewählte Domäne zuzulassen.
<code>NSExceptionRequiresForwardSecrecy</code>	Voreinstellung auf <code>YES</code> ; Wählen Sie <code>NO</code> , um die Forward Secrecy zu deaktivieren und weitere Verschlüsselungen zu akzeptieren.
<code>NSExceptionMinimumTLSVersion</code>	TLSv1.2 ; Mögliche Werte sind: TLSv1.0 , TLSv1.1 , TLSv1.2
<code>NSThirdPartyExceptionAllowsInsecureHTTPLoads</code>	Ähnlich wie <code>NSExceptionAllowsInsecureHTTPLoads</code> , jedoch für Domänen, auf die Sie keinen Einfluss haben
<code>NSThirdPartyExceptionRequiresForwardSecrecy</code>	Ähnlich wie <code>NSExceptionRequiresForwardSecrecy</code> , aber für Domänen, auf die Sie keinen Einfluss haben
<code>NSThirdPartyExceptionMinimumTLSVersion</code>	Ähnlich wie <code>NSExceptionMinimumTLSVersion</code> , aber für Domänen, auf die Sie keinen Einfluss haben

Bemerkungen

Die [App Transport Security](#) ist eine Sicherheitsfunktion in iOS und macOS. Es verhindert, dass Apps ungesicherte Verbindungen aufbauen. Standardmäßig können Apps nur sichere HTTPS-Verbindungen verwenden.

Wenn eine App über HTTP eine Verbindung zu einem Server herstellen muss, müssen Ausnahmen in der `Info.plist` definiert werden. (Weitere Informationen dazu finden Sie in den Beispielen.)

Hinweis: Im Jahr 2017 wird Apple ATS durchsetzen. Das bedeutet, dass Sie keine Apps mehr hochladen können, für die in der `Info.plist` ATS-Ausnahmen definiert `Info.plist` . Wenn Sie gute Argumente angeben können, warum Sie HTTP verwenden müssen, können Sie sich an Apple wenden. In diesem Fall können Sie Ausnahmen definieren. (Quelle: [WWDC 2016 - Sitzung 706](#))

Weitere Informationen zur App Transport Security-Konfiguration finden Sie in der [CocoaKeys-Dokumentation](#) .

Examples

Laden Sie den gesamten HTTP-Inhalt

Apple hat ATS mit iOS 9 als neue Sicherheitsfunktion eingeführt, um die Privatsphäre und

Sicherheit zwischen Apps und Webdiensten zu verbessern. ATS schlägt standardmäßig alle Nicht-HTTPS-Anforderungen fehl. Dies kann zwar für Produktionsumgebungen sehr schön sein, kann aber beim Testen lästig sein.

ATS wird in der `Info.plist` Datei des `NSAppTransportSecurity` mit dem `NSAppTransportSecurity` Wörterbuch (`App Transport Security Settings` im Xcode `Info.plist`-Editor) konfiguriert. Um den gesamten HTTP-Inhalt zuzulassen, fügen `Allow Arbitrary Loads` Boolean `Allow Arbitrary Loads` (`NSAllowsArbitraryLoads`) hinzu und setzen Sie ihn auf `YES` . Dies wird nicht für Produktionsanwendungen empfohlen. Wenn HTTP-Inhalt erforderlich ist, wird empfohlen, ihn stattdessen selektiv zu aktivieren.

HTTP-Inhalt selektiv laden

Ähnlich wie das Aktivieren des gesamten HTTP-Inhalts erfolgt die gesamte Konfiguration unter den `App Transport Security Settings` . Fügen Sie das Wörterbuch `Exception Domains` (`NSExceptionDomains`) zu den ATS-Einstellungen der obersten Ebene hinzu.

Fügen Sie für jede Domäne ein Wörterbuchelement zu den Ausnahmedomänen hinzu, wobei der Schlüssel die betreffende Domäne ist. Setzen Sie `NSExceptionAllowsInsecureHTTPLoads` auf `YES` , um die HTTPS-Anforderung für diese Domäne zu deaktivieren.

Endpunkte erfordern SSL

In iOS 9 eingeführt, müssen alle Endpunkte der HTTPS-Spezifikation entsprechen. Alle Endpunkte, die kein SSL verwenden, schlagen mit einer Warnung im Konsolenprotokoll fehl. Ihrer Bewerbung scheint es, dass die Internetverbindung fehlgeschlagen ist.

So konfigurieren Sie Ausnahmen: Fügen Sie Folgendes in Ihre `Info.plist`-Datei ein:

1. Erlauben Sie bestimmte Domäne (testdomain.com) **nur**:

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
  <key>testdomain.com</key>
  <dict>
    <key>NSIncludesSubdomains</key>
    <true/>
    <key>NSExceptionAllowsInsecureHTTPLoads</key>
    <true/>
  </dict>
</dict>
</dict>
```

Der Schlüssel, der ein solches Verhalten zulässt, ist `NSExceptionAllowsInsecureHTTPLoads` . In diesem Fall lässt die App nur eine HTTP-Verbindung zur angegebenen Domäne (testdomain.com) zu und blockiert alle anderen HTTP-Verbindungen.

Der Schlüssel `NSIncludesSubdomains` gibt an, dass auch alle **Unterdomänen** der angegebenen Domäne (testdomain.com) zulässig sein sollen.

2. Erlaube eine Domain:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```

In diesem Fall lässt die App eine HTTP-Verbindung zu einer **beliebigen** Domäne zu. Ab dem 1. Januar 2017 führt die Verwendung dieses Flags zu einer gründlichen Überprüfung des App Store. Die App-Entwickler müssen daher erklären, warum sie diese Ausnahme überhaupt verwenden müssen. Mögliche Erklärungen sind:

- Eine Anwendung, die verschlüsselten Medieninhalt lädt, der keine personalisierten Informationen enthält.
- Verbindungen zu Geräten, die nicht aktualisiert werden können, um sichere Verbindungen zu verwenden.
- Verbindung zu einem Server, der von einer anderen Entität verwaltet wird und keine sicheren Verbindungen unterstützt.

App-Transportsicherheit (ATS) online lesen: <https://riptutorial.com/de/ios/topic/5435/app-transport-sicherheit--ats->

Kapitel 13: App-weiten Operationen

Examples

Holen Sie sich den besten UIViewController

Um den besten `UIViewController` zu erhalten, ist es üblich, den `RootViewController` Ihres aktiven `UIWindow` . Ich habe dafür eine Erweiterung geschrieben:

```
extension UIApplication {  
  
    func topViewController(_ base: UIViewController? =  
        UIApplication.shared.keyWindow?.rootViewController) -> UIViewController {  
  
        if let nav = base as? UINavigationController {  
            return topViewController(nav.visibleViewController)  
        }  
  
        if let tab = base as? UITabBarController {  
            if let selected = tab.selectedViewController {  
                return topViewController(selected)  
            }  
        }  
  
        if let presented = base?.presentedViewController {  
            return topViewController(presented)  
        }  
  
        return base!  
    }  
}
```

Systemereignisse abfangen

Mit dem `NotificationCenter` von iOS, das sehr leistungsfähig sein kann, können Sie bestimmte App-weite Ereignisse abfangen:

```
NotificationCenter.default.addObserver(  
    self,  
    selector: #selector(ViewController.do(_:)),  
    name: NSNotification.Name.UIApplicationDidBecomeActive,  
    object: nil)
```

Sie können für viele weitere Veranstaltungen anmelden, nur um einen Blick

<https://developer.apple.com/reference/foundation/nsnotification.name> .

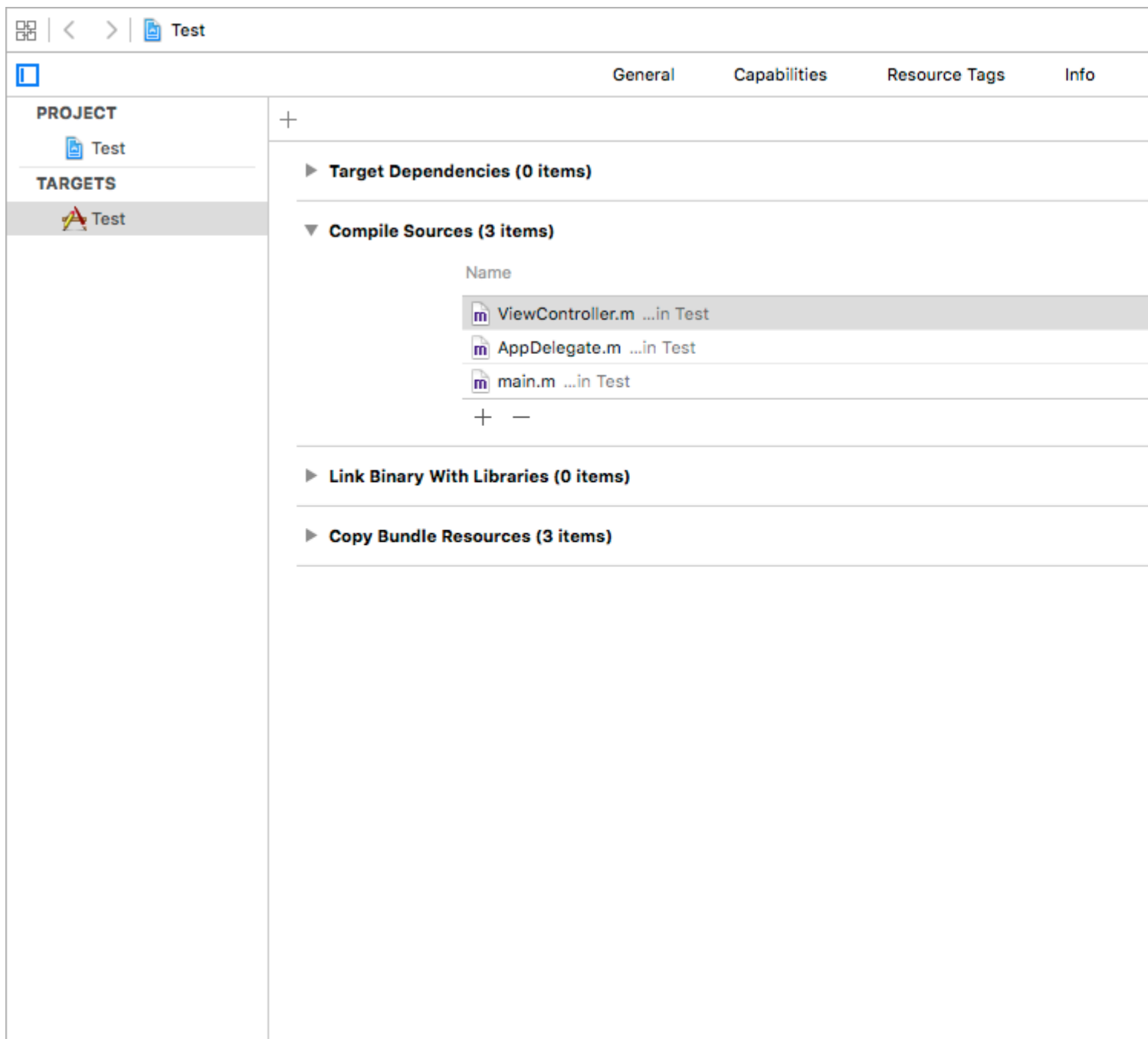
App-weiten Operationen online lesen: <https://riptutorial.com/de/ios/topic/7188/app-weiten-operationen>

Kapitel 14: ARC (automatische Referenzzählung)

Examples

Aktivieren / Deaktivieren von ARC für eine Datei

ARC kann für einzelne Dateien deaktiviert werden, indem für jede Datei das Compiler-Flag `-fno-objc-arc` . Umgekehrt kann es unter *Ziele* ▶ Buildphasen ▶ Compile Sources hinzugefügt werden



ARC (automatische Referenzzählung) online lesen: <https://riptutorial.com/de/ios/topic/4150/arc-automatische-referenzzahlung>

Kapitel 15: attributedText in UILabel

Einführung

Der aktuell gestylte Text, der vom Label angezeigt wird.

Sie können `HTML` Text in `UILabel` indem Sie die Eigenschaft `attributedText` oder einen benutzerdefinierten einzelnen `UILabel` Text mit anderen Eigenschaften verwenden

Examples

HTML-Text in UILabel

```
NSString * htmlString = @"<html><body> <b> Example bold text in HTML </b> </body></html>";
NSAttributedString * attrStr = [[NSAttributedString alloc] initWithData:[htmlString
dataUsingEncoding:NSUTF8StringEncoding] options:@{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType } documentAttributes:nil error:nil];

UILabel * yourLabel = [[UILabel alloc] init];
yourLabel.attributedText = attrStr;
```

Legen Sie eine andere Eigenschaft für Text in einem einzelnen UILabel fest

Der erste Schritt, den Sie `NSMutableAttributedString` müssen, ist das Erstellen eines `NSMutableAttributedString` Objekts. Der Grund, warum wir einen `NSMutableAttributedString` anstelle von `NSAttributedString` besteht darin, dass wir damit String anhängen können.

```
NSString *fullStr = @"Hello World!";
NSMutableAttributedString *attString =[[NSMutableAttributedString
alloc]initWithString:fullStr];

// Finding the range of text.
NSRange rangeHello = [fullStr rangeOfString:@"Hello"];
NSRange rangeWorld = [fullStr rangeOfString:@"World!"];

// Add font style for Hello
[attString addAttribute: NSFontAttributeName
                 value: [UIFont fontWithName:@"Copperplate" size:14]
                 range: rangeHello];
// Add text color for Hello
[attString addAttribute: NSForegroundColorAttributeName
                 value: [UIColor blueColor]
                 range: rangeHello];

// Add font style for World!
[attString addAttribute: NSFontAttributeName
                 value: [UIFont fontWithName:@"Chalkduster" size:20]
                 range: rangeWorld];
// Add text color for World!
[attString addAttribute: NSForegroundColorAttributeName
                 value: [UIColor colorWithRed:(66.0/255.0) green:(244.0/255.0)
```

```
blue:(197.0/255.0) alpha:1]
        range: rangeWorld];

// Set it to UILabel as attributedText
UILabel * yourLabel = [[UILabel alloc] initWithFrame:CGRectMake(10, 150, 200, 100)];
yourLabel.attributedText = attString;
[self.view addSubview:yourLabel];
```

Ausgabe :



HELLO World!

attributedString in UILabel online lesen: <https://riptutorial.com/de/ios/topic/10927/attributedtext-in-UILabel>

Kapitel 16: Auf Netzwerkverbindung prüfen

Bemerkungen

Der Quellcode für `Reachability.h` und `Reachability.m` kann auf dem Apple-Entwickler - Dokumentation zu finden [Website](#) .

Vorsichtsmaßnahmen

Im Gegensatz zu anderen Plattformen muss Apple noch einen Standardsatz von APIs bereitstellen, um den Netzwerkstatus eines iOS-Geräts zu ermitteln und nur die oben genannten Codebeispiele anzubieten. Die Quelldatei ändert sich mit der Zeit, wird jedoch nach dem Import in ein App-Projekt von den Entwicklern selten aktualisiert.

Aus diesem Grund neigen die meisten App-Entwickler dazu, eine der vielen von Github / [Cocoapod](#) verwalteten Bibliotheken zu verwenden, um erreichbar zu sein.

Apple empfiehlt auch, für im Auftrag des Benutzers gemacht Anfragen, dass Sie [immer eine Verbindung versuchen, bevor Erreichbarkeits / SCNetworkReachability mit dem Versagen zu diagnostizieren oder zu warten](#) , bis die [Verbindung zurückzukehren](#) .

Examples

Erreichbarkeitslistener erstellen

Die [Reachability](#)- Klasse von Apple überprüft regelmäßig den Netzwerkstatus und warnt die Beobachter über Änderungen.

```
Reachability *internetReachability = [Reachability reachabilityForInternetConnection];
[internetReachability startNotifier];
```

Beobachter zu Netzwerkänderungen hinzufügen

[Reachability](#) verwendet `NSNotification` Nachrichten, um Beobachter darauf aufmerksam zu machen, wenn sich der Netzwerkstatus geändert hat. Ihre Klasse muss Beobachter werden.

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reachabilityChanged:) name:kReachabilityChangedNotification object:nil];
```

Implementieren Sie an anderer Stelle in Ihrer Klasse die Methodensignatur

```
- (void) reachabilityChanged:(NSNotification *)note {
    //code which reacts to network changes
}
```

Warnung, wenn das Netzwerk nicht verfügbar ist

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    if (netStatus == NotReachable) {
        NSLog(@"Network unavailable");
    }
}
```

Warnung, wenn die Verbindung zu einem WIFI- oder Mobilfunknetz wird

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    switch (netStatus) {
        case NotReachable:
            NSLog(@"Network unavailable");
            break;
        case ReachableViaWWAN:
            NSLog(@"Network is cellular");
            break;
        case ReachableViaWiFi:
            NSLog(@"Network is WIFI");
            break;
    }
}
```

Überprüfen Sie, ob eine Verbindung zum Netzwerk besteht

Schnell

```
import SystemConfiguration

/// Class helps to code reuse in handling internet network connections.
class NetworkHelper {

    /**
     Verify if the device is connected to internet network.
     - returns: true if is connected to any internet network, false if is not
     connected to any internet network.
     */
    class func isConnectedToNetwork() -> Bool {
        var zeroAddress = sockaddr_in()

        zeroAddress.sin_len = UInt8(sizeofValue(zeroAddress))
        zeroAddress.sin_family = sa_family_t(AF_INET)

        let defaultRouteReachability = withUnsafePointer(&zeroAddress) {
            SCNNetworkReachabilityCreateWithAddress(nil, UnsafePointer($0))
        }

        var flags = SCNNetworkReachabilityFlags()
```

```

    if !SCNetworkReachabilityGetFlags(defaultRouteReachability!, &flags) {
        return false
    }

    let isReachable = (flags.rawValue & UInt32(kSCNetworkFlagsReachable)) != 0
    let needsConnection = (flags.rawValue & UInt32(kSCNetworkFlagsConnectionRequired)) != 0

    return (isReachable && !needsConnection)
}

if NetworkHelper.isConnectedToNetwork() {
    // Is connected to network
}

```

Ziel c:

Wir können die Netzwerkkonnektivität innerhalb weniger Codezeilen prüfen:

```

- (BOOL)isConnectedToNetwork
{
    Reachability *networkReachability = [Reachability reachabilityForInternetConnection];
    NetworkStatus networkStatus = [networkReachability currentReachabilityStatus];
    if (networkStatus == NotReachable)
    {
        NSLog(@"There IS NO internet connection");
        return false;
    } else
    {
        NSLog(@"There IS internet connection");
        return true;
    }
}

```

Auf Netzwerkverbindung prüfen online lesen: <https://riptutorial.com/de/ios/topic/704/auf-netzwerkverbindung-prufen>

Kapitel 17: Automatisches Layout

Einführung

Das automatische Layout berechnet dynamisch die Größe und Position aller Ansichten in Ihrer Ansichtshierarchie basierend auf den Einschränkungen, die in diesen Ansichten platziert sind.

[Quelle](#)

Syntax

- `NSLayoutConstraint` (item: Any, Attribut: `NSLayoutConstraintAttribute`, relatedBy: `NSLayoutConstraintRelation`, toItem: Any?, Attribut: `NSLayoutConstraintAttribute`, Multiplikator: `CGFloat`, Konstante: `CGFloat`) // Erstellen Sie programmgesteuert eine Kontraint

Examples

Festlegen von Einschränkungen programmgesteuert

Beispiel für Boilerplate-Code

```
override func viewDidLoad() {
    super.viewDidLoad()

    let myView = UIView()
    myView.backgroundColor = UIColor.blueColor()
    myView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(myView)

    // Add constraints code here
    // ...
}
```

In den folgenden Beispielen ist `Anchor Style` die bevorzugte Methode gegenüber `NSLayoutConstraint Style`. Sie ist jedoch nur unter iOS 9 verfügbar. Wenn Sie also iOS 8 unterstützen, sollten Sie immer noch `NSLayoutConstraint Style` verwenden.

Pinning

Ankerart

```
let margins = view.layoutMarginsGuide
myView.leadingAnchor.constraintEqualToAnchor(margins.leadingAnchor, constant: 20).active = true
```

- Neben `leadingAnchor` gibt es auch `trailingAnchor`, `topAnchor` und `bottomAnchor`.

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Leading, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: view, attribute: NSLayoutConstraintAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0).active = true
```

- Zusätzlich zum `.Leading` gibt es auch `.Trailing`, `.Top` und `.Bottom`.
- Neben `.LeadingMargin` gibt es auch `.TrailingMargin`, `.TopMargin` und `.BottomMargin`.

Visual Format Sprachstil

```
NSLayoutConstraint.constraintsWithVisualFormat("H:|-20-[myViewKey]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Breite und Höhe

Ankerart

```
myView.widthAnchor.constraintEqualToAnchor(nil, constant: 200).active = true
myView.heightAnchor.constraintEqualToAnchor(nil, constant: 100).active = true
```

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Width, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: nil, attribute: NSLayoutConstraintAttribute.NotAnAttribute, multiplier:
1, constant: 200).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Height, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: nil, attribute: NSLayoutConstraintAttribute.NotAnAttribute, multiplier:
1, constant: 100).active = true
```

Visual Format Sprachstil

```
NSLayoutConstraint.constraintsWithVisualFormat("H:[myViewKey(200)]", options: [], metrics:
nil, views: ["myViewKey": myView])
NSLayoutConstraint.constraintsWithVisualFormat("V:[myViewKey(100)]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Im Behälter zentrieren

Ankerart

```
myView.centerXAnchor.constraintEqualToAnchor(view.centerXAnchor).active = true
myView.centerYAnchor.constraintEqualToAnchor(view.centerYAnchor).active = true
```

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.CenterX, relatedBy:
```

```
NSLayoutConstraint.Equal, toItem: view, attribute: NSLayoutConstraint.CenterX, multiplier: 1,
constant: 0).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraint.CenterY, relatedBy:
NSLayoutConstraint.Equal, toItem: view, attribute: NSLayoutConstraint.CenterY, multiplier: 1,
constant: 0).active = true
```

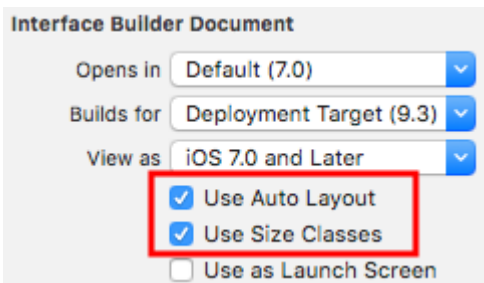
Visual Format Sprachstil

```
NSLayoutConstraint.constraintsWithVisualFormat("V:[viewKey]-(<=0)-[myViewKey]", options:
NSLayoutConstraint.FormatOptions.AlignAllCenterX, metrics: nil, views: ["myViewKey": myView, "viewKey":
view])
NSLayoutConstraint.constraintsWithVisualFormat("H:[viewKey]-(<=0)-[myViewKey]", options:
NSLayoutConstraint.FormatOptions.AlignAllCenterY, metrics: nil, views: ["myViewKey": myView, "viewKey":
view])
```

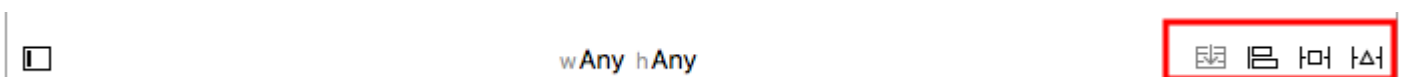
So verwenden Sie das automatische Layout

Auto-Layout wird verwendet, um Ansichten so anzuordnen, dass sie auf jedem Gerät und der Ausrichtung gut aussehen. Einschränkungen sind die Regeln, die festlegen, wie alles festgelegt werden soll. Dazu gehören unter anderem das Anheften von Kanten, das Zentrieren und das Einstellen von Größen.

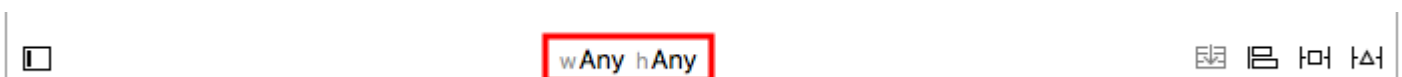
Das automatische Layout ist standardmäßig aktiviert. Sie können dies jedoch noch einmal überprüfen. Wenn Sie im *Projektnavigator* auf *Main.storyboard* klicken und dann den *Dateiinspektor anzeigen*. Stellen Sie sicher, dass Auto-Layout und Größenklassen aktiviert sind:



Automatische Layouteinschränkungen können im Interface Builder oder im Code festgelegt werden. Im Interface Builder finden Sie unten rechts die Werkzeuge für das automatische Layout. Wenn Sie darauf klicken, werden verschiedene Optionen zum Einstellen der Einschränkungen in einer Ansicht angezeigt.

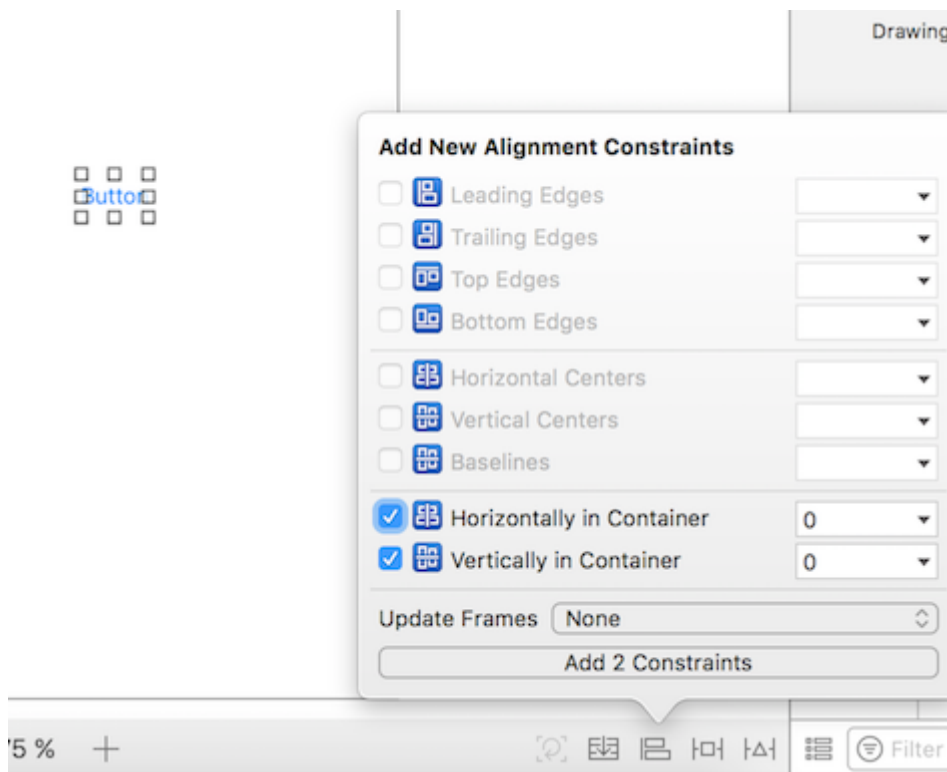


Wenn Sie unterschiedliche Einschränkungen für unterschiedliche Gerätegrößen oder -ausrichtungen wünschen, können Sie sie in den Optionen für die Größenklassen unten in der Mitte einstellen.

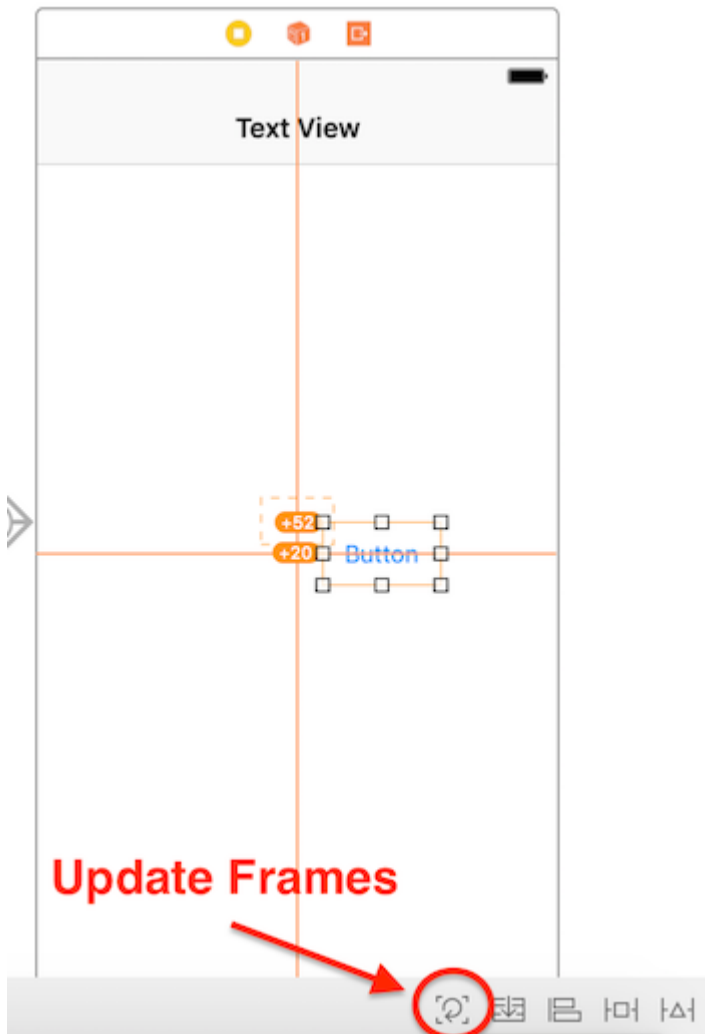


Zentrumsbeschränkungen

Wählen Sie im **Storyboard** Ihre Schaltfläche (oder die Ansicht, die Sie zentrieren möchten). Klicken Sie dann rechts unten auf die Schaltfläche zum Ausrichten. Wählen Sie `Horizontally in Container` und `Vertically in Container`. Klicken Sie auf "Add 2 Constraints".



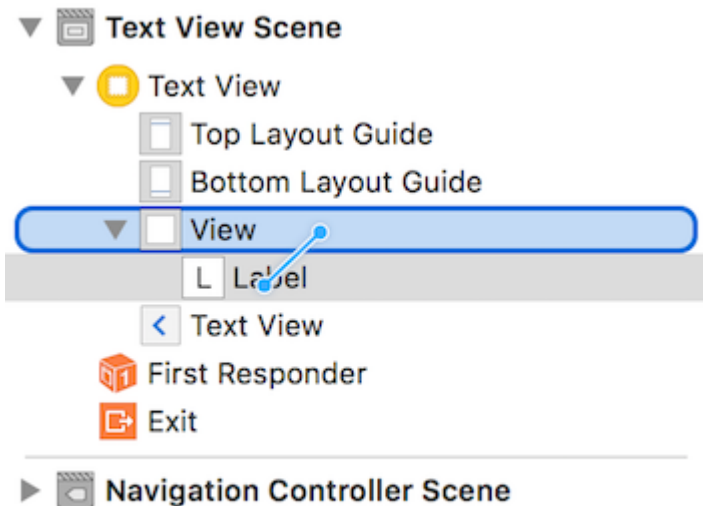
Wenn es noch nicht perfekt zentriert ist, müssen Sie möglicherweise noch etwas tun. Klicken Sie auf die Schaltfläche "Frames aktualisieren", die sich zwei neben der Schaltfläche "In Stapel einbetten" in der unteren Leiste befindet.



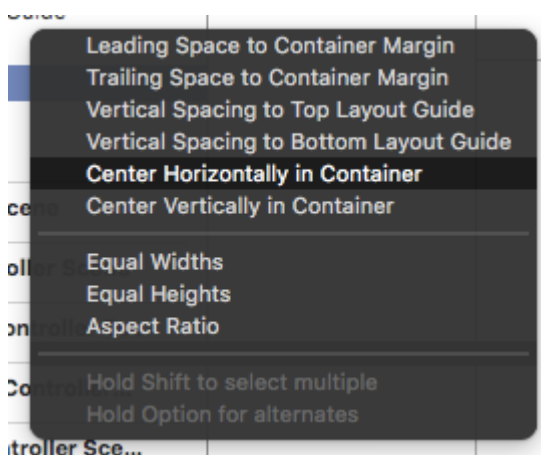
Sie können auch "Bilder nach Bedarf aktualisieren", indem Sie nach dem Auswählen der Ansicht zusammen $\square + \square + =$ (Befehl + Wahl und gleich) drücken. Dies kann etwas Zeit sparen.

Wenn Sie jetzt Ihre App ausführen, sollte sie zentriert sein, unabhängig von der verwendeten Gerätegröße.

Eine andere Möglichkeit, Ansichten mit dem Interface Builder zu zentrieren, ist das Ziehen und Ziehen mit der Maus. `UILabel` Sie möchten ein `UILabel` in einer Ansicht `UILabel`. Öffnen Sie die `Document Outline` in Ihrem Storyboard, indem Sie unten links auf die Seitenleistenschaltfläche klicken. Klicken Sie bei gedrückter `Strg`-Taste (Steuerung) von der Beschriftung in die Ansicht, und eine blaue Linie sollte angezeigt werden:



Bei der Freigabe erscheint ein Menü mit Einschränkungsoptionen:



Wählen Sie "Horizontal in Container zentrieren" und "Vertikal in Container zentrieren". Aktualisieren Sie die Frames nach Bedarf und voila! Ein zentriertes Etikett.

Alternativ können Sie die Einschränkungen programmgesteuert hinzufügen. Erstellen Sie die Abhängigkeiten und fügen Sie sie den gewünschten UI-Elementen und -Sichten hinzu, wie im folgenden Beispiel beschrieben. Dort erstellen wir eine Schaltfläche und richten sie in der Mitte horizontal und vertikal zum übergeordneten Bereich aus:

Ziel c

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    UIButton *yourButton = [[UIButton alloc] initWithFrame:CGRectMake(0, 0, 100, 18)];
    [yourButton setTitle:@"Button" forState:UIControlStateNormal];

    [self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterY multiplier:1 constant:0]]; //Align vertically center to
superView

```

```

[self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterX multiplier:1 constant:0]]; //Align horizontally center to
superView

[self.view addSubview:yourButton]; //Add button to superView
}

```

Schnell

```

override func viewDidLoad()
{
    super.viewDidLoad()
    let yourButton: UIButton = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 18))
    yourButton.setTitle("Button", forState: .Normal)

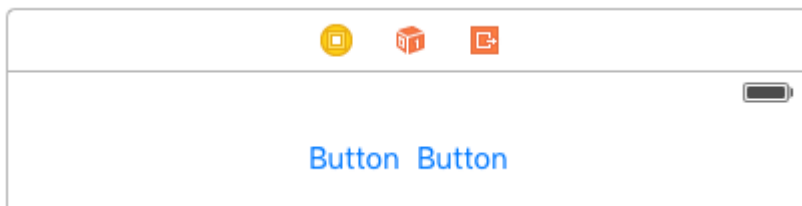
    let centerVertically = NSLayoutConstraint(item: yourButton,
                                             attribute: .CenterX,
                                             relatedBy: .Equal,
                                             toItem: view,
                                             attribute: .CenterX,
                                             multiplier: 1.0,
                                             constant: 0.0)

    let centerHorizontally = NSLayoutConstraint(item: yourButton,
                                               attribute: .CenterY,
                                               relatedBy: .Equal,
                                               toItem: view,
                                               attribute: .CenterY,
                                               multiplier: 1.0,
                                               constant: 0.0)

    NSLayoutConstraint.activateConstraints([centerVertically, centerHorizontally])
}

```

Raumansichten gleichmäßig



Es ist üblich, zwei Ansichten nebeneinander und zentriert in ihrem Überblick zu haben. Die übliche Antwort für Stack Overflow besteht darin, diese beiden Ansichten in eine `UIView` und die `UIView`. Dies ist nicht notwendig oder empfohlen. Aus den [UILayoutGuide](#)- Dokumenten:

Das Hinzufügen von Dummy-Ansichten zu Ihrer Ansichtshierarchie ist mit einer Reihe von Kosten verbunden. Erstens gibt es die Kosten für die Erstellung und Pflege der Ansicht selbst. Zweitens ist die Dummy-Ansicht ein vollständiges Mitglied der Ansichtshierarchie, was bedeutet, dass sie zu jeder Aufgabe, die die Hierarchie durchführt, zusätzlichen Aufwand verursacht. Am schlimmsten ist, dass die unsichtbare Dummy-Ansicht Nachrichten abfangen kann, die für andere Ansichten vorgesehen sind, was Probleme verursacht, die sehr schwer zu finden sind.

Sie können `UILayoutGuide` dazu verwenden, anstatt die Schaltflächen zu einer unnötigen `UIView`. Ein `UILayoutGuide` ist im Wesentlichen ein rechteckiger Bereich, der mit dem automatischen Layout interagieren kann. Sie platzieren einen `UILayoutGuide` auf der linken und rechten Seite der Schaltflächen und legen deren Breite gleich fest. Dadurch werden die Tasten zentriert. So machen Sie es im Code:

Visual Format Sprachstil

```
view.addSubview(button1)
view.addSubview(button2)

let leftSpace = UILayoutGuide()
view.addLayoutGuide(leftSpace)

let rightSpace = UILayoutGuide()
view.addLayoutGuide(rightSpace)

let views = [
    "leftSpace" : leftSpace,
    "button1" : button1,
    "button2" : button2,
    "rightSpace" : rightSpace
]

// Lay the buttons and layout guides out horizontally in a line.
// Put the layout guides on each end.
NSLayoutConstraint.activateConstraints(NSLayoutConstraint.constraintsWithVisualFormat("H:|[leftSpace][button1][button2][rightSpace]|", options: [], metrics: nil, views: views))

// Now set the layout guides widths equal, so that the space on the
// left and the right of the buttons will be equal
leftSpace.widthAnchor.constraintEqualToAnchor(rightSpace.widthAnchor).active = true
```

Ankerart

```
let leadingSpace = UILayoutGuide()
let trailingSpace = UILayoutGuide()
view.addLayoutGuide(leadingSpace)
view.addLayoutGuide(trailingSpace)

leadingSpace.widthAnchor.constraintEqualToAnchor(trailingSpace.widthAnchor).active = true

leadingSpace.leadingAnchor.constraintEqualToAnchor(view.leadingAnchor).active = true
leadingSpace.trailingAnchor.constraintEqualToAnchor(button1.leadingAnchor).active = true

trailingSpace.leadingAnchor.constraintEqualToAnchor(button2.trailingAnchor).active = true
trailingSpace.trailingAnchor.constraintEqualToAnchor(view.trailingAnchor).active = true
```

Sie müssen dazu auch vertikale Beschränkungen hinzufügen. Dadurch werden die Schaltflächen in der Ansicht zentriert, ohne "Dummy-Ansichten" hinzuzufügen! Dies erspart dem System, CPU-Zeit für das Anzeigen dieser "Dummy-Ansichten" zu verschwenden. In diesem Beispiel werden Schaltflächen verwendet. Sie können die Schaltflächen jedoch für jede Ansicht austauschen, für die Sie Einschränkungen festlegen möchten.

Wenn Sie iOS 8 oder eine frühere Version unterstützen, können Sie dieses Layout am einfachsten

erstellen, indem Sie versteckte Dummy-Ansichten hinzufügen. Mit iOS 9 können Sie Dummy-Ansichten durch Layout-Anleitungen ersetzen.

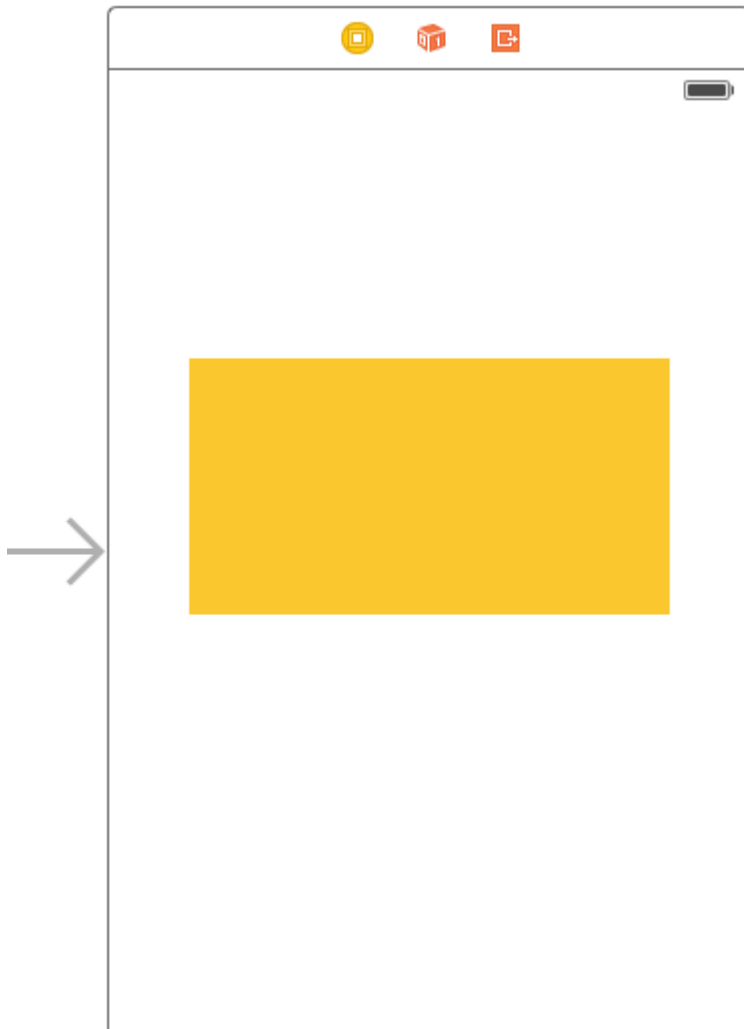
Hinweis: Interface Builder unterstützt noch keine Layout-Handbücher (Xcode 7.2.1). Wenn Sie sie verwenden möchten, müssen Sie Ihre Einschränkungen im Code erstellen. [Quelle](#) .

UILabel intrinsische Größe

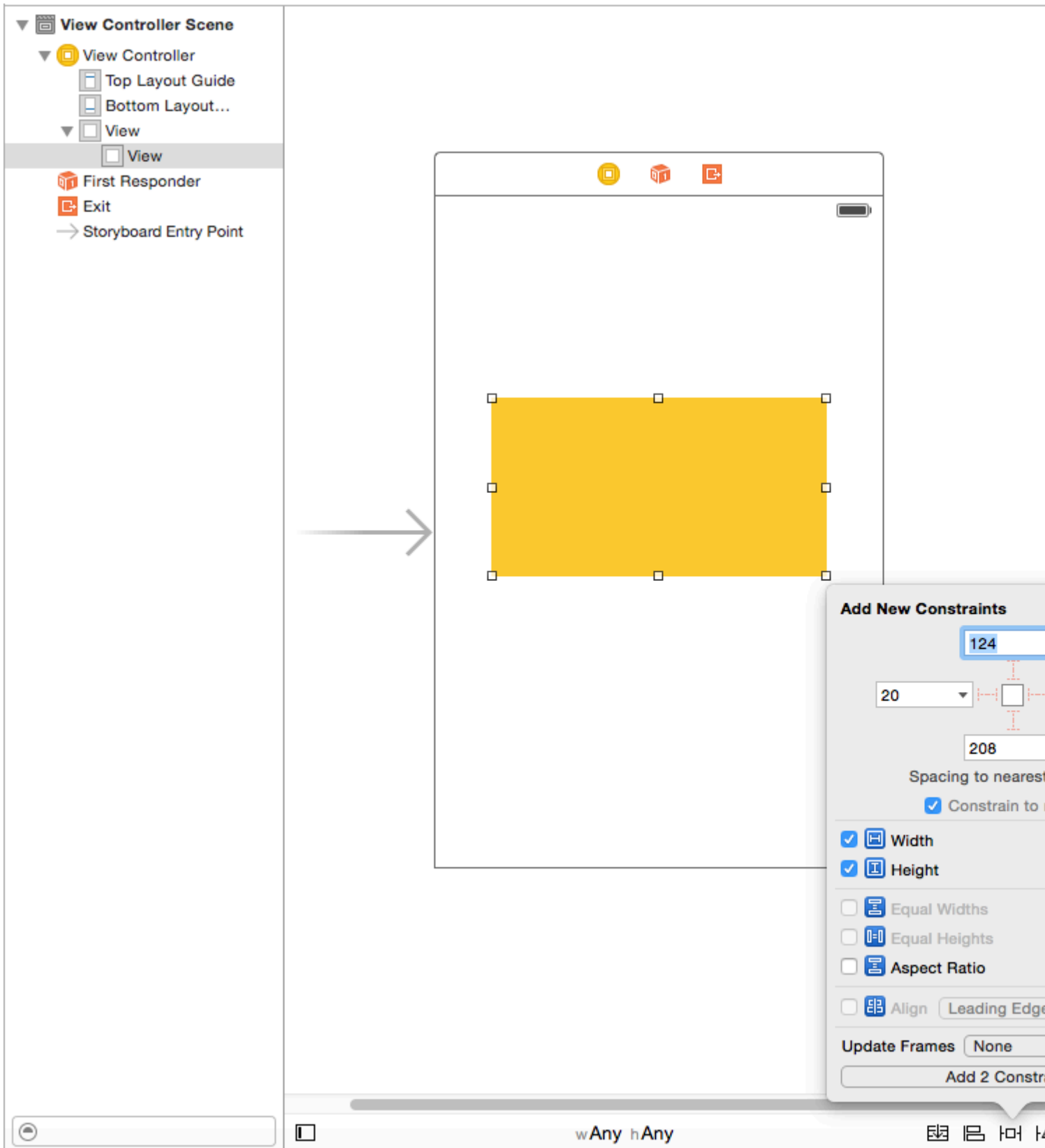
Wir müssen eine Ansicht erstellen, die einen Text mit einem Bildpräfix versehen wird. Text kann von variabler Länge sein. Wir müssen ein Ergebnis erzielen, bei dem sich Text + Bild immer in der Mitte einer übergeordneten Ansicht befindet.



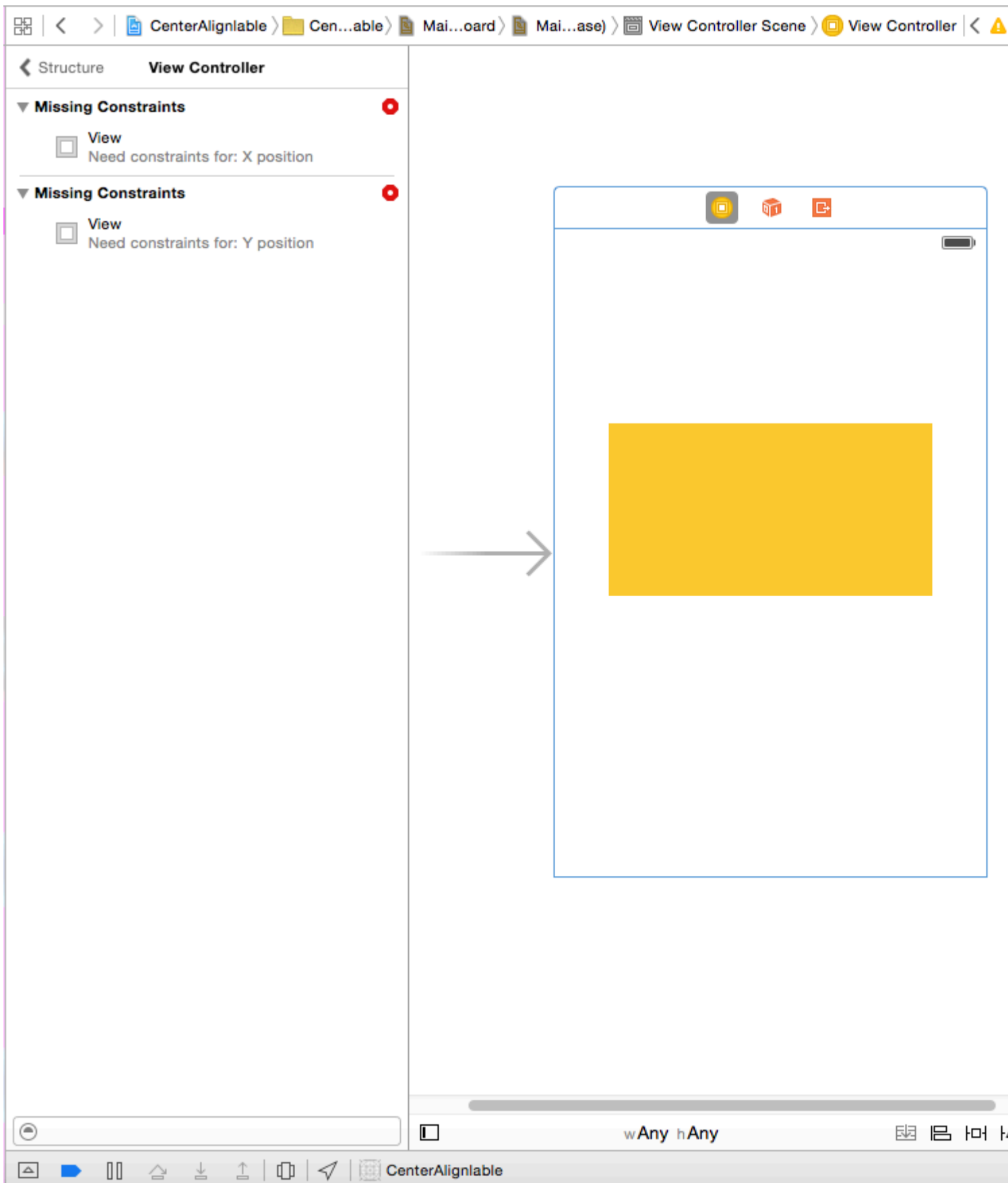
Schritt 1: Erstellen Sie zunächst ein einzelnes Ansichtsprojekt und benennen Sie es nach Belieben. Öffnen Sie die Faustansicht des Storyboards. Beschreiben Sie eine Ansicht mit angemessener Größe und setzen Sie die Hintergrundfarbe auf gelb. Ich habe meinen Viewcontroller auf 3,5 " verkleinert view sollte so aussehen



Schritt 2: Jetzt fügen wir der gelben Ansicht Einschränkungen hinzu. Zuerst werden Breiten- und Höhenbeschränkungen hinzugefügt. (Warte eine Minute, haben wir nicht gesagt, dass die Ansicht dynamische Breite haben wird? Ok, wir werden später darauf zurückkommen.) Die folgenden Einschränkungen, wie in der Abbildung unten dargestellt, stören sich nicht mit dem width-Wert. Jeder Wert wird für die Breite gut sein. Lassen Sie ihn einfach groß genug, damit wir Autolayouts richtig hinzufügen können.



Nachdem Sie diese beiden Einschränkungen hinzugefügt haben, werden Sie feststellen, dass XCode Fehler ausgibt, da in der folgenden Abbildung diese angezeigt und verstanden werden können.

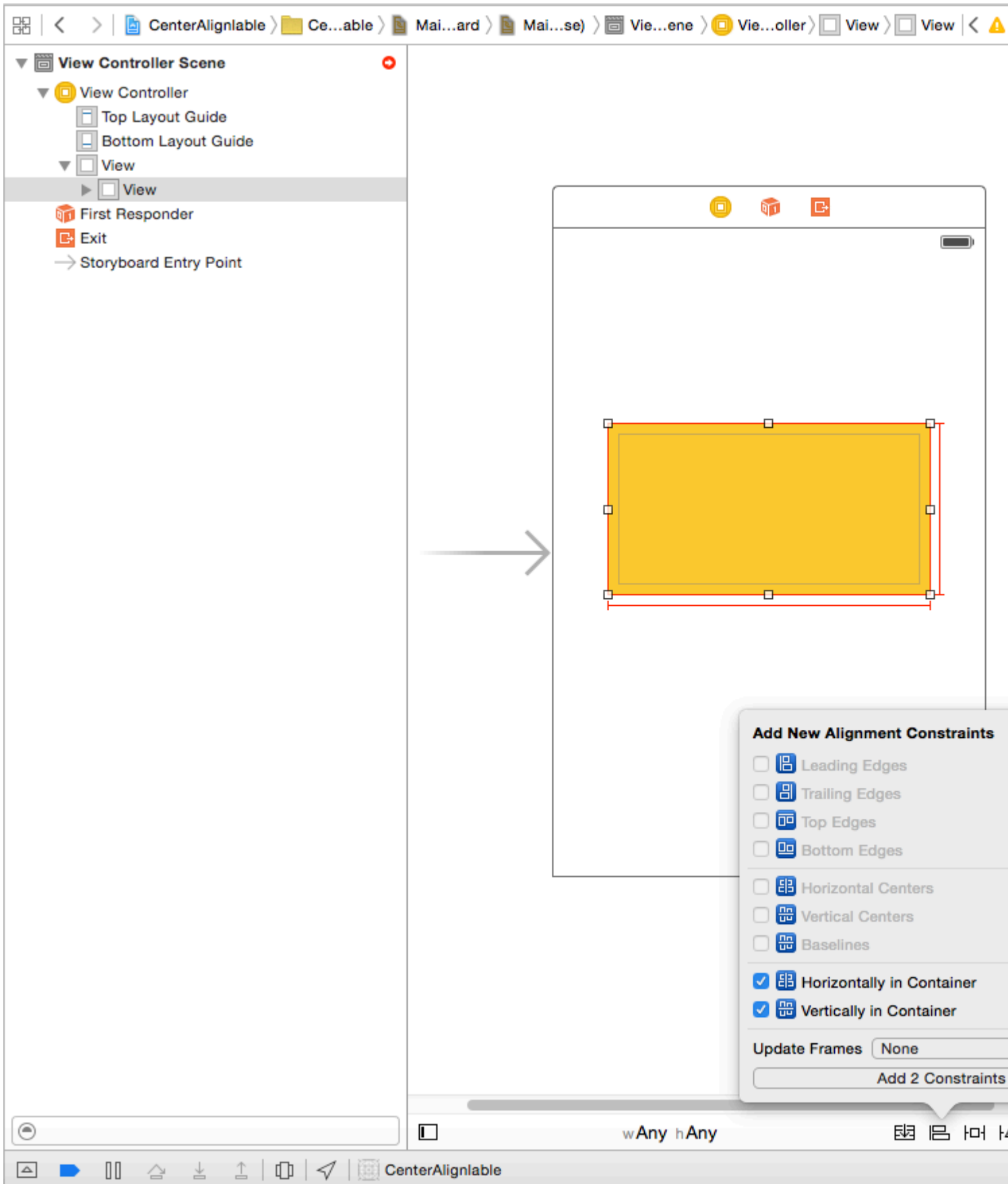


Wir haben zwei Fehler (Rot bedeutet Fehler). Wie oben erläutert, können Sie den Mehrdeutigkeits-Teil erneut aufrufen

Fehlende Einschränkungen: Benötigen Sie Einschränkungen für: X-Position: - Wie oben erläutert, haben wir der Ansicht eine Breite und eine Höhe gegeben, damit die „BOUNDS“ definiert

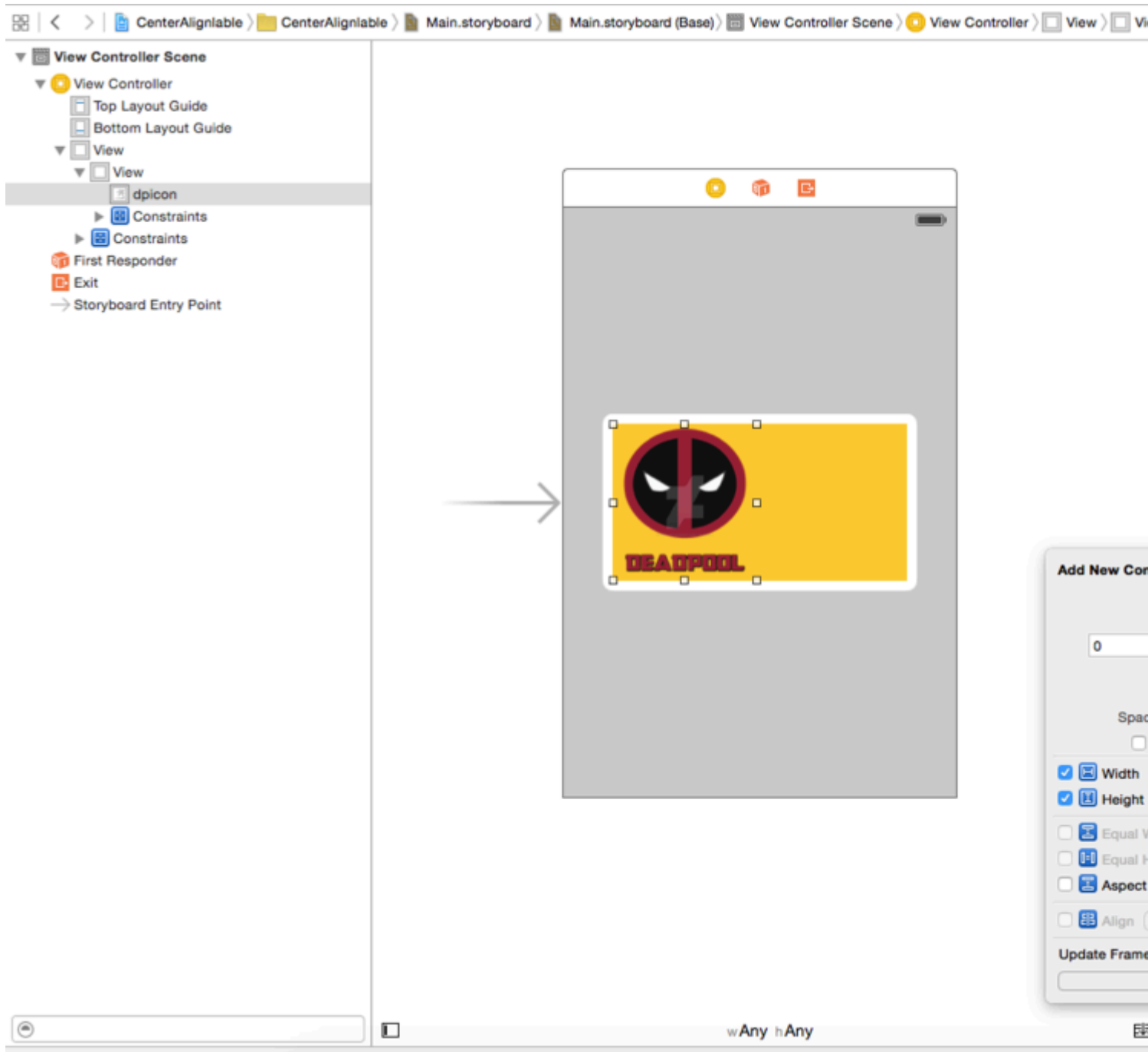
werden, aber wir haben nicht den Ursprung angegeben, sodass der „FRAME“ nicht definiert ist. Autolayout kann nicht die X-Position unserer gelben Ansicht bestimmen

Fehlende Einschränkungen: Benötigen Sie Einschränkungen für: Y-Position: - Wie oben erläutert, haben wir der Ansicht eine Breite und eine Höhe gegeben, so dass die „BOUNDS“ definiert sind, der Ursprung jedoch nicht angegeben wurde, sodass der „FRAME“ nicht definiert ist. Autolayout kann nicht bestimmen, was die Y-Position unserer gelben Ansicht sein wird. Um dieses Problem zu lösen, müssen Sie Autolayout etwas geben, um X und Y zu resolvieren. Da wir keine Frames setzen können, werden wir dies automatisch tun. Fügen Sie die folgenden Einschränkungen hinzu Bild unten werde ich es später erklären



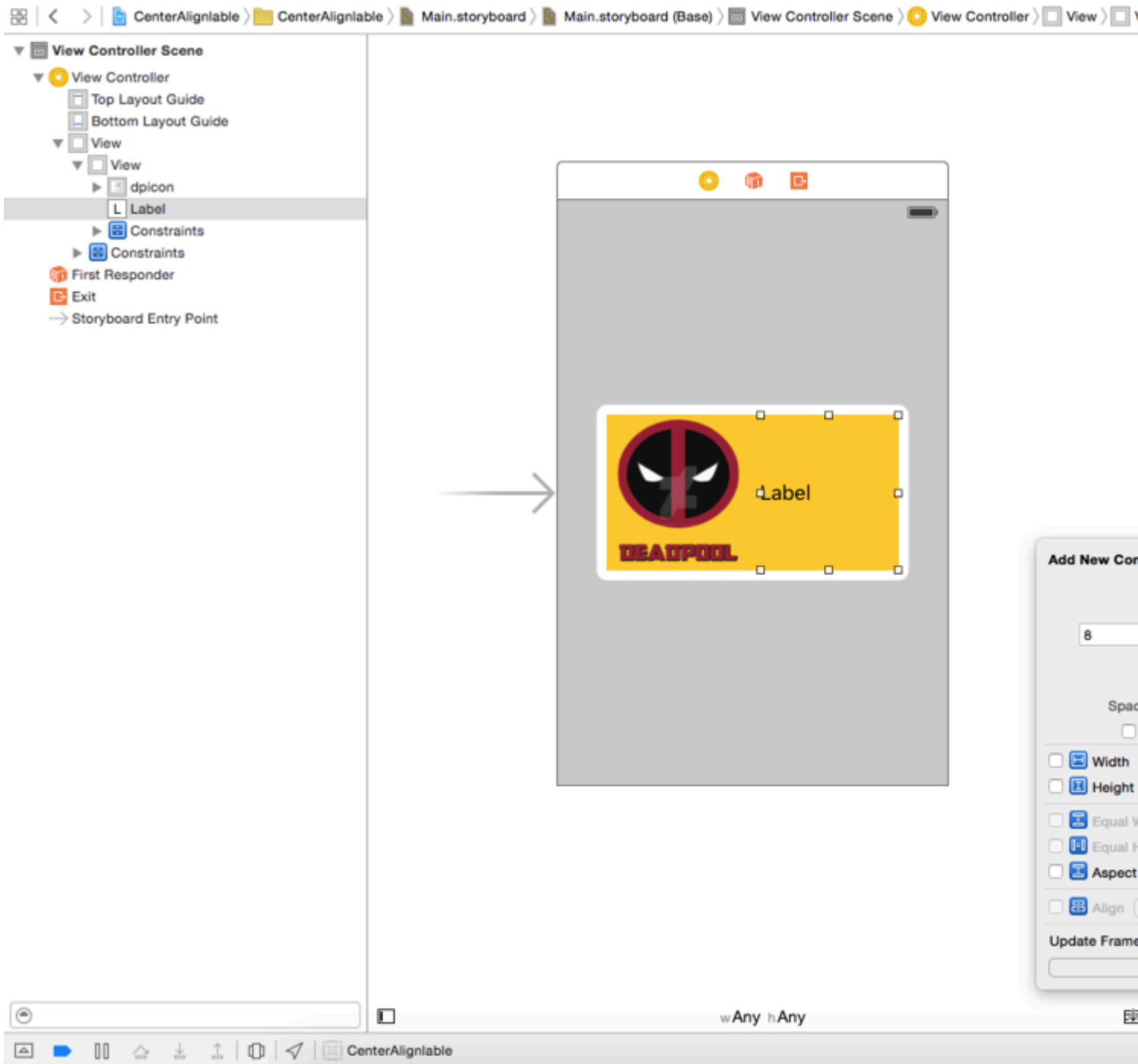
Was wir getan haben, ist, dass wir ein "Vertikales Zentrum" und ein "Horizontales Zentrum" hinzugefügt haben. Diese Einschränkung sagt dem Autolayout, dass sich unsere gelbe Ansicht immer in der Mitte befindet. Horizontal ist also X in bestimmtem Gleichem mit vertikaler Beschränkung und Y wird bestimmt. Möglicherweise muss der Rahmen angepasst werden).

Schritt 3: Inzwischen ist unsere gelbe Basisansicht fertig. Wir fügen das Präfixbild als Unteransicht unserer gelben Ansicht mit den folgenden Einschränkungen hinzu. Sie können ein beliebiges Bild Ihrer Wahl auswählen.



Da wir ein festes Maß für unser Präfixbild haben, haben wir eine feste Breite für diese Bildansicht. Fügen Sie die Einschränkungen hinzu und fahren Sie mit dem nächsten Schritt fort.

Schritt 4: Fügen Sie ein UILabel als Untersicht unserer gelben Ansicht hinzu und fügen Sie die folgenden Einschränkungen hinzu

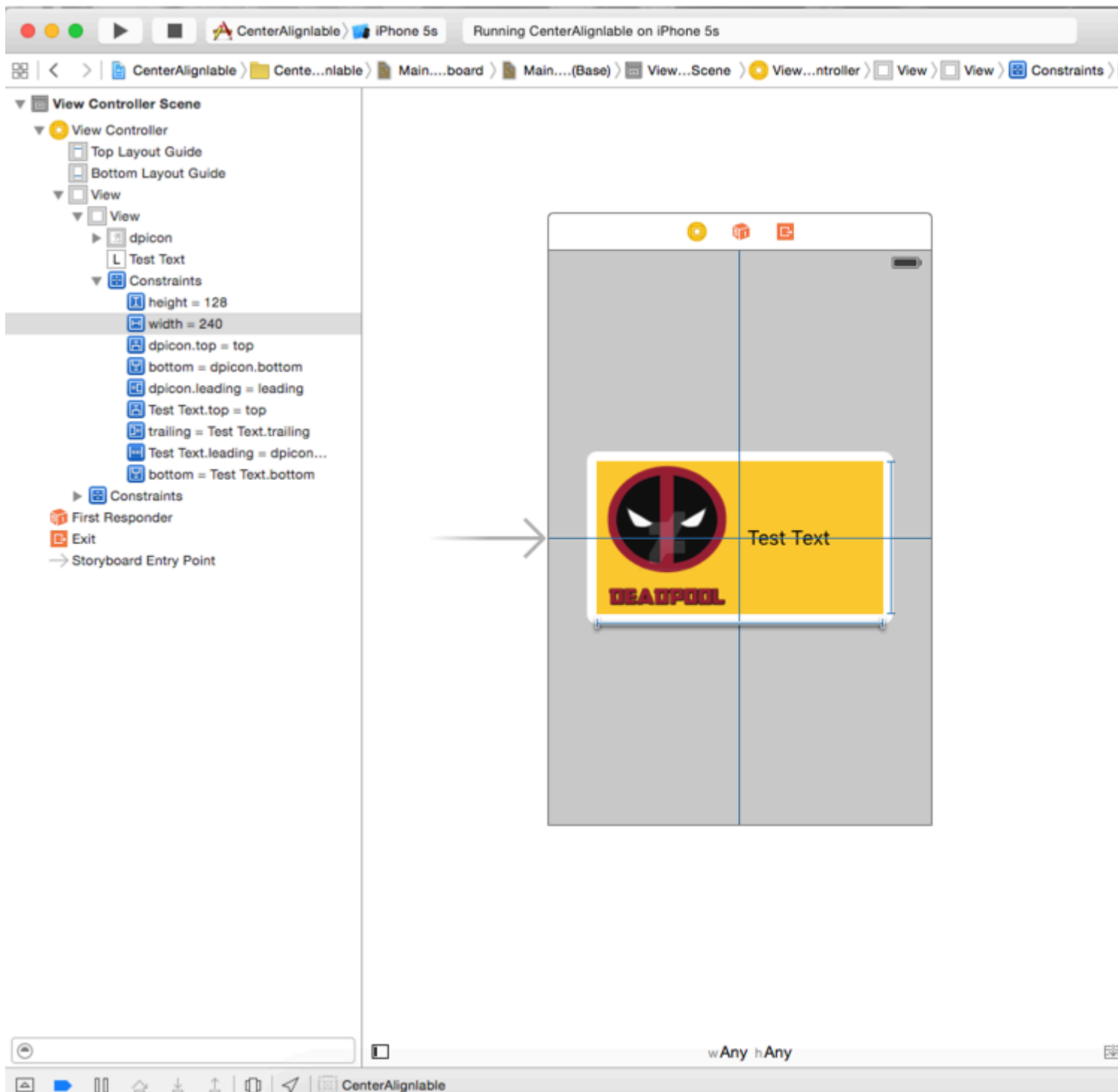


Wie Sie sehen, habe ich unserem UILabel nur relative Beschränkungen auferlegt. Seine 8 Punkte aus dem Präfix image und 0,0,0 oben und unten aus der gelben Ansicht. Da die Breite dynamisch sein soll, geben wir keine Breiten- oder Höhenbeschränkungen an .

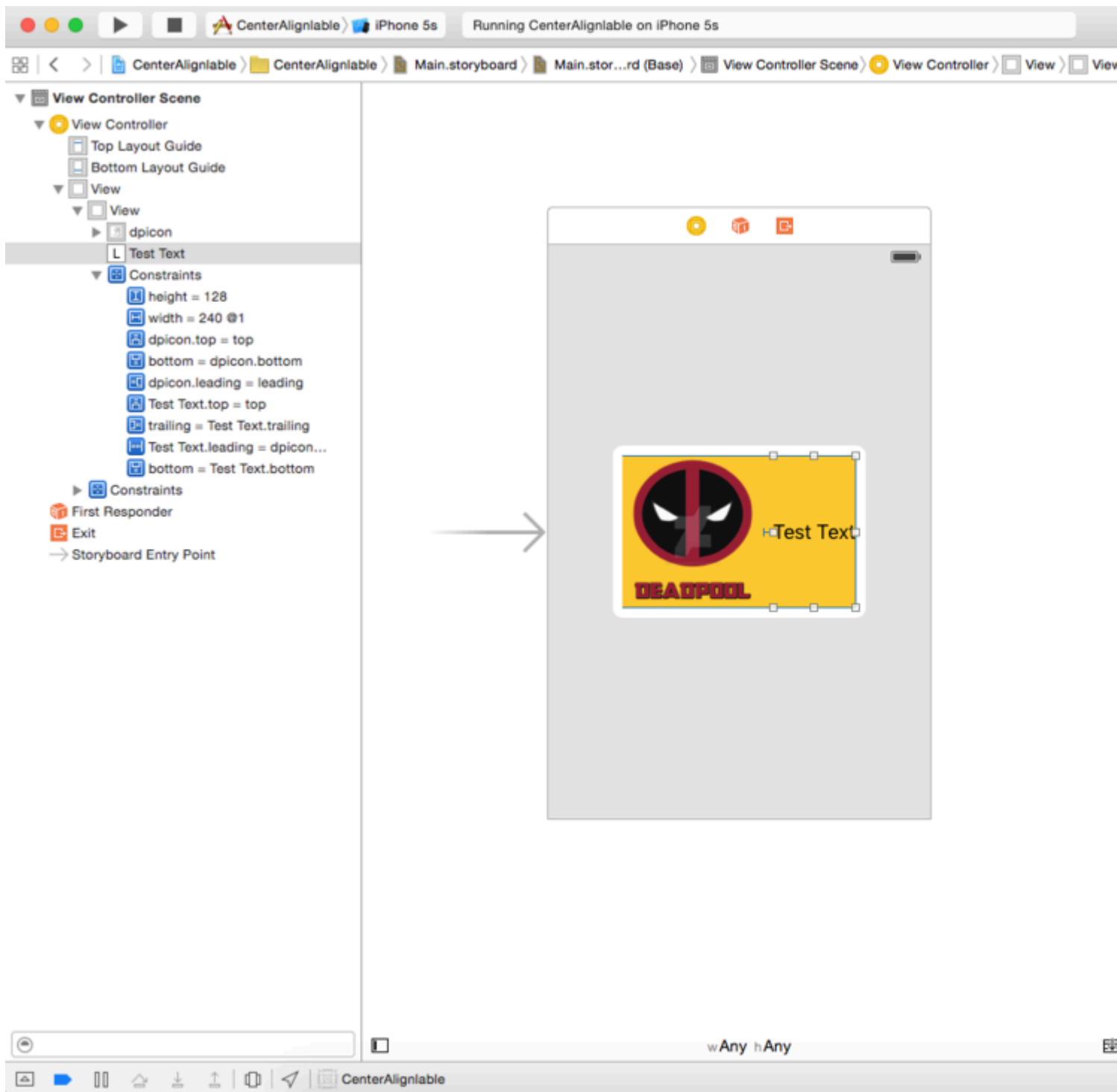
F: Warum bekommen wir jetzt keine Fehler, wir haben keine Breite und Höhe angegeben?
 Antwort: - Fehler oder Warnungen werden nur dann angezeigt, wenn das automatische Layout keine Elemente auflösen kann, die erforderlich sind, um eine Ansicht auf dem Bildschirm darzustellen. Die Breite oder der Ursprung der Komponente ist jedoch. Da unser Etikett relativ zu dem gelben Bild und dem Präfixbild und deren Rahmen ist ist klar definiert, dass autolayout den Rahmen unseres Labels berechnen kann.

Schritt 5: Wenn wir uns erinnern, werden wir feststellen, dass wir eine feste Ansicht für die gelbe Ansicht gegeben haben, aber wir möchten, dass sie abhängig vom Text unseres Etiketts

dynamisch ist. So ändern wir unsere Breitenbeschränkung der gelben Ansicht. Breite der gelben Ansicht ist notwendig, um Mehrdeutigkeiten aufzulösen, wir möchten jedoch, dass sie zur Laufzeit basierend auf dem Inhalt von UILabel außer Kraft gesetzt werden. Wir wählen also unsere gelbe Ansicht aus, gehen zum Größeninspektor und reduzieren die Priorität der Breitenbeschränkung auf 1, so dass sie übersteuert wird. Folgen Sie der Abbildung unten.

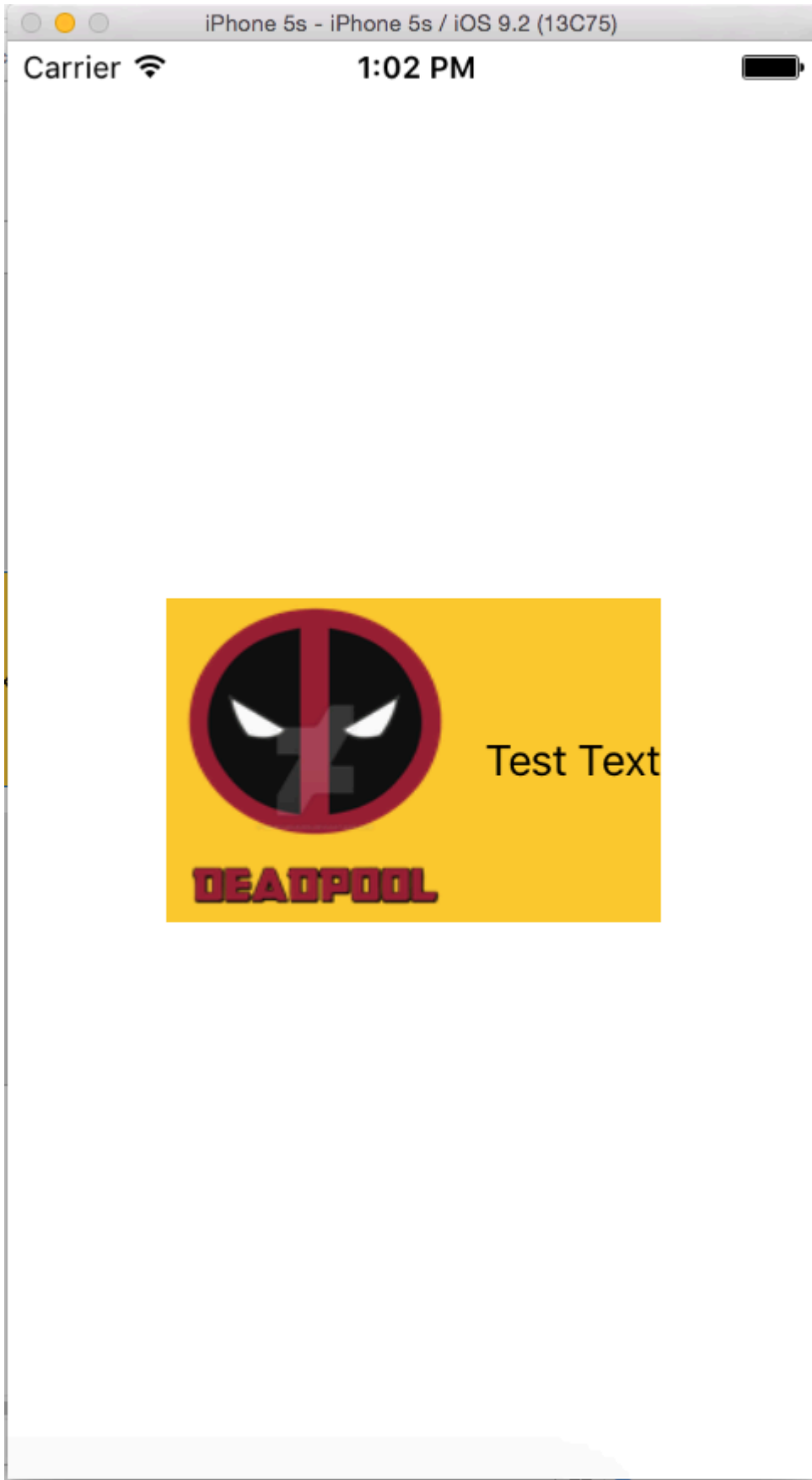


Schritt 6: Wir möchten, dass UILabel nach Text expandiert und unsere gelbe Ansicht erweitert wird. Daher haben wir die Priorität der gelben Ansichtsbreite reduziert. Jetzt werden wir die Priorität der Textkomprimierungsbeständigkeit unseres UILabels erhöhen. Wir möchten, dass unsere Ansicht darunter fällt Nun, wir werden die Priorität der Inhaltsanpassung von UILabel erhöhen. Folgen Sie dem Bild unten



Wie Sie sehen, haben wir die Priorität der Inhalte auf 500 und die Priorität der Komprimierungsresistenz auf 751 erhöht. Damit wird die Priorität der Breitenbeschränkung mit 1 Priorität überwunden.

Jetzt erstellen und ausführen, werden Sie Folgendes sehen.



Wie man mit Auto Layout animiert

Ohne Auto-Layout wird die Animation im Laufe der Zeit geändert. Bei automatischem Layout bestimmen die Einschränkungen den Ansichtsrahmen. Daher müssen Sie die Einschränkungen

stattdessen animieren. Diese Umkehrung macht es schwieriger, Animationen zu visualisieren.

Hier sind die Möglichkeiten, mit Auto-Layout zu animieren:

1. **Ändern Sie die Konstante der Einschränkung** nach der Erstellung mithilfe von periodischen Aufrufen (`CADisplayLink` , `dispatch_source_t` , `dispatch_after` , `NSTimer`). Rufen `layoutIfNeeded` dann `layoutIfNeeded` auf, um die Einschränkung zu aktualisieren. Beispiel:

Ziel c:

```
self.someConstraint.constant = 10.0;
[UIView animateWithDuration:0.25 animations:^(
    [self.view layoutIfNeeded];
)];
```

Schnell:

```
self.someConstraint.constant = 10.0
UIView.animate(withDuration: 0.25, animations: self.view.layoutIfNeeded)
```

2. **Ändern Sie die Einschränkungen** und rufen Sie `[view layoutIfNeeded]` in einem Animationsblock auf. Dies interpoliert zwischen den beiden Positionen und ignoriert Einschränkungen während der Animation.

```
[UIView animateWithDuration:0.5 animations:^(
    [view layoutIfNeeded];
)]
```

3. **Ändern Sie die Priorität der Einschränkungen** . Dies ist weniger CPU-intensiv als das Hinzufügen und Entfernen von Einschränkungen.
4. **Entfernen Sie alle Einschränkungen, und verwenden Sie Masken zum automatischen Ändern** . Für später muss `view.translatesAutoresizingMaskIntoConstraints = YES` .
5. **Verwenden Sie Einschränkungen, die die beabsichtigte Animation nicht beeinträchtigen** .
6. **Verwenden Sie eine Containeransicht** . Positionieren Sie den Überblick mit Einschränkungen. Fügen Sie dann eine Unteransicht mit Einschränkungen hinzu, die der Animation nicht widersprechen, z. B. ein Zentrum relativ zum Superview. Dies entlädt einen Teil der Einschränkungen in das Superview, sodass die Animation in der Unteransicht nicht bekämpft wird.
7. **Animieren Sie stattdessen Ansichten** . Layer-Transformationen lösen das automatische Layout nicht aus.

```
CABasicAnimation* ba = [CABasicAnimation animationWithKeyPath:@"transform"];
ba.autoreverses = YES;
ba.duration = 0.3;
ba.toValue = [NSValue valueWithCATransform3D:CATransform3DMakeScale(1.1, 1.1, 1)];
```

```
[v.layer addAnimation:ba forKey:nil];
```

8. Überschreiben Sie `layoutSubviews` . Rufen Sie `[super layoutSubviews]` und `[super layoutSubviews]` die Einschränkungen an.

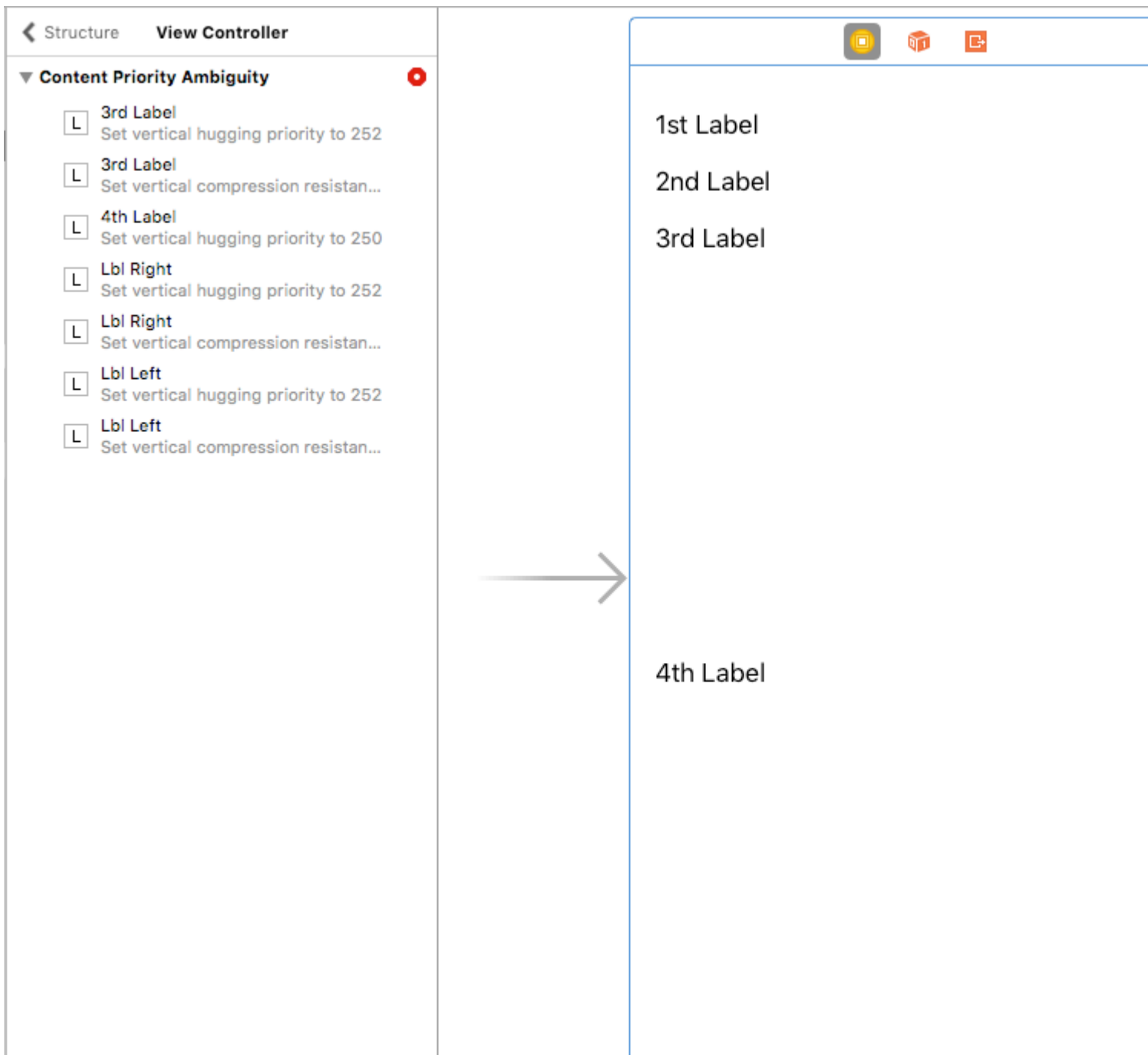
9. Ändern Sie den Frame in `viewDidLayoutSubviews` . Das automatische Layout wird in `layoutSubviews viewDidLayoutSubviews` . Ändern Sie es in `viewDidLayoutSubviews` .

10. Deaktivieren Sie das automatische Layout und setzen Sie die Ansichten manuell. Sie können dies tun, indem Sie `layoutSubviews / layout` überschreiben, ohne die Implementierung der `layoutSubviews` aufzurufen.

Schneller Tipp: Wenn das übergeordnete `layoutIfNeeded()` der animierten Ansicht nicht interpoliert wird (`layoutIfNeeded()` die Animation springt vom Anfangs- in den `layoutIfNeeded()`), rufen Sie `layoutIfNeeded()` in der tiefsten Ansicht des übergeordneten `layoutIfNeeded()` der animierten Ansicht (mit anderen Worten) auf , das ist von der Animation nicht betroffen). Ich weiß nicht genau, warum das funktioniert.

Lösen Sie den UILabel-Prioritätskonflikt

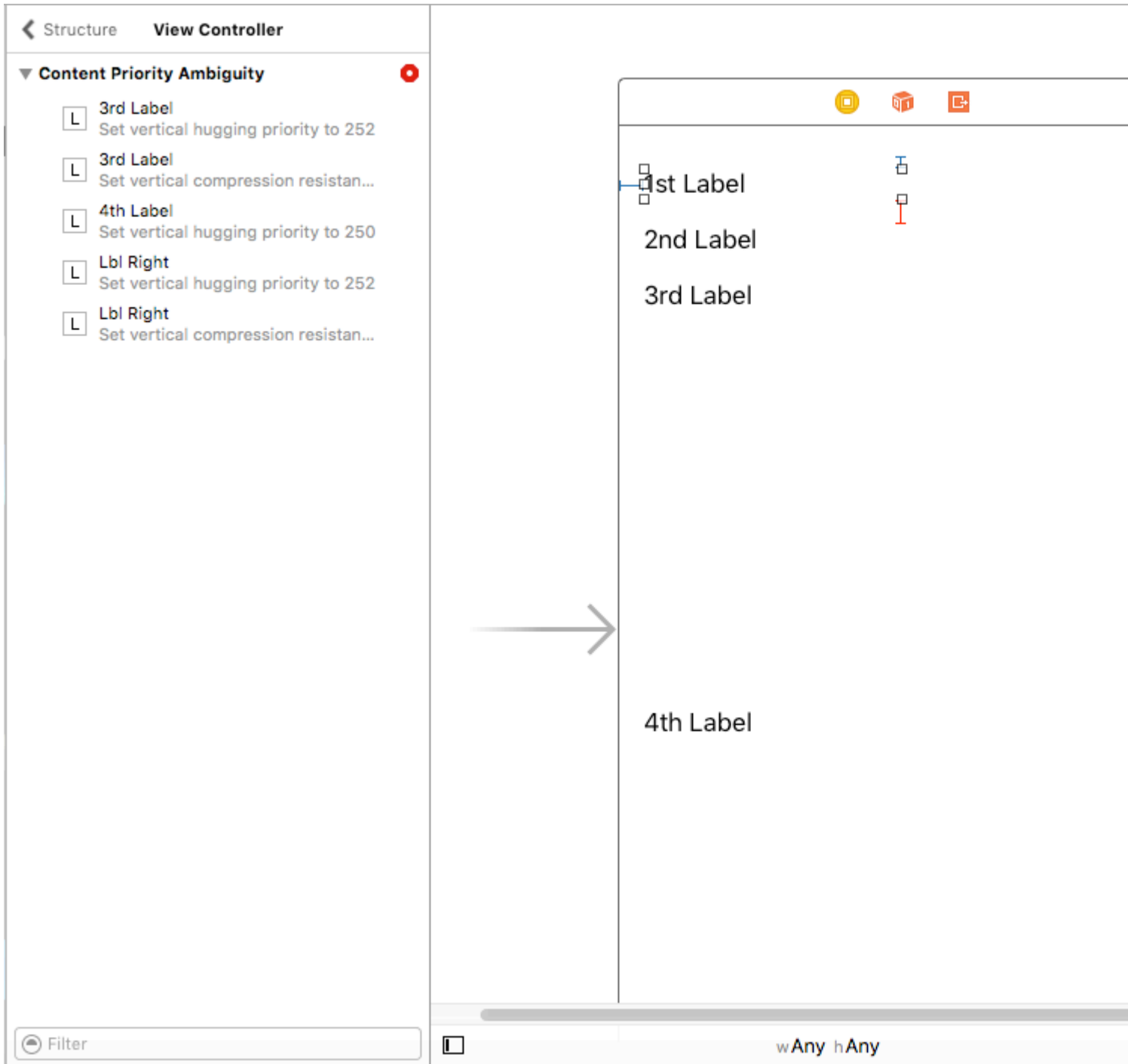
Problem : Wenn Sie viele Beschriftungen in einer Ansicht verwenden, wird möglicherweise eine **Warnung angezeigt :**

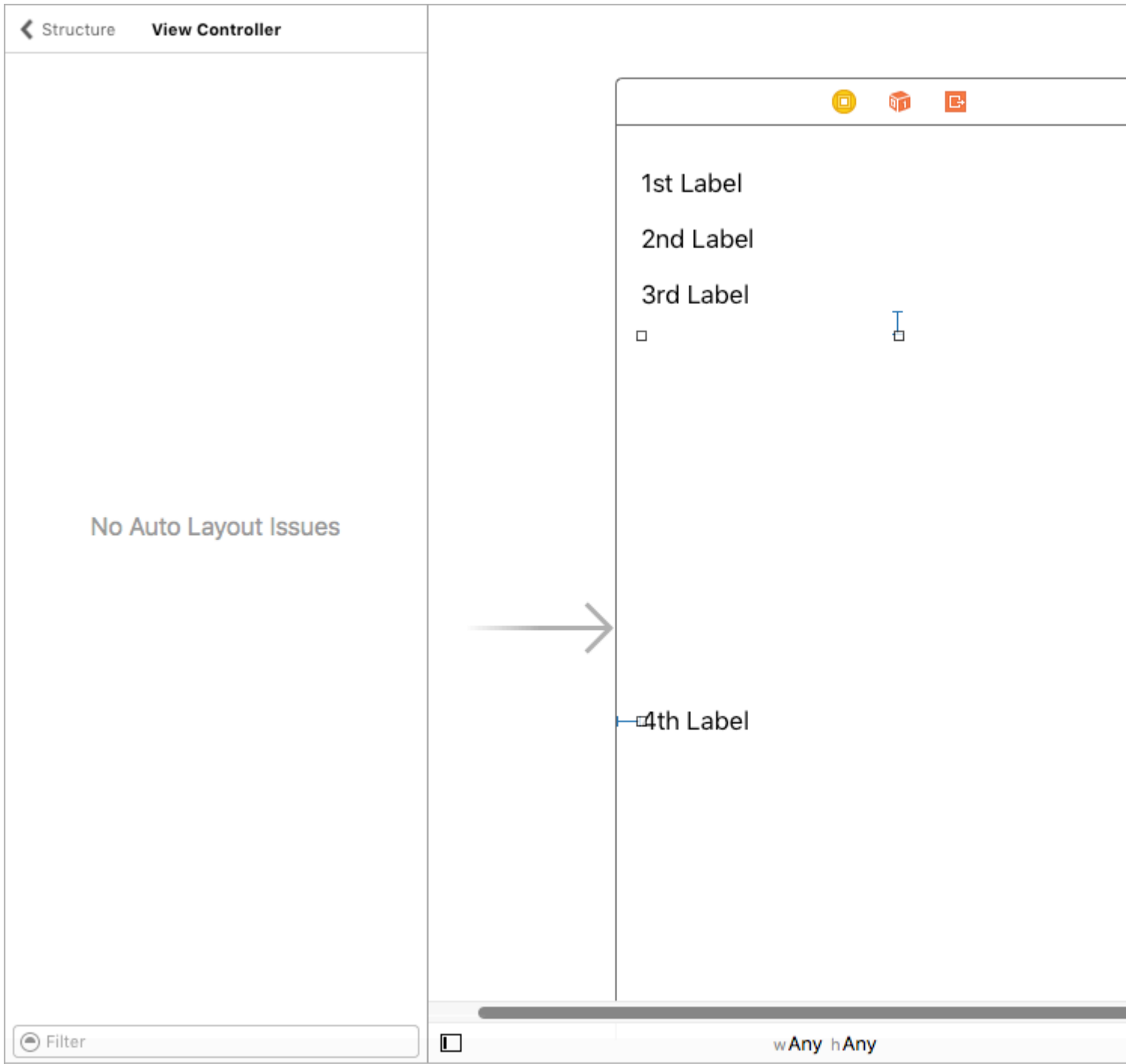


Wie können wir diese **Warnung** beheben?

Lösung : Wir berechnen und setzen die Prioritäten in Reihenfolge. Die Prioritäten müssen sich von den Etiketten unterscheiden. Dies bedeutet, dass das, was wichtig ist, eine höhere Priorität erhält. In meinem Fall stelle ich beispielsweise die vertikalen Prioritäten für meine Etiketten so ein:

Ich habe die höchste Priorität für das 1. Label und die niedrigste für das 4. Label festgelegt.





In einem ViewController glaube ich, dass Sie die Auswirkungen dieser Prioritäten nur schwer erkennen können. Mit UITableViewCell + schätzen Sie die Zellenhöhe jedoch sehr deutlich.

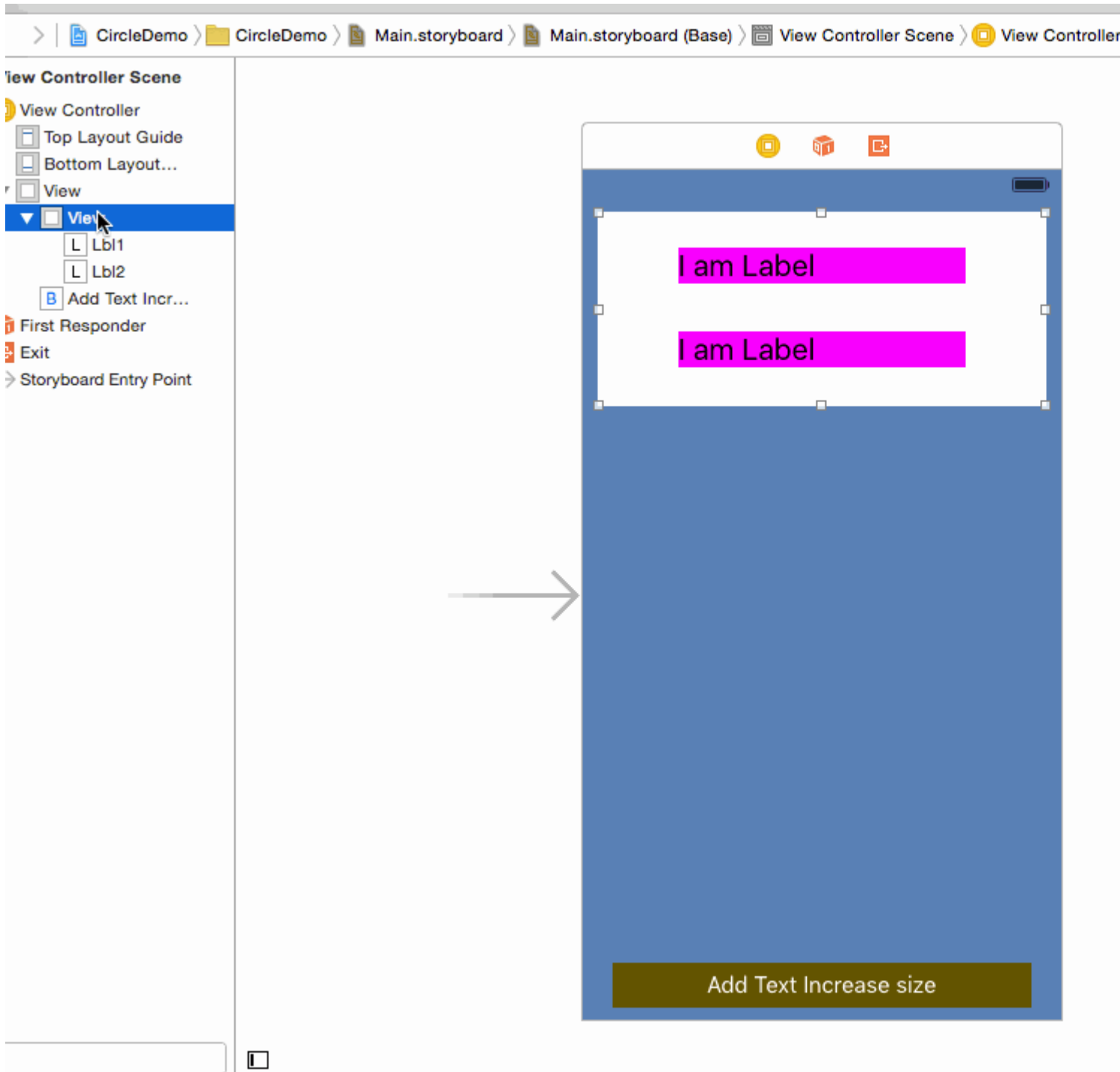
Ich hoffe das hilft.

UILabel & Parentview-Größe Gemäß Text in UILabel

Schritt für Schritt Anleitung :-

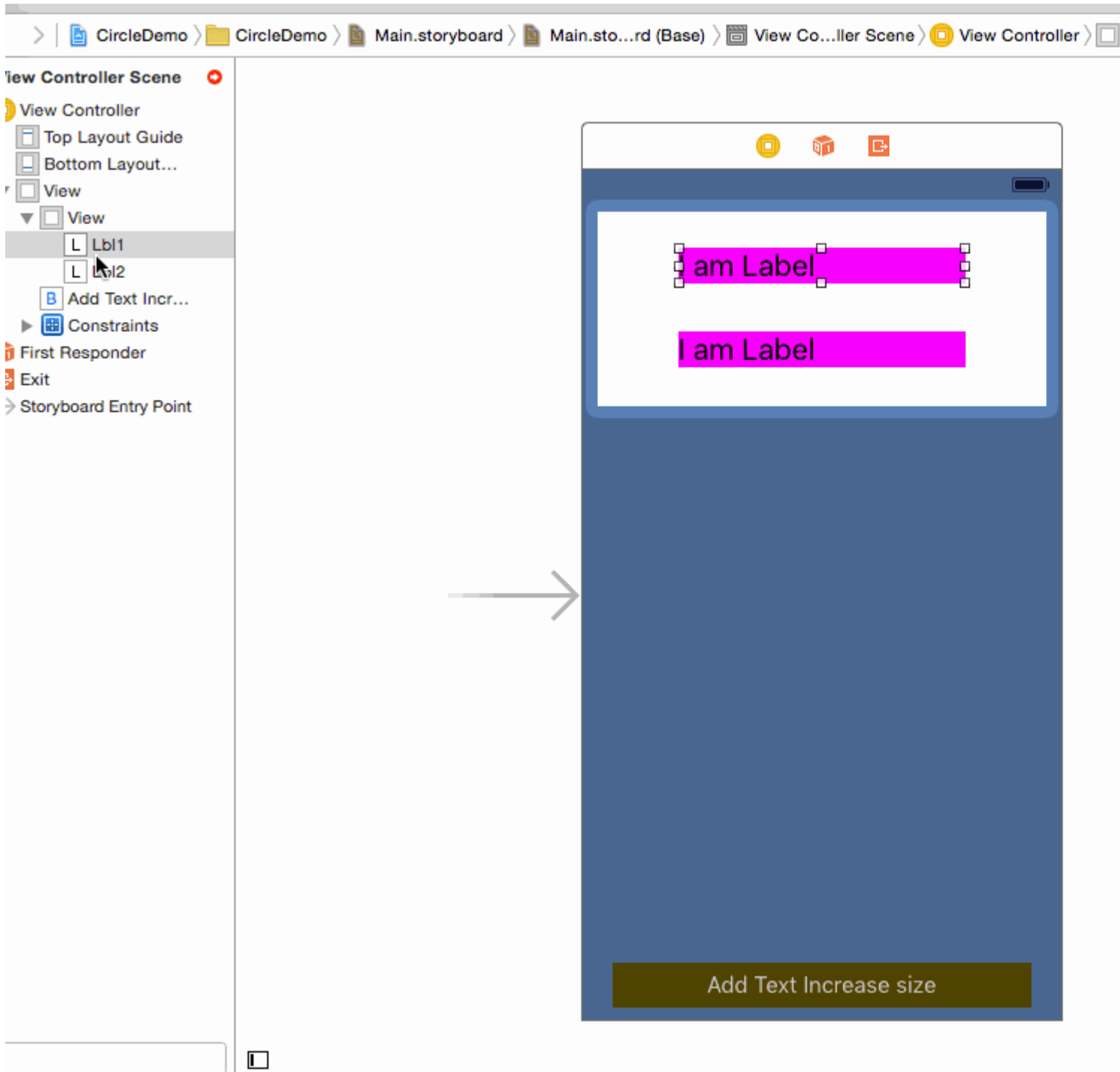
Schritt 1: - Setzen Sie die Einschränkung auf UIView

1. Führend 2) oben. 3) Nachlaufen. (Von der Hauptansicht)



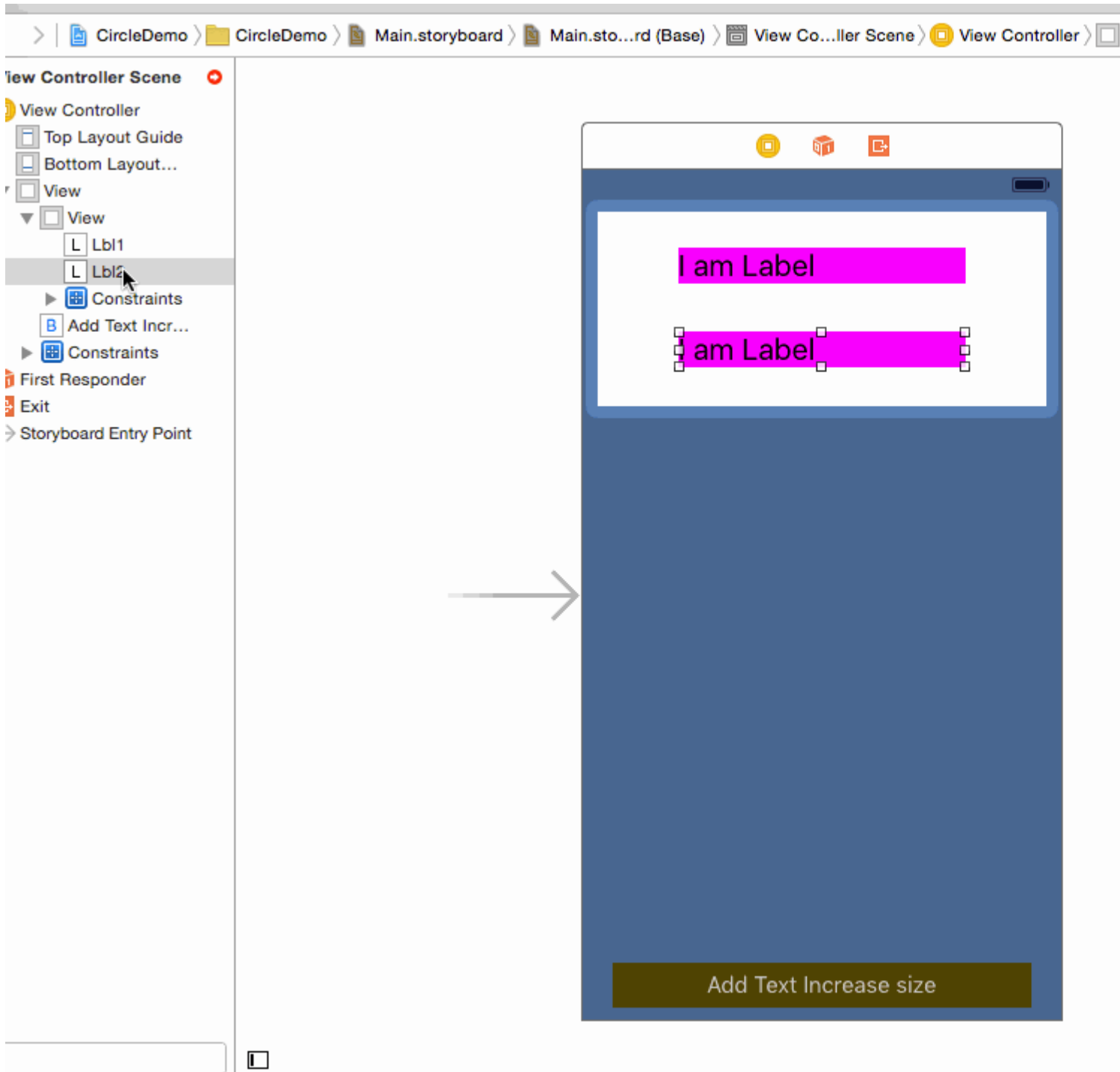
Schritt 2: - Setzen Sie die Einschränkung auf Label 1

1. Leading 2) Top 3) Trailing (Aus dem Superview)

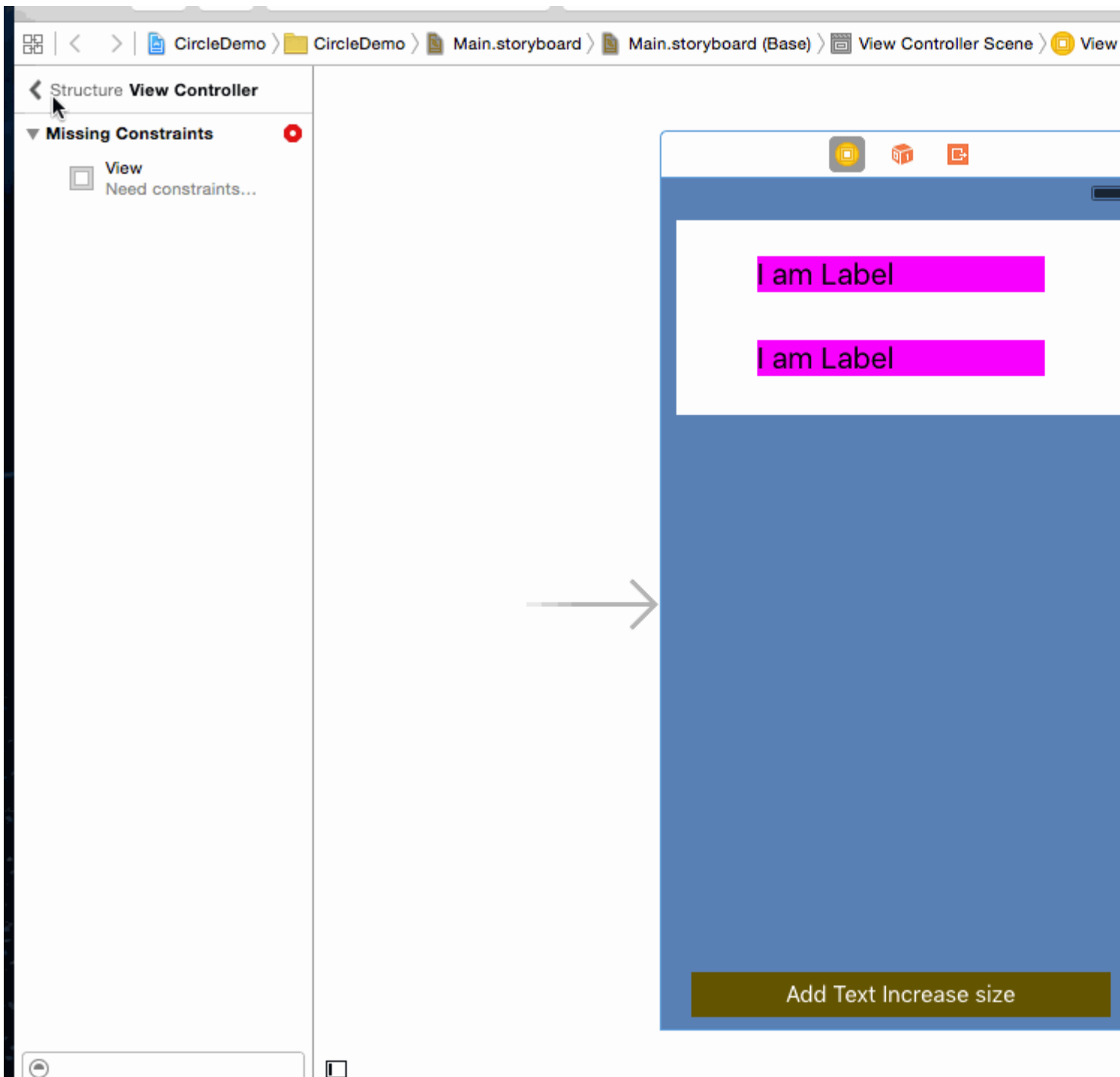


Schritt 3: - Setzen Sie die Einschränkung auf Label 2

1. Leading 2) Top 3) Trailing (Aus dem Überblick)

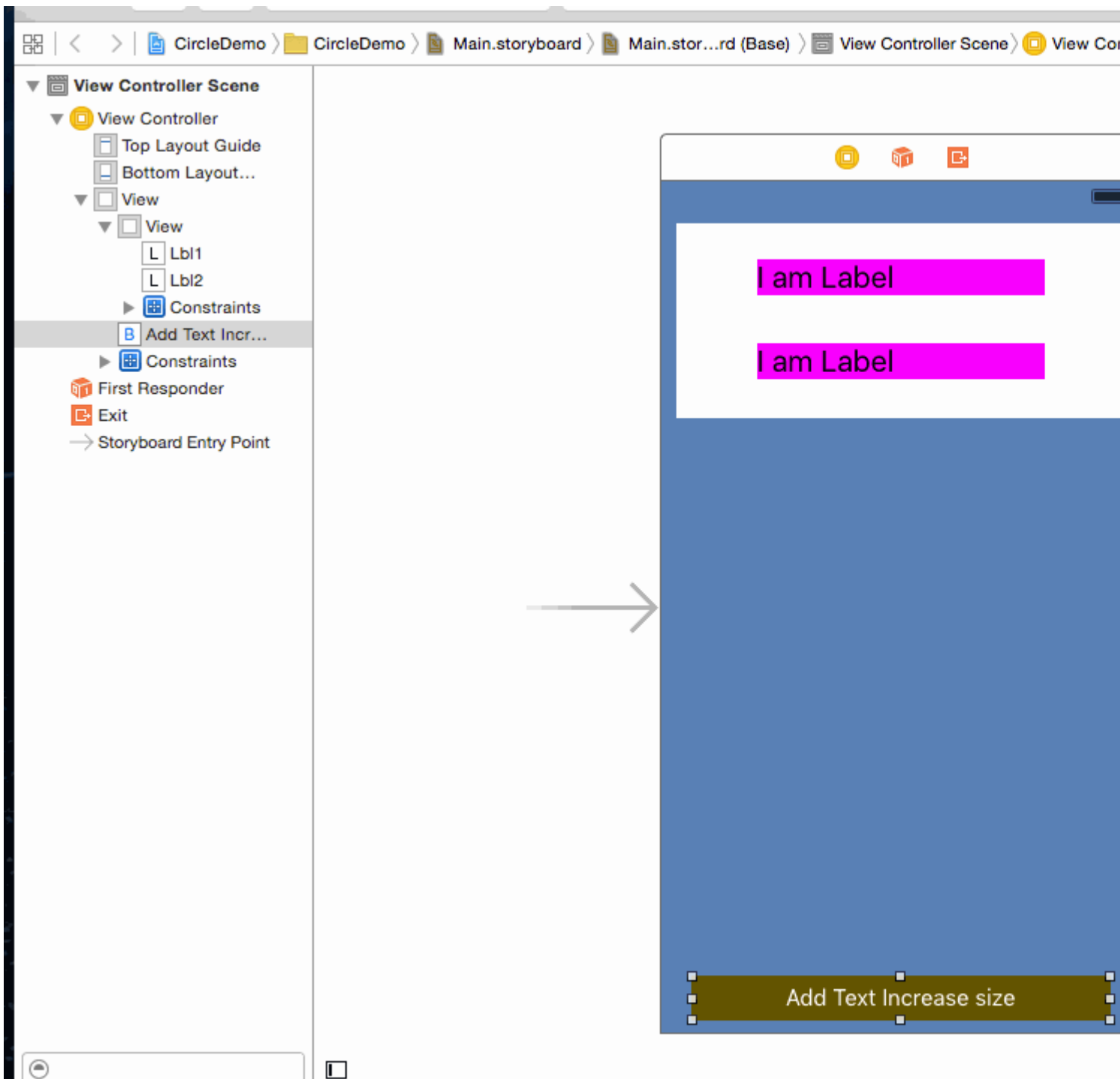


Schritt 4: - Am schwierigsten geben Sie UILabel von UIView einen Boden.



Schritt 5: - (Optional) Legen Sie die Einschränkung auf UIButton fest

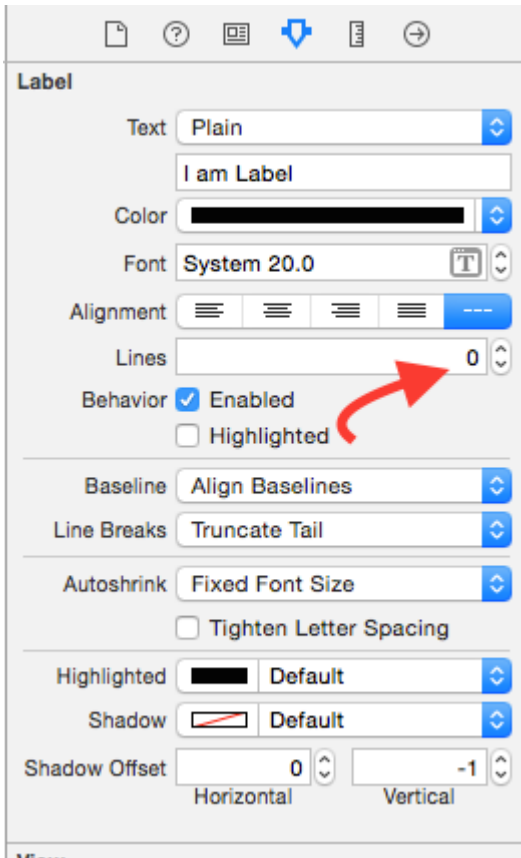
1. Führen 2) Unten 3) Nachlaufen 4) Feste Höhe (Von der Hauptansicht)



Ausgabe :-



Hinweis: - Stellen Sie sicher, dass Sie in der Eigenschaft Label die Anzahl der Zeilen = 0 festgelegt haben.



Ich hoffe, diese Informationen reichen aus, um die Autoresize-UIView nach der Höhe von UILabel und die Autoresize-UILabel nach Text zu verstehen.

Visual Format Language Basics: Einschränkungen im Code!

HVFL ist eine Sprache, mit der Benutzeroberflächenelemente auf einfache und schnelle Weise eingeschränkt werden können. Im Allgemeinen hat VFL gegenüber der traditionellen Benutzeroberflächenanpassung im Interface Builder einen Vorteil, da es deutlich lesbarer, zugänglicher und kompakter ist.

Hier ist ein Beispiel für VFL, bei dem drei UIViews von links nach rechts eingeschränkt sind, um `superView.width mit aGradeView zu aGradeView`

```
"H:|[bgView][aGradeView(40)][bGradeView(40)]|"
```

Es gibt zwei Achsen, auf die wir UI-Objekte einschränken können, horizontal und vertikal.

Jede Zeile von VFL beginnt immer mit `H:` oder `V:`. Wenn keine vorhanden ist, ist die Standardoption `H:`.

Weiter geht's, wir haben eine Pipeline. `|` Dieses Symbol oder die Pfeife bezieht sich auf den Überblick. Wenn Sie sich den Ausschnitt des VFL-Codes oben genauer ansehen, werden Sie zwei dieser Pipelines bemerken.

Dies bedeutet die zwei horizontalen Enden des Superview, die äußeren und äußeren Grenzen.

Als nächstes sehen Sie einige eckige Klammern, innerhalb der ersten eckigen Klammern haben

wir `bgView` . Wenn wir eckige Klammern haben, bezieht sich dies auf ein UI-Element. Nun fragen Sie sich vielleicht, wie wir eine Verbindung zwischen dem Namen und dem eigentlichen UI-Element herstellen, einem Auslass vielleicht?

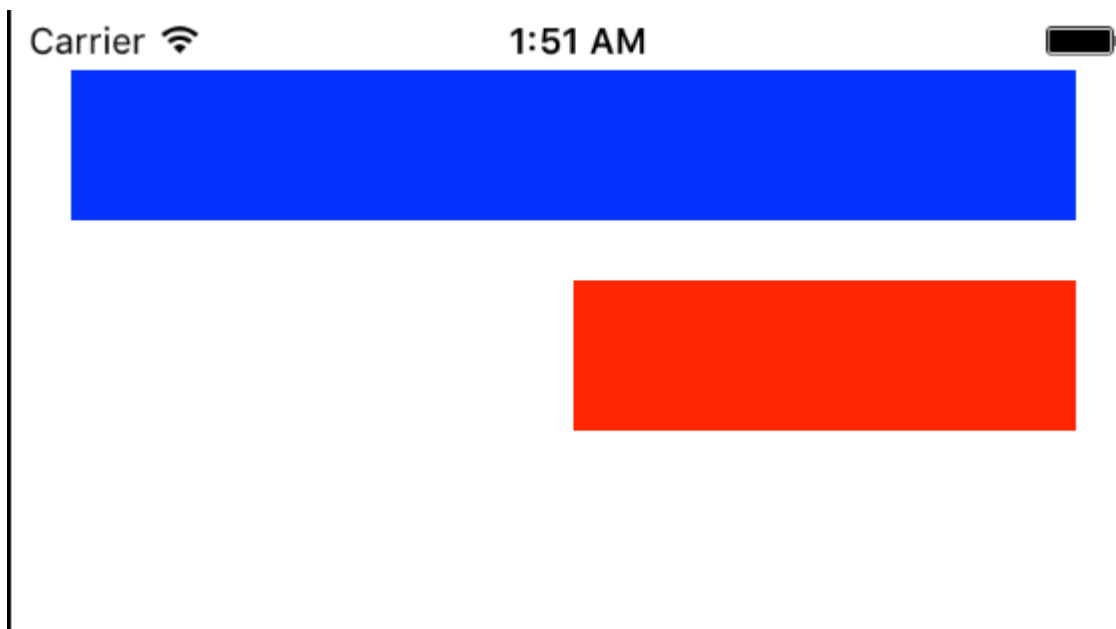
Ich werde das am Ende des Beitrags behandeln.

Wenn Sie sich das zweite Paar eckiger Klammern `[aGradeView(50)]` , werden auch einige Klammern darin eingeschlossen. Wenn dies vorhanden ist, wird die Breite / Höhe in Abhängigkeit von den Achsen definiert, in diesem Fall 50 Pixel in der Breite.

Die ersten eckigen Klammern `[bgView]` hatten keine explizit definierte Breite, was bedeutet, dass sie sich so weit wie möglich erstrecken wird.

Okay, das wars für die Grundlagen, mehr zu den fortgeschrittenen Sachen in einem anderen Beispiel.

zum Beispiel:



```
// 1. create views
UIView *blueView = [[UIView alloc] init];
blueView.backgroundColor = [UIColor blueColor];
[self.view addSubview:blueView];

UIView *redView = [[UIView alloc] init];
redView.backgroundColor = [UIColor redColor];
[self.view addSubview:redView];

// 2. forbid Autoresizing
blueView.translatesAutoresizingMaskIntoConstraints = NO;
redView.translatesAutoresizingMaskIntoConstraints = NO;

// 3. make constraints
// horizontal
NSArray *blueH = [NSLayoutConstraint constraintsWithVisualFormat:@"H:|-20-[blueView]-20-|"
options:NSLayoutFormatAlignAllLeft metrics:nil views:@{@"blueView" : blueView}];
[self.view addConstraints:blueH];
```

```

// vertical
NSArray *blueVandRedV = [NSLayoutConstraint constraintsWithVisualFormat:@"V:|-20-[blueView(50)]-20-[redView(==blueView)]" options:NSLayoutFormatAlignAllTrailing metrics:nil
views:@{@"blueView" : blueView, @"redView" : redView}];
[self.view addConstraints:blueVandRedV];

NSLayoutConstraint *redW = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
[self.view addConstraint:redW];

```

Gemischte Verwendung des automatischen Layouts mit nicht automatischem Layout

In einigen `UIKit` möchten Sie möglicherweise einige zusätzliche Aktionen für **automatische Layout-** Berechnungen ausführen, die von `UIKit` selbst ausgeführt werden.

Beispiel: Wenn Sie über eine `UIView`, die über einen `maskLayer`, müssen Sie möglicherweise den `maskLayer` aktualisieren, sobald **Auto Layout den** `frame` `UIView` ändert

```

// CustomView.m
- (void)layoutSubviews {
    [super layoutSubviews];
    // now you can assume Auto Layout did its job
    // you can use view's frame in your calculations
    CALayer maskLayer = self.maskLayer;
    maskLayer.bounds = self.bounds;
    ...
}

```

oder wenn Sie zusätzliche Aktionen für das **automatische Layout** in `ViewController`

```

- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    // now you can assume all your subviews are positioned/resized correctly
    self.customView.frame = self.containerView.frame;
}

```

Proportionales Layout

Einschränkung erstellt als

```

NSLayoutConstraint(item: myView, attribute: NSLayoutConstraint.Leading, relatedBy:
NSLayoutConstraint.Equal, toItem: view, attribute: NSLayoutConstraint.LeadingMargin, multiplier:
1.0, constant: 20.0)

```

oder aus mathematischer Sicht:

```

view.attribute * multiplier + constant (1)

```

Sie können den Multiplikator verwenden, um ein proportionales Layout für verschiedene

Größenfaktoren zu erstellen.

Beispiel:

Die Türkisansicht (V1) ist ein Quadrat mit einer Breite, die der Breite des Verhältnisses im Verhältnis 1: 1,1 entspricht

Gary Square (V2) ist eine Unteransicht von V1. Unterer Raum wird durch Konstante = 60, Nachlauf durch Multiplikator = 1,125 und Konstante = 0 festgelegt

Hinterer Raum proportional gesetzt, unterer Raum als Konstante.





- Ziel c

```
// 1. create views
UIView *blueView = [[UIView alloc] init];
blueView.backgroundColor = [UIColor blueColor];
[self.view addSubview:blueView];

UIView *redView = [[UIView alloc] init];
redView.backgroundColor = [UIColor redColor];
[self.view addSubview:redView];

// 2. forbid Autoresizing
blueView.translatesAutoresizingMaskIntoConstraints = NO;
redView.translatesAutoresizingMaskIntoConstraints = NO;

// 3. make constraints
// 3.1 blueView
NSLayoutConstraint *blueLeft = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeLeft multiplier:1 constant:20];
[self.view addConstraint:blueLeft];

NSLayoutConstraint *blueTop = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeTop multiplier:1 constant:20];
[self.view addConstraint:blueTop];

NSLayoutConstraint *blueRight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
[self.view addConstraint:blueRight];

NSLayoutConstraint *blueHeight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil
attribute:NSLayoutAttributeNotAnAttribute multiplier:1 constant:50];
[self.view addConstraint:blueHeight];

// 3.2 redView
NSLayoutConstraint *redTop = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeBottom multiplier:1 constant:20];
[self.view addConstraint:redTop];
```

```
NSLayoutConstraint *redRight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
[self.view addConstraint:redRight];

NSLayoutConstraint *redHeight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeHeight multiplier:1 constant:0];
[self.view addConstraint:redHeight];

NSLayoutConstraint *redWidth = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
[self.view addConstraint:redWidth];
```

Automatisches Layout online lesen: <https://riptutorial.com/de/ios/topic/792/automatisches-layout>

Kapitel 18: AVPlayer und AVPlayerViewController

Bemerkungen

AVKit importieren, AVFoundation importieren.

Examples

Wiedergabe von Medien mit AVPlayerViewController

Ziel c

```
NSURL *url = [[NSURL alloc] initWithString:@"YOUR URL"]; // url can be remote or local

AVPlayer *player = [AVPlayer playerWithURL:url];
// create a player view controller

AVPlayerViewController *controller = [[AVPlayerViewController alloc] init];
[self presentViewController:controller animated:YES completion:nil];
controller.player = player;
[player play];
```

Schnell

```
let player = AVPlayer(URL: url) // url can be remote or local

let playerViewController = AVPlayerViewController()
// creating a player view controller
playerViewController.player = player
self.presentViewController(playerViewController, animated: true) {

    playerViewController.player!.play()
}
```

Wiedergabe von Medien mit AVPlayer und AVPlayerLayer

Ziel c

```
NSURL *url = [NSURL URLWithString:@"YOUR URL"];
AVPlayer *player = [AVPlayer playerWithURL:videoURL];
AVPlayerLayer *playerLayer = [AVPlayerLayer playerLayerWithPlayer:player];
playerLayer.frame = self.view.bounds;
[self.view.layer addSublayer:playerLayer];
[player play];
```

Schnell

```
let url = NSURL(string: "YOUR URL")
let player = AVPlayer(URL: videoURL!)
let playerLayer = AVPlayerLayer(player: player)
playerLayer.frame = self.view.bounds
self.view.layer.addSublayer(playerLayer)
player.play()
```

AVPlayer-Beispiel

```
AVPlayer * avPlayer = [AVPlayer playerWithURL: [NSURL URLWithString: @"IHRE URL"]];
```

```
AVPlayerViewController *avPlayerCtrl = [[AVPlayerViewController alloc] init];
avPlayerCtrl.view.frame = self.view.frame;
avPlayerCtrl.player = avPlayer;
avPlayerCtrl.delegate = self;
[avPlayer play];
[self presentViewController:avPlayerCtrl animated:YES completion:nil
```

AVPlayer und AVPlayerViewController online lesen:

<https://riptutorial.com/de/ios/topic/5092/avplayer-und-avplayerviewController>

Kapitel 19: AVSpeechSynthesizer

Syntax

- AVSpeechSynthesizer () // Erzeugt einen Sprachsynthesizer
- speaker.speakUtterance (speech) // Wandelt den Text in Sprache um

Parameter

Parameter	Einzelheiten
Redner	AVSpeechSynthesizer-Objekt
Rede	AVSpeechUtterance-Objekt

Examples

Erstellen eines grundlegenden Textes zum Sprechen

Verwenden Sie die `speakUtterance:` -Methode von `AVSpeechSynthesizer` , um Text in Sprache zu konvertieren. Sie müssen ein `AVSpeechUtterance` Objekt an diese Methode übergeben, die den Text enthält, der gesprochen werden soll.

Ziel c

```
AVSpeechSynthesizer *speaker = [[AVSpeechSynthesizer alloc] init];
AVSpeechUtterance *speech     = [AVSpeechUtterance speechUtteranceWithString:@"Hello World"];
[speaker speakUtterance:speech];
```

Schnell

```
let speaker = AVSpeechSynthesizer()
let speech = AVSpeechUtterance(string: "Hello World")
speaker.speakUtterance(speech)
```

AVSpeechSynthesizer online lesen: <https://riptutorial.com/de/ios/topic/1526/avspeechsynthesizer>

Kapitel 20: AWS SDK

Examples

Laden Sie ein Bild oder ein Video mit AWS SDK in S3 hoch

Bevor Sie mit dem Beispiel beginnen, würde ich empfehlen, ein Singleton mit einem Delegate-Klassenmitglied zu erstellen, sodass Sie die Möglichkeit haben könnten, eine Datei im Hintergrund hochzuladen, und den Benutzer die Verwendung Ihrer App weiterhin zulassen, während die Dateien auch während der App hochgeladen werden ist der Hintergrund.

Beginnen wir mit dem Erstellen eines Enums, das die S3-Konfiguration darstellt:

```
enum S3Configuration : String
{
    case IDENTITY_POOL_ID = "YourIdentityPoolId"
    case BUCKET_NAME = "YourBucketName"
    case CALLBACK_KEY = "YourCustomStringForCallBackWhenUploadingInTheBackground"
    case CONTENT_TYPE_IMAGE = "image/png"
    case CONTENT_TYPE_VIDEO = "video/mp4"
}
```

Jetzt sollten wir die Anmeldeinformationen festlegen, wenn Ihre App zum ersten Mal `AppDelegate` `didFinishLaunchingWithOptions` sollten Sie sie in der `didFinishLaunchingWithOptions` Methode der Methode `AppDelegate` `didFinishLaunchingWithOptions` (achten Sie darauf, dass Sie Ihre Region auf `regionType` param setzen):

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool
{
    let credentialProvider = AWSCognitoCredentialsProvider(regionType: .EUWest1, identityPoolId:
S3Configuration.IDENTITY_POOL_ID.rawValue)
    let configuration = AWSServiceConfiguration(region: .EUWest1, credentialsProvider:
credentialProvider)
    AWSS3TransferUtility.registerS3TransferUtilityWithConfiguration(configuration, forKey:
S3Configuration.CALLBACK_KEY.rawValue)
}
```

Da wir uns bereits in `AppDelegate` befinden, sollten wir den Hintergrund-Callback implementieren, der vom AWS SDK verarbeitet wird:

```
func application(application: UIApplication, handleEventsForBackgroundURLSession identifier:
String, completionHandler: () -> Void)
{
    // Will print the identifier you have set at the enum: .CALLBACK_KEY
    print("Identifier: " + identifier)
    // Stores the completion handler.
    AWSS3TransferUtility.interceptApplication(application,
                                             handleEventsForBackgroundURLSession: identifier,
                                             completionHandler: completionHandler)
}
```

Wenn der Benutzer die App in den Hintergrund verschiebt, wird der Upload nun fortgesetzt.

Um die Datei mit dem AWS SDK hochladen zu können, müssen Sie die Datei auf das Gerät schreiben und dem SDK den tatsächlichen Pfad angeben. Stellen Sie sich vor, wir hätten ein UIImage (könnte auch ein Video sein ...) und wir schreiben es in einen temporären Ordner:

```
// Some image....
let image = UIImage()
let fileURL = NSURL(fileURLWithPath:
NSTemporaryDirectory()).URLByAppendingPathComponent(fileName)
let filePath = fileURL.path!
let imageData = UIImageJPEGRepresentation(image, 1.0)
imageData!.writeToFile(filePath, atomically: true)
```

FileURL und Dateiname werden später für das eigentliche Hochladen verwendet.

Es gibt zwei Schließungen, die wir definieren müssen, die vom AWS SDK bereitgestellt werden.

1. `AWSS3TransferUtilityUploadCompletionHandlerBlock` - Eine Schließung, die benachrichtigt, wenn der Upload abgeschlossen ist (oder nicht).
2. `AWSS3TransferUtilityUploadProgressBlock` - Eine Schließung, die jedes gesendete Byte benachrichtigt

Wenn Sie ein Singleton planen, sollten Sie diese Typen als Klassenmitglieder definieren. Die Implementierung sollte folgendermaßen aussehen:

```
var completionHandler : AWSS3TransferUtilityUploadCompletionHandlerBlock? =
    { (task, error) -> Void in

        if ((error) != nil)
        {
            print("Upload failed")
        }
        else
        {
            print("File uploaded successfully")
        }
    }

var progressBlock : AWSS3TransferUtilityUploadProgressBlock? =
    { [unowned self] (task, bytesSent: Int64, totalBytesSent: Int64,
totalBytesExpectedToSend: Int64) -> Void in

        let progressInPercentage = Float(Double(totalBytesSent) /
Double(totalBytesExpectedToSend)) * 100
        print(progressInPercentage)
    }
```

ANMERKUNG: Wenn Sie ein Singleton verwenden, möchten Sie möglicherweise einen Delegaten definieren, der den Fortschritt oder die Fertigstellung der Datei anzeigt. Wenn Sie kein Singleton verwenden, können Sie eine statische Methode mit den entsprechenden Typen erstellen:

```
static func uploadImageToS3(fileURL : NSURL,
```



```

        fileName : String,
        progressFunctionUpdater : Float -> Void,
        resultBlock : (NSError?) -> Void)
{
    // Actual implementation .....
    // ...
    // ...
}

```

1. `progressFunctionUpdater` - berichtet an eine Funktion mit Fortschritt.
2. `resultBlock` - Wenn Sie nil zurückgegeben haben und der Upload erfolgreich war, wurde das `resultBlock`

Sehr geehrte Damen und Herren, der eigentliche Upload:

```

let fileData = NSData(contentsOfFile: fileURL.relativePath!)

let expression = AWSS3TransferUtilityUploadExpression()
expression.uploadProgress = progressBlock

let transferUtility =
AWSS3TransferUtility.S3TransferUtilityForKey(S3Configuration.CALLBACK_KEY.rawValue)

transferUtility?.uploadData(fileData!,
    bucket: S3Configuration.BUCKET_NAME.rawValue,
    key: fileName,
    contentType: S3Configuration.CONTENT_TYPE_IMAGE.rawValue,
    expression: expression,
    completionHandler: completionHandler).continueWithBlock
{ (task : AWSTask) -> AnyObject? in

    if let error = task.error
    {
        print(error)
    }
    if let exception = task.exception
    {
        print("Exception: " + exception.description)
    }
    if let uploadTask = task.result as? AWSS3TransferUtilityUploadTask
    {
        print("Upload started...")
    }

    return nil
}

```

Glückliches S3 hochladen :)

AWS SDK online lesen: <https://riptutorial.com/de/ios/topic/4734/aws-sdk>

Kapitel 21: Beacons mit CoreBluetooth konfigurieren

Einführung

Hot zum Lesen und Schreiben von Daten auf ein Bluetooth-Gerät mit niedrigem Energieverbrauch.

Bemerkungen

Einige wichtige Punkte

- Es sind keine Fähigkeiten erforderlich.
- iPhone speichert Bytes im Little Endian-Format. Überprüfen Sie daher, ob Bluetooth-Zubehör auch Little Endian verwendet. Beispiel:
 - Intel-CPU verwendet normalerweise Little Endian.
 - Die ARM-Architektur war Little Endian vor der Version 3, als sie Big-Endian wurde.
- Nach einem Einzel- oder Batch-Vorgang geht die Verbindung verloren, daher müssen Sie die Verbindung wiederherstellen, bevor Sie fortfahren.

Scannen Sie nach SERVICE UUID

```
func SearchBLE() {
    cb_manager.scanForPeripherals(withServices:[service_uuid], options: nil)
    StopSearchBLE()
}
```

So entdecken Sie SERVICE UUID ohne Dokumentation

```
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
    peripheral.delegate = self
    peripheral.discoverServices(nil)
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    for service in peripheral.services! {
        print("Service: \(service)\n error: \(error)")
    }
}
```

- `discoverServices (nil)` - NIL bedeutet, dass alle Dienste zurückgegeben werden, was keine gute Option ist. (LESEN Sie die Bemerkungen 3).

- Wenn Sie die SERVICE-UUID nicht gefunden haben, führen Sie Ihren Code aus und suchen Sie in der Konsole

```
Service: <CBService: 0x171e75280, isPrimary = YES, UUID = Battery>
error: nil
Service: <CBService: 0x171e74c40, isPrimary = YES, UUID = Device Information>
error: nil
Service: <CBService: 0x171e75300, isPrimary = YES, UUID = FFF0>
error: nil
```

- Ich habe 3 Dienste gefunden: Batterie, Geräteinformationen (Firmware) und FFF0
- Dieser uuid-Dienst ist kein Standardservice. Eine Liste mit Standards finden Sie [hier](#)
- In diesem Fall ist FFF0 die SERVICE-UUID

Konvertieren Sie Daten in UInt16 und umgekehrt

Fügen Sie diese Erweiterungen Ihrer Klasse hinzu

```
protocol DataConvertible {
    init?(data: Data)
    var data: Data { get }
}

extension DataConvertible {

    init?(data: Data) {
        guard data.count == MemoryLayout<Self>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }

    var data: Data {
        var value = self
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}

extension UInt16 : DataConvertible {
    init?(data: Data) {
        guard data.count == MemoryLayout<UInt16>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }

    var data: Data {
        var value = CFSwapInt16HostToBig(self)
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}
```

Examples

Namen aller Bluetooth Low Energy (BLE) anzeigen

- Für dieses Beispiel habe ich einen kontrollierten Raum mit einer einzigen BLE-Gerätefreigabe.
- Ihre Klasse sollte CBCentralManagerDelegate erweitern.

- Implementieren Sie die Methode: `centralManagerDidUpdateState` (`_ central: CBCentralManager`).
- Verwenden Sie die globale Warteschlange, um den Bildschirm bei der Suche nach einem Gerät nicht einzufrieren.
- Instanzieren Sie `CBCentralManager` und warten Sie auf die Antwort von Callback `centralManagerDidUpdateState`.

```
class BLEController: CBCentralManagerDelegate{

var cb_manager: CBCentralManager!
var bles : [CBPeripheral] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        cb_manager = CBCentralManager(delegate: self, queue: DispatchQueue.global())
    }

    func centralManagerDidUpdateState(_ central: CBCentralManager) {
        print("UPDATE STATE - \(central)")
    }
}
```

Ein Rückruf an `centralManagerDidUpdateState` zeigt an, dass CoreBluetooth bereit ist, sodass Sie jetzt nach BLE suchen können. Aktualisieren Sie den `centralManagerDidUpdateState`-Code, um nach allen BLE-Geräten zu suchen, wenn sie bereit sind.

```
func centralManagerDidUpdateState(_ central: CBCentralManager) {
    print("UPDATE STATE - \(central)")
    SearchBLE()
}

func SearchBLE(){
    cb_manager.scanForPeripherals(withServices: nil, options: nil)
    StopSearchBLE()
}

func StopSearchBLE() {
    let when = DispatchTime.now() + 5 // change 5 to desired number of seconds
    DispatchQueue.main.asyncAfter(deadline: when) {
        self.cb_manager.stopScan()
    }
}
```

- `SearchBLE ()` sucht nach BLE-Geräten und stoppt die Suche nach 5s
- `cb_manager.scanForPeripherals (withServices: nil, options: nil)` sucht nach jeder BLE, die sich in Reichweite befindet.
- `StopSearchBLE ()` stoppt die Suche nach 5s.
- Jede gefundene BLE ruft den `func centralManager (_ central: CBCentralManager, didDiscover-Peripherie: CBPeripheral, advertisementData: [String: Any], RSSI: NSNumber)` auf

```
func centralManager(_ central: CBCentralManager, didDiscover peripheral:
    CBPeripheral, advertisementData: [String : Any], rssi RSSI: NSNumber) {
```

```

guard let name = peripheral.name else {
    return
}
print(name)
bles.append(peripheral)
}

```

Verbinden Sie und lesen Sie den wichtigsten Wert

- Ich befinde mich in einem kontrollierten Raum mit einem einzigen Minew Beacon, der das IBEACON-Protokoll verwendet.
- BLEController muss CBPeripheralDelegate erweitern
- Ich verwende die erste BLE, um eine Verbindung herzustellen, nachdem die Suche beendet wurde.
- Ändern Sie die Methode StopSearchBLE ().

```

class BLEController: CBCentralManagerDelegate, CBPeripheralDelegate{
//...
    func StopSearchMiniewBeacon() {
        let when = DispatchTime.now() + 5 // change 2 to desired number of seconds
        DispatchQueue.main.asyncAfter(deadline: when) {
            self.cb_manager.stopScan()
            self.cb_manager.connect(bles.first)
        }
    }
}
//...
}

```

- In der Dokumentation Ihres BLE-Geräts sollten Sie nach SERVICE UUID und MAJOR UUID CHARACTERISTIC suchen

```

var service_uuid = CBUUID(string: "0000fff0-0000-1000-8000-00805f9b34fb")
var major_uuid = CBUUID(string: "0000fff2-0000-1000-8000-00805f9b34fb")
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
    peripheral.delegate = self
    peripheral.discoverServices([service_uuid])
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    print("Service: \(service)\n error: \(error)")
    peripheral.discoverCharacteristics([major_uuid], for: (peripheral.services?[0])!)
}

```

- Erstellen Sie eine Variable wie "service_uuid" und "major_uuid" wie oben. '-0000-1000-8000-00805f9b34fb' gehört zum Standard. 'fff0' ist meine SERVICE-UUID, 'fff2' ist meine wichtige UUID-Eigenschaft und '0000' ist erforderlich, um den 4-Byte-Uuid-1⁰-Block zu füllen.
- discoverCharacteristics ([major_uuid], für: (peripherie.services?[0])!) wird die Hauptcharakteristik von meinem gatt-Server meines Geräts erhalten, und es wird NIL als Wert für jetzt angezeigt.
- (peripherie.services?[0])! - 0 beacuse gibt einen einzelnen Wert zurück, sobald ich peripher.discoverServices ([service_uuid]) ausgeführt habe.

```

func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
            peripheral.readValue(for: characteristic)
        }
    }
}

func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
}

```

- Der Merkmalswert ist nur nach dem Aufruf `peripher.readValue` (für: Merkmal) lesbar.
- `readValue` führt zu einem Peripheriegerät (`_ Peripheriegerät: CBPeripheral`, `didUpdateValueFor`-Eigenschaft: `CBCharacteristic`, Fehler: Fehler?) mit dem Wert in Datentyp.

Schreibe den Hauptwert

- Sie müssen die Dienste und Merkmale entdecken
- Sie müssen den Wert des Merkmals nicht lesen, bevor Sie darüber schreiben.
- wird für dieses Beispiel nach dem gelesenen Wert fortgesetzt. Ändern Sie das Peripheriegerät (`_ Peripheriegerät: CBPeripheral`, `didUpdateValueFor`-Eigenschaft: `CBCharacteristic`, Fehler: Fehler?).
- Fügen Sie eine Variable `new_major` und `reset_characteristic` hinzu

```

var reset_characteristic : CBCharacteristic!
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
            peripheral.readValue(for: characteristic)
        }
        if(characteristic.uuid.uuidString == "FFFF"){
            reset_characteristic = characteristic
        }
    }
}

let new_major : UInt16 = 100
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
    peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
}

```

- Das iPhone von default sendet und empfängt Bytes im Little Endian-Format, aber mein

Gerät MINEW, bei dem der NRF51822-Chipsatz ARM-Architektur besitzt, benötigt Bytes im Big Endian-Format.

- In der Dokumentation zu BLE-Geräten wird angegeben, welche Art von Eingabe und Ausgabe jedes Merkmal hat und ob Sie es wie oben lesen können (CBCharacteristicWriteType.withResponse).

```
func peripheral(_ peripheral: CBPeripheral, didWriteValueFor characteristic: CBCharacteristic,
error: Error?) {
    print("Characteristic write: \(characteristic)\n error: \(error)")
    if(characteristic.uuid.uuidString == "FFF2"){
        print("Resetting")
        peripheral.writeValue("minew123".data(using: String.Encoding.utf8)!, for:
reset_characteristic, type: CBCharacteristicWriteType.withResponse)
    }
    if(characteristic.uuid.uuidString == "FFFF"){
        print("Reboot finish")
        cb_manager.cancelPeripheralConnection(peripheral)
    }
}
```

- Um die Informationen eines gatt-Servers zu aktualisieren, müssen Sie ihn programmgesteuert neu starten oder Daten darauf speichern und manuell ausschalten und einschalten.
- FFFF ist charakteristisch für dieses Gerät.
- In diesem Fall ist 'minew123' das Standardkennwort für den Neustart oder das Speichern von Informationen.
- Starten Sie Ihre App und beobachten Sie die Konsole auf Fehler, ich hoffe nicht, aber Sie werden den neuen Wert noch nicht sehen.

```
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
    //peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
```

```
}
```

- Der letzte Schritt besteht darin, die letzte Zeile in der Methode didUpdateValueFor zu kommentieren und die App erneut auszuführen. Jetzt wird der neue Wert angezeigt.

Beacons mit CoreBluetooth konfigurieren online lesen:

<https://riptutorial.com/de/ios/topic/9488/beacons-mit-corebluetooth-konfigurieren>

Kapitel 22: Behandeln Sie mehrere Umgebungen mit Makro

Examples

Behandeln Sie mehrere Umgebungen mit mehreren Zielen und Makros

Zum Beispiel haben wir zwei Umgebungen: CI - Staging und möchten einige Anpassungen für jede Umgebung hinzufügen. Hier werde ich versuchen, die Server-URL und den App-Namen anzupassen.

Zunächst erstellen wir zwei Ziele für zwei Umgebungen, indem wir das Hauptziel duplizieren:

- MultipleEnvironments M
 - MultipleEnvironments
 - AppDelegate.h
 - AppDelegate.m
 - ViewController.h
 - ViewController.m
 - Main.storyboard
 - Assets.xcassets
 - LaunchScreen.storyboard
 - Info.plist
 - Supporting Files
 - Products
 - MultipleEnviron...s copy-Info.plist A
 - CI copy-Info.plist A

- PROJECT
 - MultipleEnvironme...
- TARGETS
 - MultipleEnvironme...
 - CI
 - Staging

General

Identity

Deployment Info

App Icons and Launch Images

Embedded Binaries

Linked Frameworks and Libraries

- Wenn wir mit CI-Ziel ausführen / archivieren, lautet die SERVER_URL
<http://ci.api.example.com/>
<http://ci.api.example.com/>
- Wenn wir mit STAGING target ausführen / archivieren, lautet die SERVER_URL
<http://stg.api.example.com/>

Wenn Sie weitere Anpassungen vornehmen möchten, ändern Sie beispielsweise den App-Namen für jedes Ziel:



Xcode

File

Edit

View

Find

Navigate



CI >



iPhone 6s



M



MultipleEnvironments

M



MultipleEnvironments



AppDelegate.h



AppDelegate.m



ViewController.h



ViewController.m



Main.storyboard



Assets.xcassets



LaunchScreen.storyboard



Info.plist



Supporting Files



AppConfigurations.h

A



Products



MultipleEnviron...s copy-Info.plist

A



CI copy-Info.plist

A



PROJECT



MultipleEnv

TARGETS



MultipleEnv



CI



Staging



MultipleEn...



App CI



App STG

<https://riptutorial.com/de/ios/topic/6849/behandeln-sie-mehrere-umgebungen-mit-makro>

Kapitel 23: Benutzerdefinierte Auswahlmethoden für UITableViewCells

Einführung

Erweiterte Möglichkeiten zum Verwalten von Auswahlen von UITableViewCell. Beispiele, wenn das einfache `didSelect...` UITableViewDelegate nicht ausreicht, um etwas zu erreichen.

Examples

Unterscheidung zwischen Einzel- und Doppelauswahl in Zeile.

Ein Implementierungsbeispiel, das die Möglichkeit bietet, festzustellen, ob ein Benutzer UITableViewCell ein- oder zweimal antippen muss

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Benutzerdefinierte Auswahlmethoden für UITableViewCells online lesen:

<https://riptutorial.com/de/ios/topic/9961/benutzerdefinierte-auswahlmethoden-fur-uitableviewcells>

Kapitel 24: Benutzerdefinierte Auswahlmethoden für UITableViewCells

Examples

Unterscheidung zwischen Einzel- und Doppelauswahl in Zeile.

Ein Beispiel für die Implementierung von UITableView, mit der festgestellt werden kann, ob die Zelle einmal oder doppelt gezählt wurde.

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Benutzerdefinierte Auswahlmethoden für UITableViewCells online lesen:

<https://riptutorial.com/de/ios/topic/9962/benutzerdefinierte-auswahlmethoden-fur-uitableviewcells>

Kapitel 25: Benutzerdefinierte Schriftarten

Examples

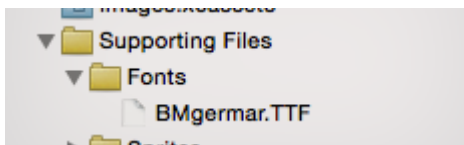
Einbetten benutzerdefinierter Schriftarten

Unterstützung für benutzerdefinierte Schriftarten

Anwendungen, die benutzerdefinierte Schriftarten verwenden möchten, können diese Schriftarten jetzt in ihr Anwendungspaket aufnehmen und sie beim System registrieren, indem sie den `UIAppFonts`-Schlüssel in ihre `Info.plist`-Datei aufnehmen. Der Wert dieses Schlüssels ist ein Array von Zeichenfolgen, das die Zeichensatzdateien im Bundle der Anwendung identifiziert. Wenn das System den Schlüssel sieht, lädt es die angegebenen Schriftarten und stellt sie der Anwendung zur Verfügung.

Nachdem die Schriftarten in der `Info.plist`, können Sie Ihre benutzerdefinierten Schriftarten wie jede andere Schriftart in IB oder programmgesteuert verwenden.

1. Ziehen Sie die Schriftart in den Ordner "Xcode-Unterstützungsdateien". Vergessen Sie nicht, Ihre App im Abschnitt "Zu Zielen hinzufügen" zu markieren. Ab diesem Zeitpunkt können Sie diese Schrift in IB verwenden und aus der Schriftpalette auswählen.



2. Um diese Schriftart auf dem Gerät verfügbar zu machen, öffnen Sie `Info.plist` und fügen Sie `Fonts provided by application key (UIAppFonts)` `Fonts provided by application key` hinzu. Fügen Sie dem Schlüssel 0 den Schriftnamen als Wert hinzu. Hinweis: Der Name der Schriftart kann von der Schriftartdatei abweichen.

▼ Fonts provided by application	Array	(1 item)
Item 0	String	BMgermar.TTF

3. Rufen Sie den benutzerdefinierten hinzugefügten Schriftartnamen mit dem folgenden Snippet ab

[*Swift 3*]

```
for family in UIFont.familyNames {
    print("\(family)")

    for name in UIFont.fontNames(forFamilyName: family) {
        print("    \(name)")
    }
}
```

[*Ziel - C*]

```

for (NSString *familyName in [UIFont familyNames]){
    NSLog(@"Family name: %@", familyName);
    for (NSString *fontName in [UIFont fontNamesForFamilyName:familyName]) {
        NSLog(@"--Font name: %@", fontName);
    }
}

```

Benutzerdefinierte Schriftarten mit Storyboard

Benutzerdefinierte Schriftarten für UI-Komponenten aus dem Storyboard können mit [benutzerdefinierten Laufzeitattributen](#) in Storyboard und [Kategorien](#) problemlos erstellt werden.

Die Vorteile sind wie

- Es müssen keine Auslässe für das Ui-Element definiert werden
- Es ist nicht erforderlich, die Schriftart für Elemente programmgesteuert festzulegen.

Schritte zum folgen

1. **Font - Datei:** Fügen Sie die Font - Datei (.ttf) mit dem Anwendungspaket und den Eintrag für die Schriftart in Info.plist unter **Font durch Anwendung** wie in diesem [bereitgestellt](#) hinzufügen [Dokumentation](#) individueller Schriftarten.
2. **Kategorien definieren:** Fügen Sie eine Datei wie **UIKit + IBExtensions hinzu** und fügen Sie die Kategorien für UI-Elemente wie UILabel, UIButton usw. hinzu, für die Sie eine benutzerdefinierte Schriftart festlegen möchten. Alle Kategorien **verfügen** über eine benutzerdefinierte Eigenschaft, beispielsweise **fontName** . Dies wird später im Storyboard zum Festlegen der benutzerdefinierten Schriftart verwendet (wie in Schritt 4).

UIKit + IBExtensions.h

```

#import <UIKit/UIKit.h>

//Category extension for UILabel
@interface UILabel (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UITextField
@interface UITextField (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UIButton
@interface UIButton (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

```

3. **Getter und Setter:** Definieren Sie für jede hinzugefügte Kategorie Getter und Setter für die Eigenschaft `fontName`.

UIKit + IBExtensions.m

```
#import "UIKit+IBExtensions.h"

@implementation UILabel (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}

@end

@implementation UITextField (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}

@end

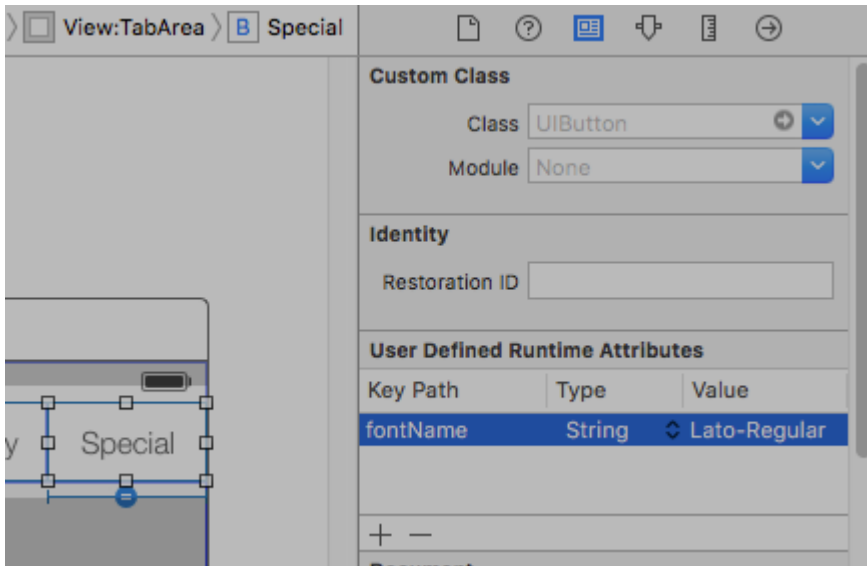
@implementation UIButton (IBExtensions)

- (NSString *)fontName {
    return self.titleLabel.font.fontName;
}

- (void)setFontName:(NSString *)fontName{
    self.titleLabel.font = [UIFont fontWithName:fontName size:self.titleLabel.font.pointSize];
}

@end
```

4. **Festlegen der Schriftart im Storyboard:** Fügen Sie in den benutzerdefinierten Laufzeitattributen einen Eintrag mit **fontName** als keyPath und **den Namen** Ihrer **benutzerdefinierten Schriftart** als Typ mit dem Typ String als **Zeichenfolge** hinzu.



Dadurch wird Ihre benutzerdefinierte Schriftart festgelegt, während Sie die App ausführen.

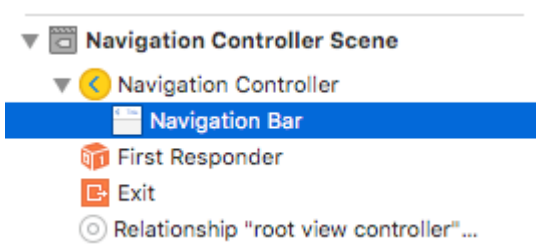
Anmerkungen:

- Lato-Regular ist die benutzerdefinierte Schriftart, die ich verwendet habe.
- Der gleiche Name in der im Bundle hinzugefügten **.ttf**-Datei sollte ohne Erweiterung im Storyboard verwendet werden.
- Die Schriftgröße ist dieselbe wie im Attribut-Inspektor des Oberflächenelements definiert.

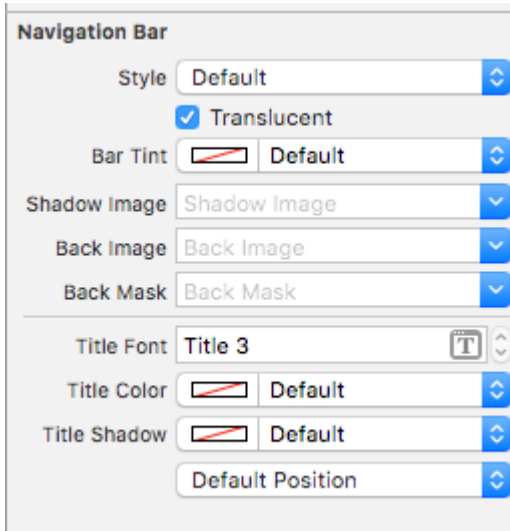
Anwenden benutzerdefinierter Schriftarten auf Steuerelemente in einem Storyboard

Das folgende Beispiel zeigt, wie Sie benutzerdefinierte Schriftarten auf eine Navigationsleiste anwenden, und enthält Korrekturen für einige schrullige Verhaltensweisen in Xcode. Sie können die benutzerdefinierten Schriftarten auch auf **andere UIControls** wie **UILabels**, **UIButtons** usw. **anwenden**, indem Sie den Attributinspektor verwenden, nachdem die benutzerdefinierte Schriftart dem Projekt hinzugefügt wurde. Bitte beachten Sie die externen Links zu Arbeitsproben und Videos im unteren Bereich.

1. Wählen Sie Ihre Navigationsleiste in Ihrem Navigationscontroller aus



2. Ändern Sie die Titelschriftart im Bereich "Attribute"

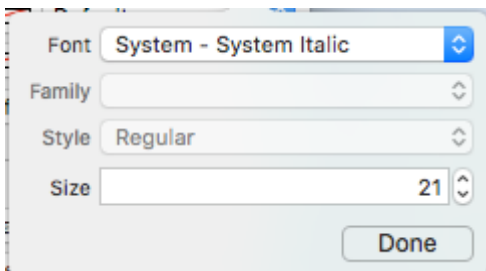


(Sie müssen wahrscheinlich den Balkenfarbton für die Navigationsleiste umschalten, bevor Xcode die neue Schriftart aufnimmt.)

Hinweise (Vorsichtsmaßnahmen)

Verifiziert, dass dies unter Xcode 7.1.1+ funktioniert. (**Siehe die Beispiele unten**)

1. Sie müssen den Farbton der Nav-Leiste umschalten, bevor die Schrift wirksam wird (scheint in Xcode ein Fehler zu sein; Sie können sie auf die Standardeinstellung zurücksetzen und die Schrift bleibt hängen).
2. Wenn Sie eine Systemschriftart auswählen ~ Stellen Sie sicher, dass die Größe nicht 0.0 ist (andernfalls wird die neue Schriftart ignoriert).



3. Anscheinend funktioniert das problemlos, wenn sich nur eine NavBar in der Ansichtshierarchie befindet. Es scheint, dass sekundäre NavBars im selben Stack ignoriert werden. (Wenn Sie die NavBar des Master-Navigationscontrollers anzeigen, werden alle anderen benutzerdefinierten NavBar-Einstellungen ignoriert.)

Gotchas (deux)

Einige davon wiederholen sich, was sehr wahrscheinlich eine Bemerkung wert ist.

1. Manchmal wird das Storyboard-XML beschädigt. Dazu müssen Sie die Struktur im Storyboard als Quellcode-Modus überprüfen (Rechtsklick auf die Storyboard-Datei > Öffnen als ...).
2. In einigen Fällen wurde das mit dem benutzerdefinierten Laufzeitattribut verknüpfte

navigationItem-Tag anstelle des view controller-Tags als untergeordnetes XML-Element des view-Tags festgelegt. Wenn dies der Fall ist, entfernen Sie es zwischen den Tags, um einen ordnungsgemäßen Betrieb zu gewährleisten.

3. Schalten Sie den NavBar-Farbton um, um sicherzustellen, dass die benutzerdefinierte Schriftart verwendet wird.
4. Überprüfen Sie den Größenparameter der Schriftart, sofern Sie keinen dynamischen Schriftstil verwenden
5. Ansichtshierarchie überschreibt die Einstellungen. Es scheint, dass eine Schriftart pro Stapel möglich ist.

Ergebnis

Proben

- [Video, das mehrere Schriftarten im erweiterten Projekt zeigt](#)
- [Einfacher Quelldownload](#)
- [Erweiterter Projekt-Download ~ Zeigt mehrere NavBar-Schriften und Workaround für benutzerdefinierte Schriften](#)
- [Video mit mehreren Schriftarten und benutzerdefinierten Schriftarten](#)

Umgang mit benutzerdefinierten Schriftarten

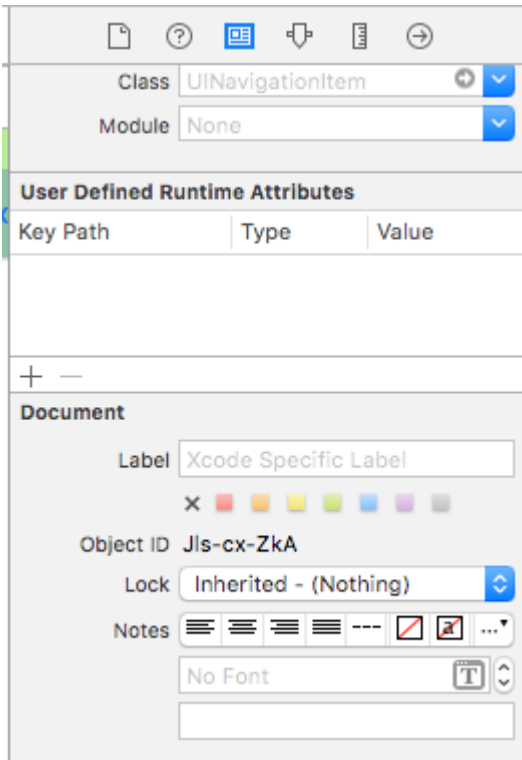
Hinweis ~ Eine [nette Checkliste](#) ist auf der Code With Chris-Website zu finden und Sie können das Beispiel-Download-Projekt sehen.

Wenn Sie eine eigene Schriftart haben und diese in Ihrem Storyboard verwenden möchten, finden Sie in der folgenden [SO-Frage](#) eine anständige Antwort. Eine Antwort identifiziert diese Schritte.

1. Erhalten Sie Ihre benutzerdefinierte Schriftartdatei (.ttf, .ttc)
2. Importieren Sie die Schriftdateien in Ihr Xcode-Projekt
3. Fügen Sie in app-info.plist einen Schlüssel mit dem Namen "Fonts" hinzu, der von der Anwendung bereitgestellt wird. Fügen Sie dem Array alle Array-Dateinamen hinzu.
Anmerkung: einschließlich der Dateierweiterung.
4. Klicken Sie im Storyboard in der Navigationsleiste auf den Attributinspektor, und klicken Sie auf die Schaltfläche mit dem rechten Symbol im Bereich zur Auswahl von Schriftarten.

Umgehung der benutzerdefinierten Schriftart

Daher sieht Xcode natürlich so aus, als könnte er mit benutzerdefinierten Schriftarten in UINavigationController umgehen, diese Funktion wird jedoch nicht ordnungsgemäß aktualisiert (die ausgewählte Schriftart wird ignoriert).



Um dieses Problem zu umgehen:

Eine Möglichkeit besteht darin, die Verwendung des Storyboards und das Hinzufügen einer Codezeile zu korrigieren: Fügen Sie dem View Controller zunächst eine UIView (UIButton, UILabel oder eine andere UIView-Unterklasse) hinzu (nicht das Navigationselement ...). Das). Nachdem Sie das Steuerelement hinzugefügt haben, können Sie die Schriftart im Storyboard ändern und Ihrem View Controller eine Referenz als Auslass hinzufügen. Weisen Sie diese Ansicht einfach der UINavigationController.titleView zu. Sie können den Textnamen ggf. auch im Code festlegen. Gemeldeter Fehler (23600285).

```
@IBOutlet var customFontTitleView: UIButton!  
  
//Sometime later...  
self.navigationItem.titleView = customFontTitleView
```

Hinweis - Dieses Beispiel ist von einer Antwort abgeleitet, die ich zu SO ([hier](#)) gepostet habe.

Benutzerdefinierte Schriftarten online lesen:

<https://riptutorial.com/de/ios/topic/1504/benutzerdefinierte-schriftarten>

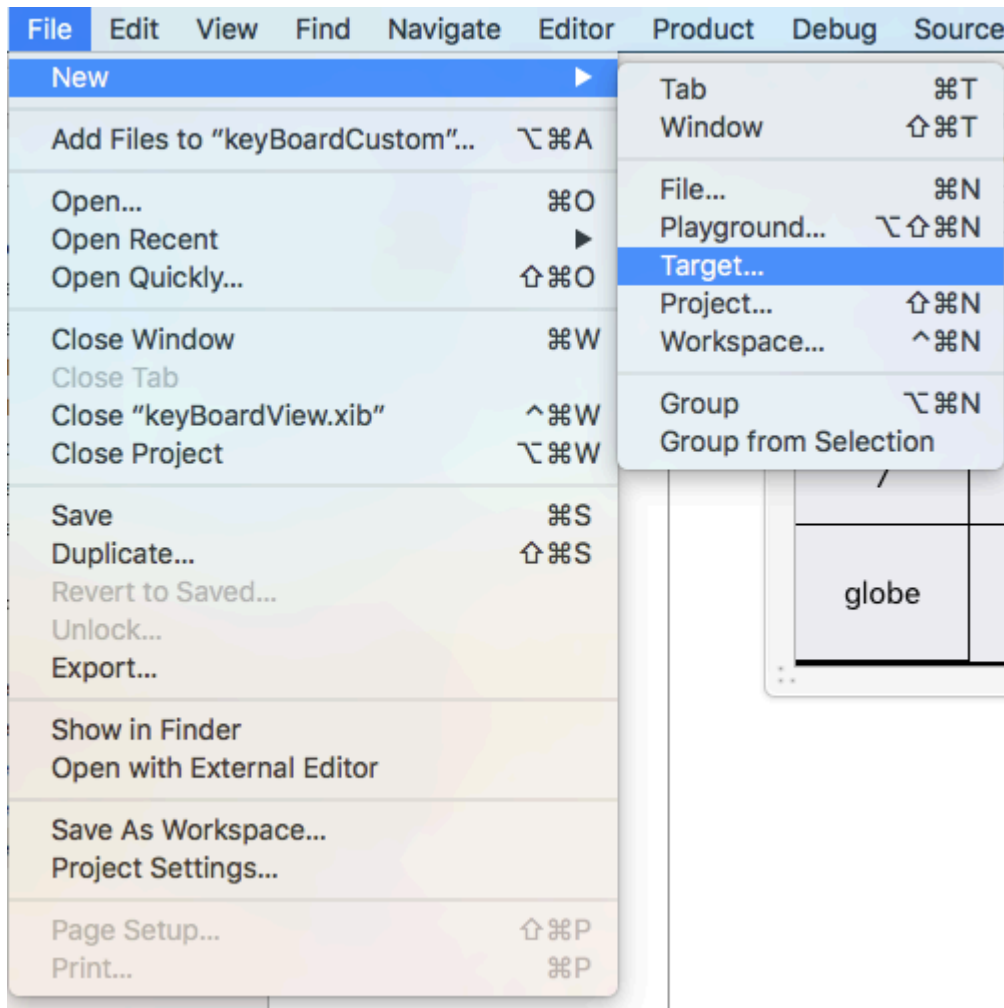
Kapitel 26: Benutzerdefinierte Tastatur

Examples

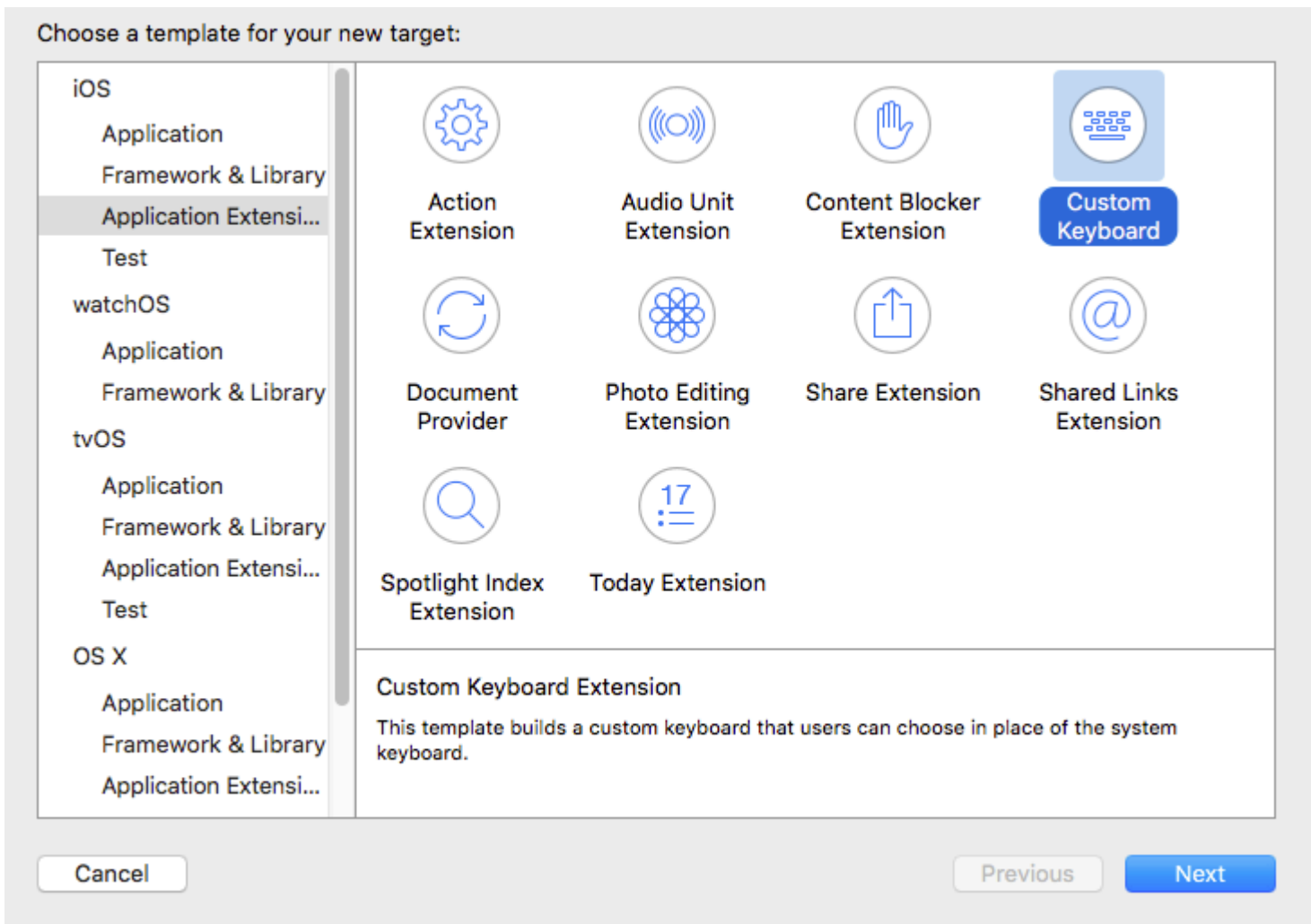
Custom KeyBoard-Beispiel

Ziel-C und Xib

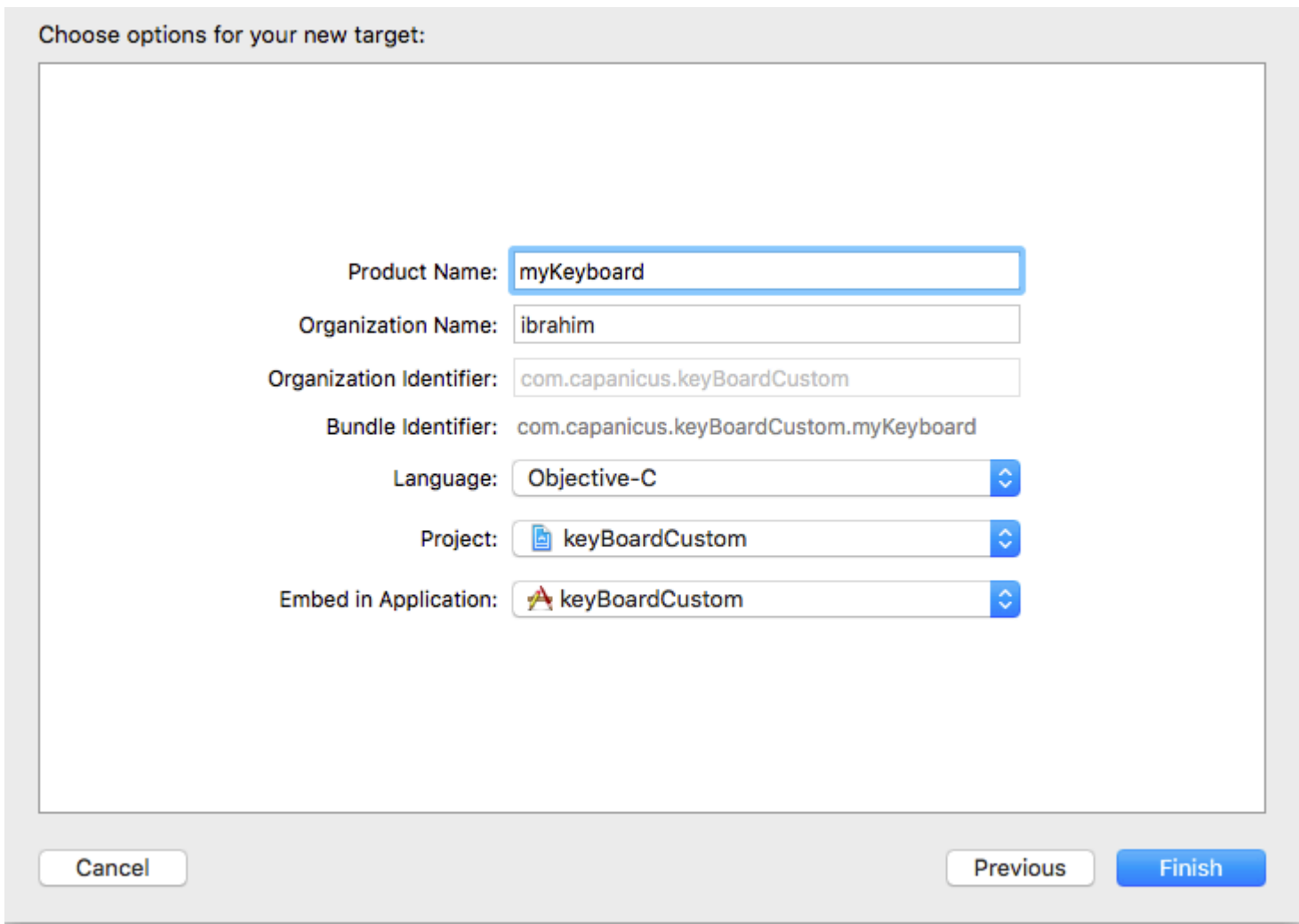
Fügen Sie einem vorhandenen XCode-Projekt ein Ziel hinzu



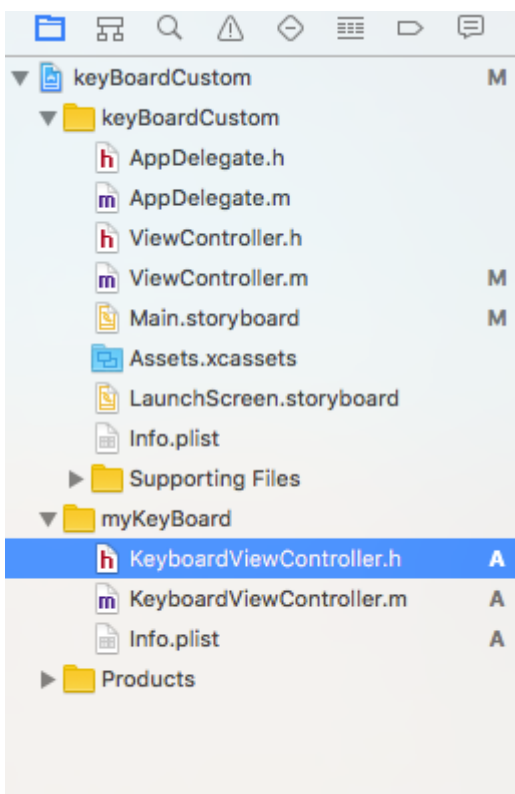
Wählen Sie im Ziel hinzufügen die Option Benutzerdefiniertes KeyBoard aus



Fügen Sie das Ziel wie folgt hinzu:

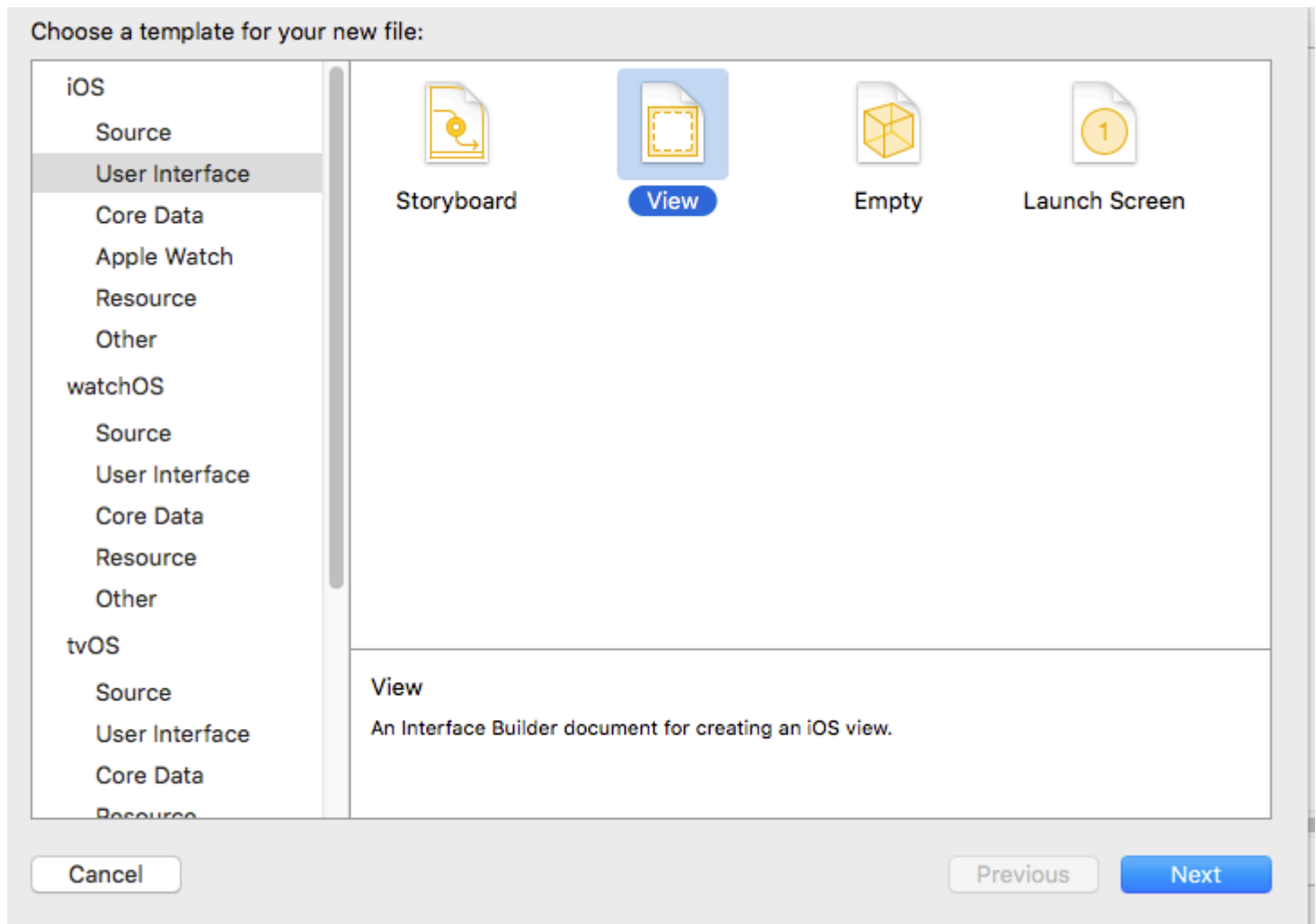


Ihr Projektdateiverzeichnis sollte ungefähr so aussehen

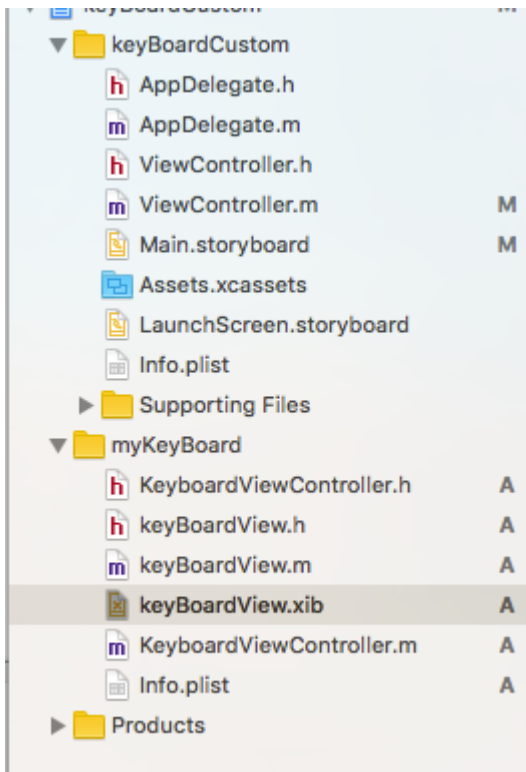


Hier ist myKeyBoard der Name des hinzugefügten Ziels

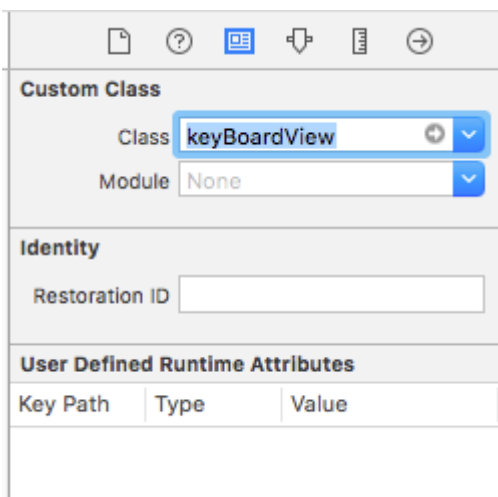
Fügen Sie eine neue Cocoatouch-Datei vom Typ UIView hinzu und fügen Sie eine Schnittstellendatei hinzu



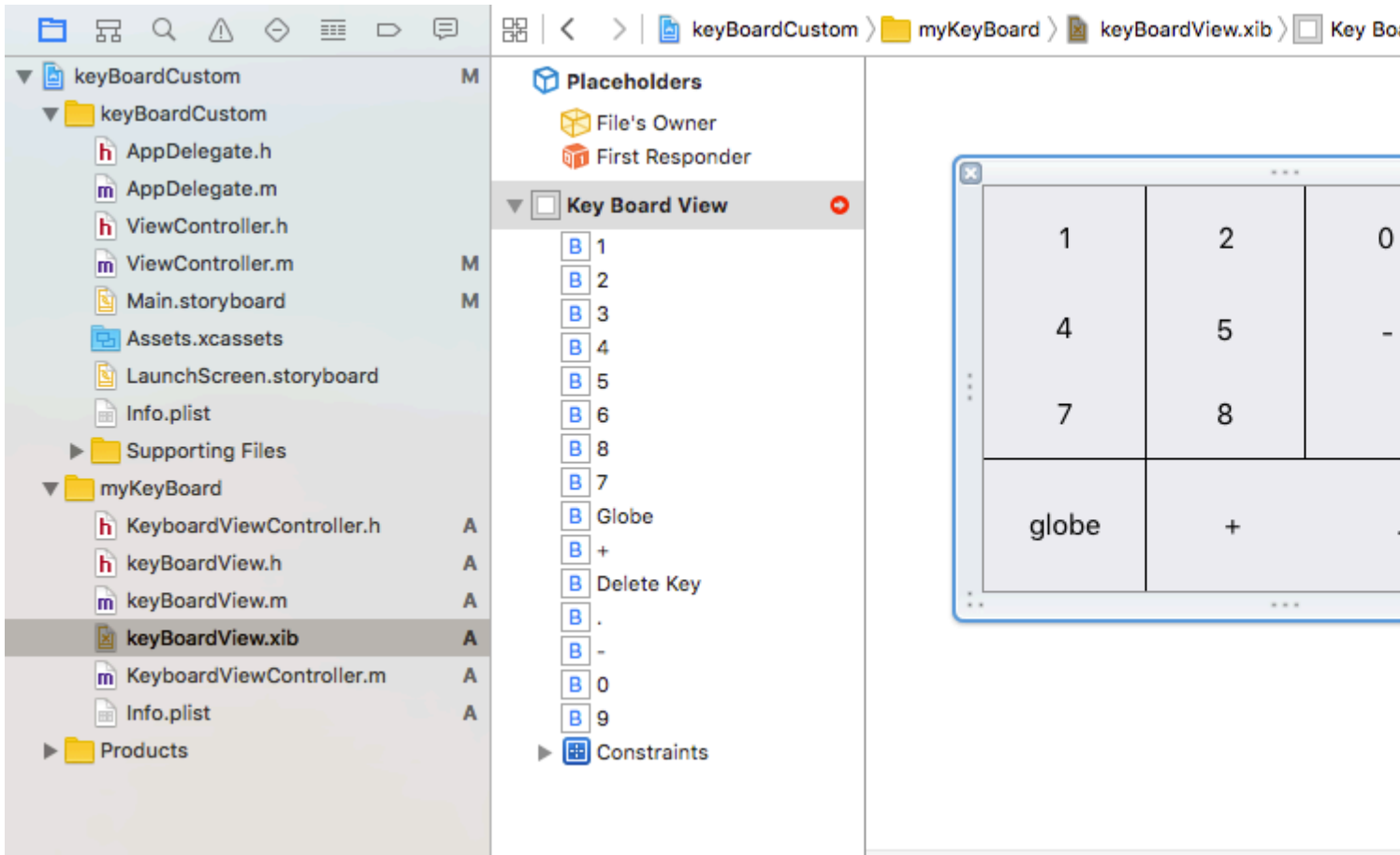
Schließlich sollte Ihr Projektverzeichnis so aussehen



Machen Sie die `keyBoardView.xib` einer Unterklasse von `keyBoardView`



`keyBoardView.xib` Sie die Schnittstelle in der Datei `keyBoardView.xib`



keyBoardView.xib Sie Verbindungen von der Datei keyBoardView.xib zur Datei keyBoardView.h

keyBoardView.h sollte so aussehen

```
#import <UIKit/UIKit.h>

@interface keyBoardView : UIView

@property (weak, nonatomic) IBOutlet UIButton *deleteKey;
//IBOutlet for the delete Key
@property (weak, nonatomic) IBOutlet UIButton *globe;
//Outlet for the key with title globe which changes the keyboard type
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *keys;
//Contains a collection of all the keys '0 to 9' '+' '-' and '.'

@end
```

In der keyBoardViewController.h Datei importieren #import "keyBoardView.h"

Deklarieren Sie eine Eigenschaft für keyboard @property (strong, nonatomic)keyBoardView *keyboard;

Kommentieren Sie das aus

```
@property (nonatomic, strong) UIButton *nextKeyboardButton and all the code associated with it
```

Die viewDidLoad () - Funktion der Datei KeyboardViewController.m sollte so aussehen

```

- (void) viewDidLoad {
    [super viewDidLoad];
    self.keyboard=[[NSBundle mainBundle]loadNibNamed:@"keyBoardView" owner:nil
options:nil]objectAtIndex:0];
    self.inputView=self.keyboard;
    [self addGestureToKeyboard];

    // Perform custom UI setup here
    // self.nextKeyboardButton = [UIButton buttonWithType:UIButtonTypeSystem];
    //
    // [self.nextKeyboardButton setTitle:NSLocalizedString(@"Next Keyboard", @"Title for 'Next
Keyboard' button") forState:UIControlStateNormal];
    // [self.nextKeyboardButton sizeToFit];
    // self.nextKeyboardButton.translatesAutoresizingMaskIntoConstraints = NO;
    //
    // [self.nextKeyboardButton addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];
    //
    // [self.view addSubview:self.nextKeyboardButton];
    //
    // [self.nextKeyboardButton.leftAnchor constraintEqualToAnchor:self.view.leftAnchor].active
= YES;
    // [self.nextKeyboardButton.bottomAnchor
constraintEqualToAnchor:self.view.bottomAnchor].active = YES;
}

```

Die Funktionen `addGestureToKeyboard` , `pressDeleteKey` , `keyPressed` sind unten definiert

```

-(void) addGestureToKeyboard
{
    [self.keyboard.deleteKey addTarget:self action:@selector(pressDeleteKey)
forControlEvents:UIControlEventTouchUpInside];
    [self.keyboard.globe addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];

    for (UIButton *key in self.keyboard.keys)
    {
        [key addTarget:self action:@selector(keyPressed:)
forControlEvents:UIControlEventTouchUpInside];
    }
}

-(void) pressDeleteKey
{
    [self.textDocumentProxy deleteBackward];
}

-(void) keyPressed:(UIButton *)key
{
    [self.textDocumentProxy insertText:[key currentTitle]];
}

```

Führen Sie die Hauptanwendung aus und gehen Sie zu Einstellungen-> Allgemein-> Tastatur-> Neue Tastatur hinzufügen-> und fügen Sie über den Tastaturabschnitt des Drittanbieters Tastatur hinzu.

Der Tastaturname kann durch Hinzufügen eines Schlüssels mit dem Namen " `Bundle display name`

geändert werden. `Bundle display name` in der Wertzeichenfolge den gewünschten Namen für die Tastatur des Hauptprojekts ein.

The screenshot shows the Xcode interface. On the left, the project browser displays the file structure for 'keyBoardCustom'. The 'Info.plist' file is selected. On the right, the 'Information Property List' is expanded, showing a list of keys and their values.

key	type	value
Localization native development re...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle display name	String	keyBoardMi
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone enviro...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)

Sie können dieses [Youtube-Video](#) auch ansehen

Benutzerdefinierte Tastatur online lesen:

<https://riptutorial.com/de/ios/topic/7358/benutzerdefinierte-tastatur>

Kapitel 27: Benutzerdefinierte UIViews aus XIB-Dateien

Bemerkungen

Von Apple: Erstellen einer benutzerdefinierten Ansicht, die im Interface Builder dargestellt wird

- Hinweis: Wenn Sie in Ihren XIB-Elementen (wie UILabel, UITextField usw.) ausgefallene "benutzerdefinierte" Schriftarten verwenden, ist die anfängliche Ladezeit Ihrer XIB je nach gewählter Schriftart und Systemversion länger.

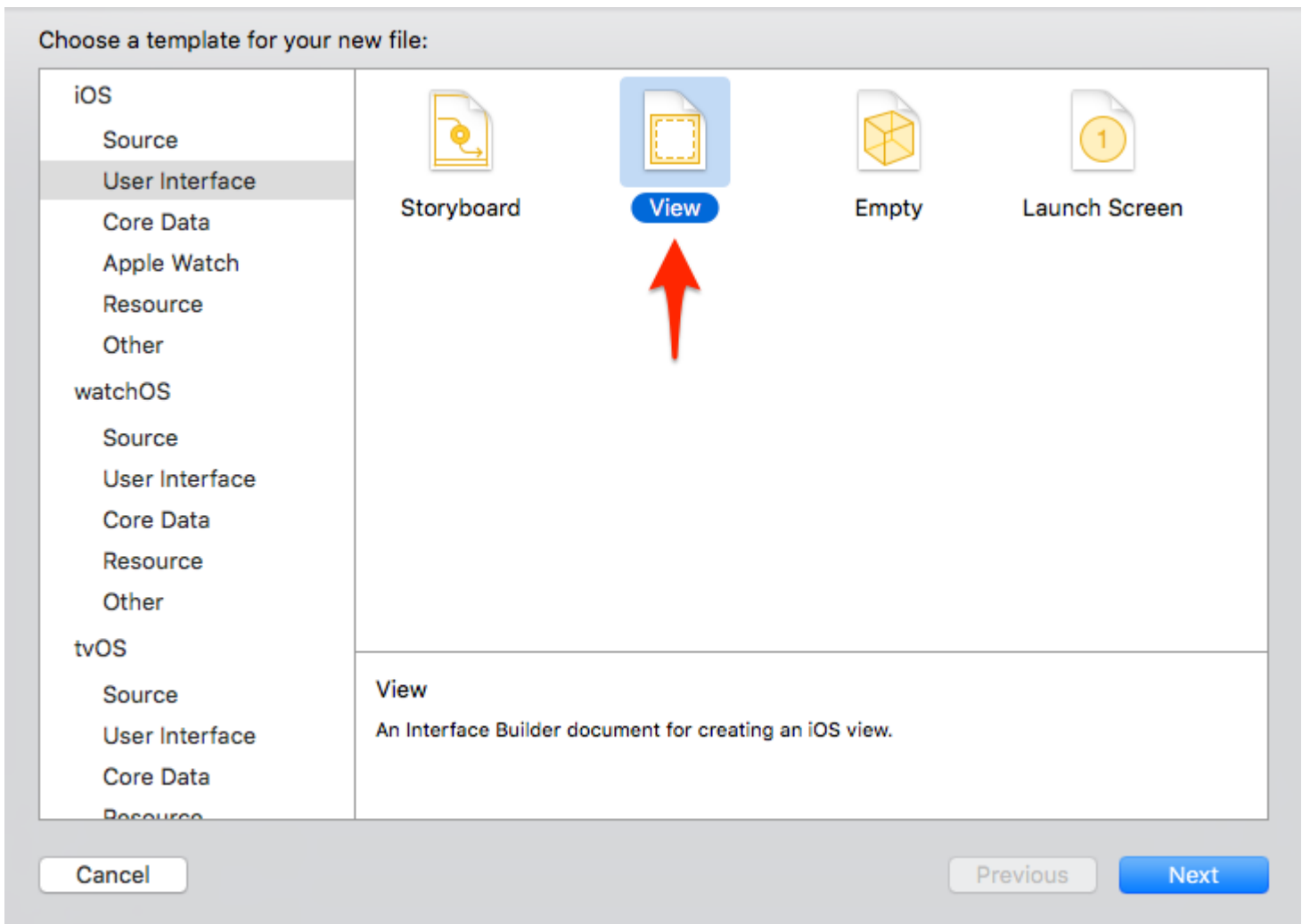
Examples

Verdrahtungselemente

Erstellen Sie eine XIB-Datei

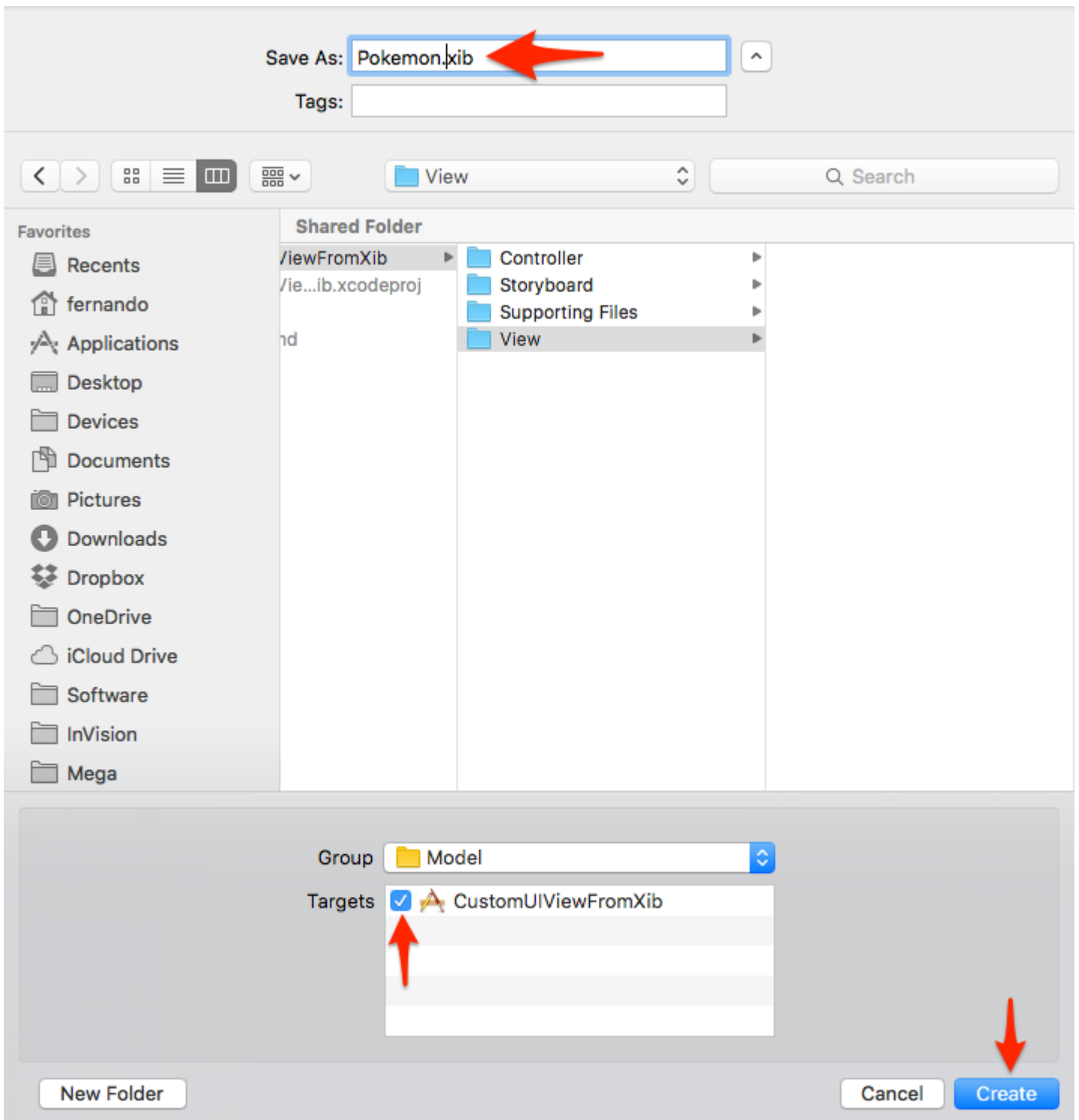
Xcode-Menüleiste > Datei > Neu > Datei.

Wählen Sie iOS, Benutzeroberfläche und dann "Ansicht":



Gib deiner XIB einen Namen (ja, wir machen ein Pokemon-Beispiel).

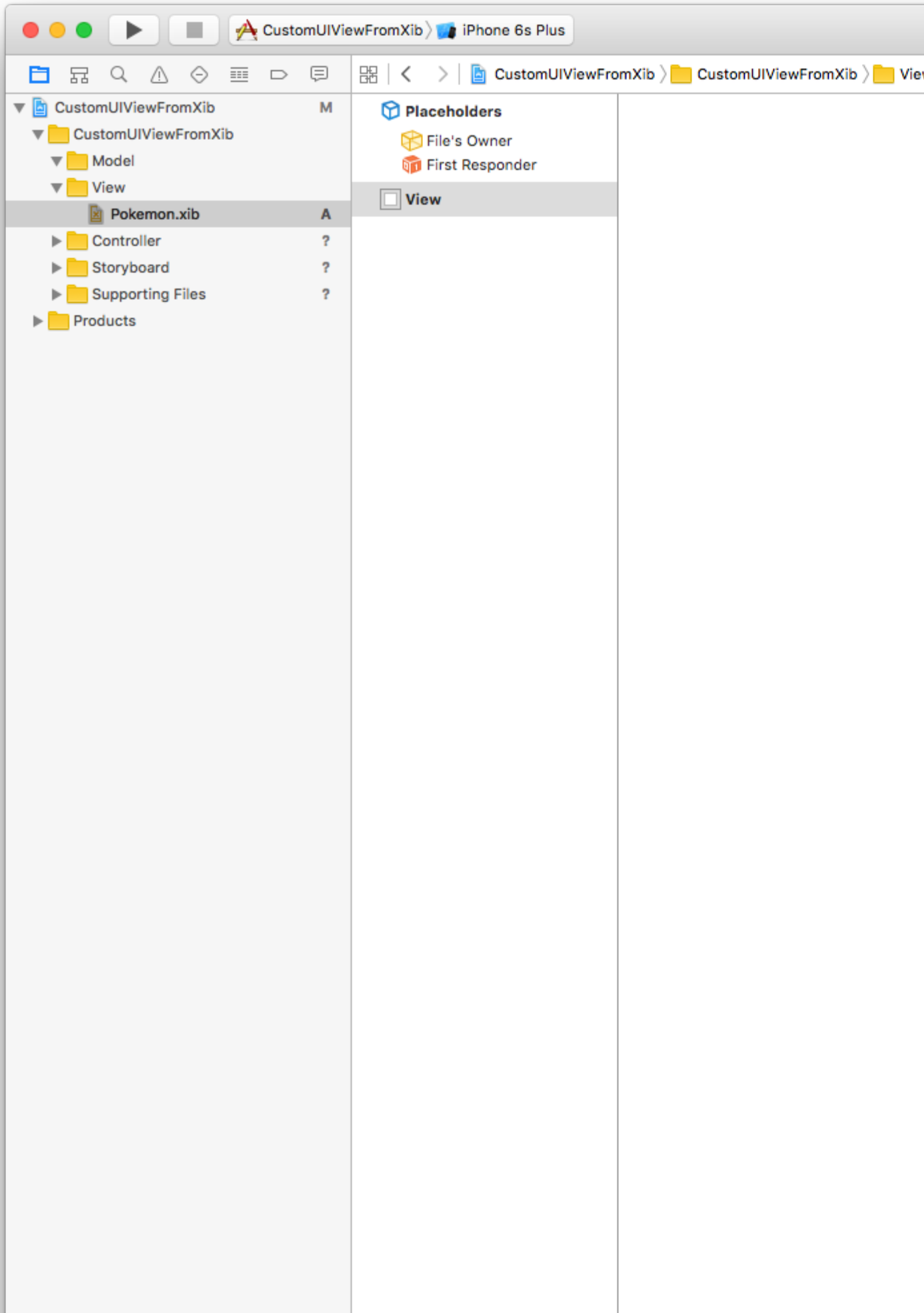
Denken Sie daran, Ihr Ziel zu überprüfen und klicken Sie auf "Erstellen".



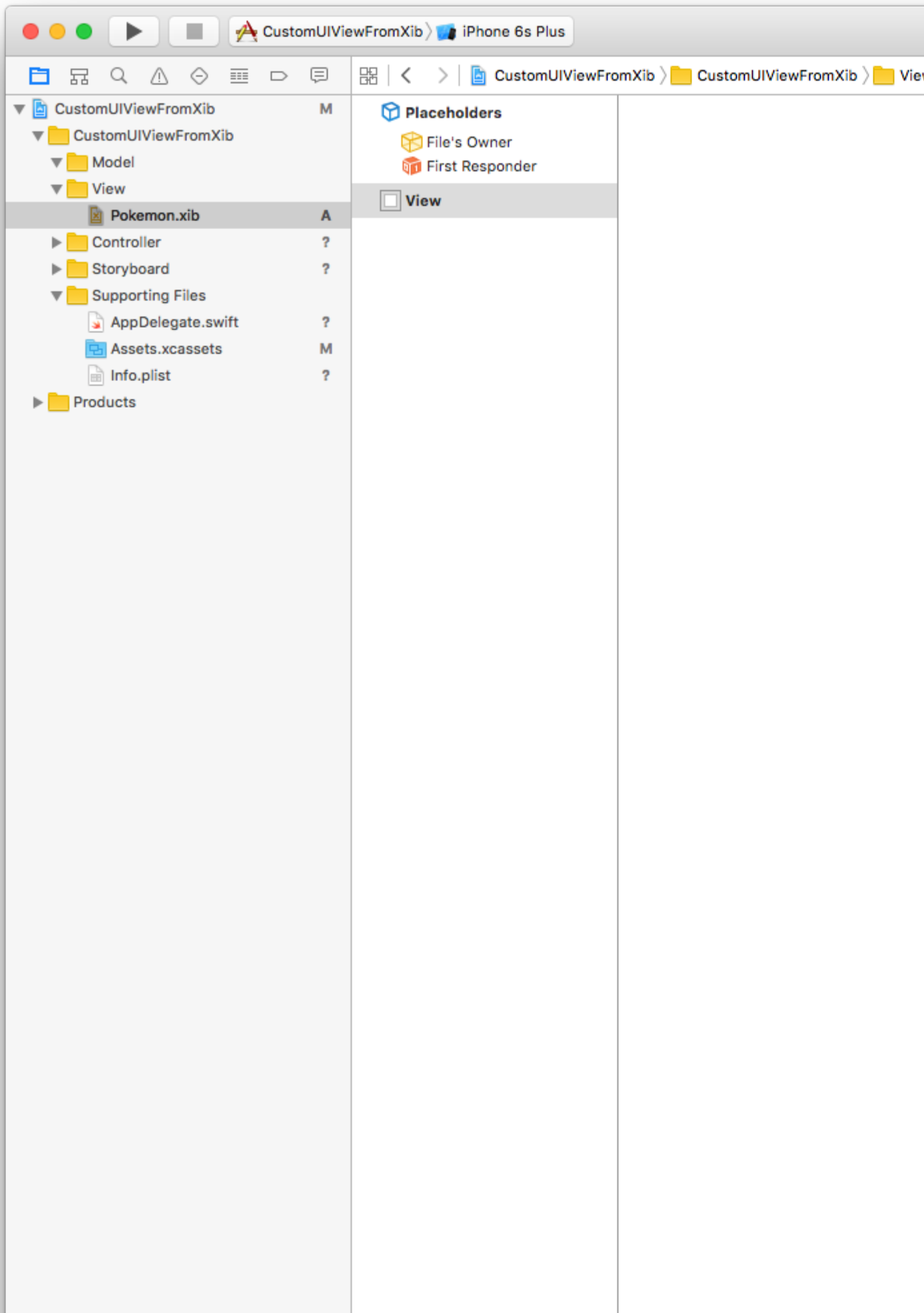
Gestalten Sie Ihre Sicht

Zur Vereinfachung stellen Sie Folgendes ein:

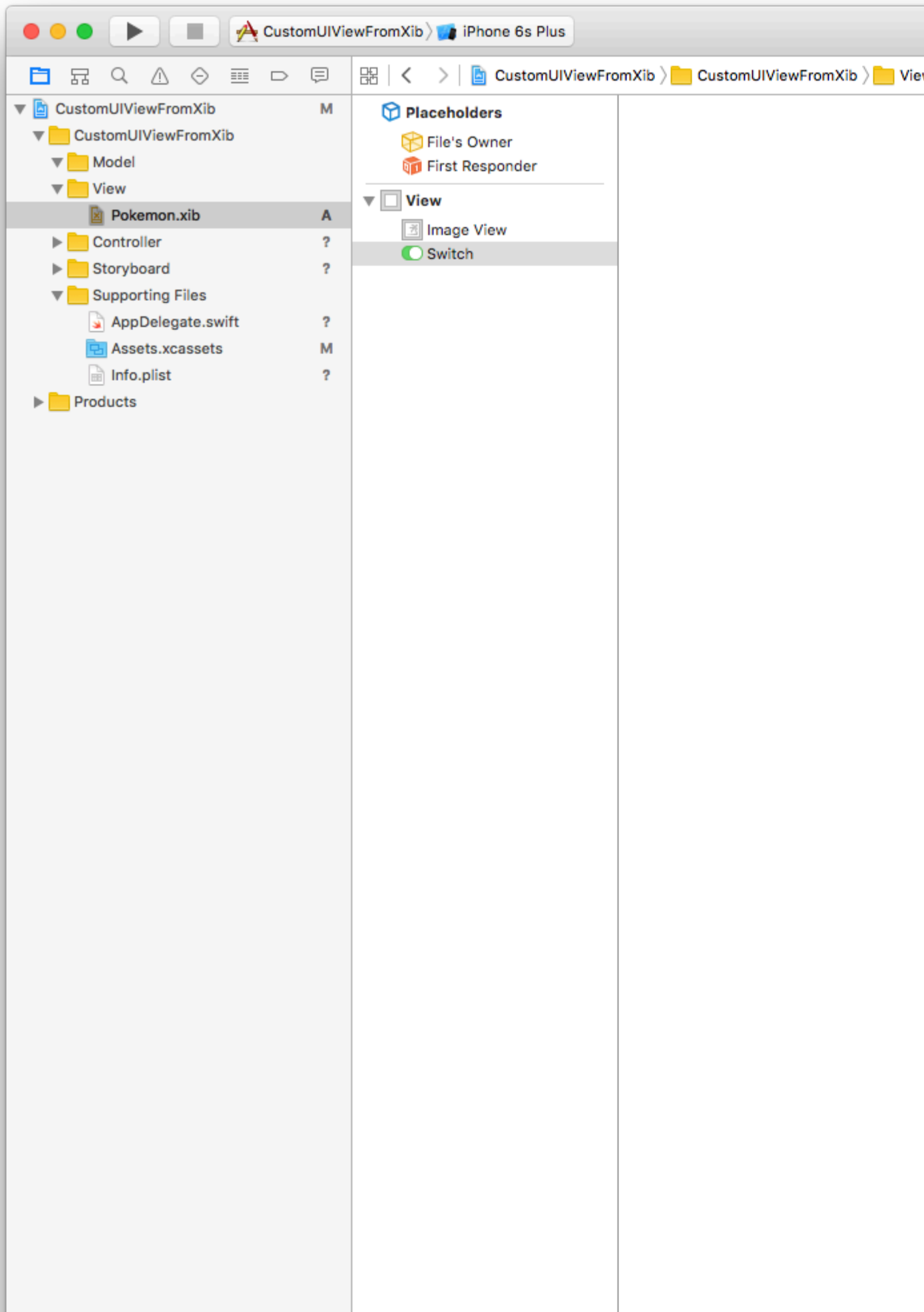
- Größe: Freeform
- Statusleiste: Keine
- Top Bar: Keine
- Untere Leiste: Keine



In diesem Beispiel werden Breite 321 und Höhe 256 verwendet.

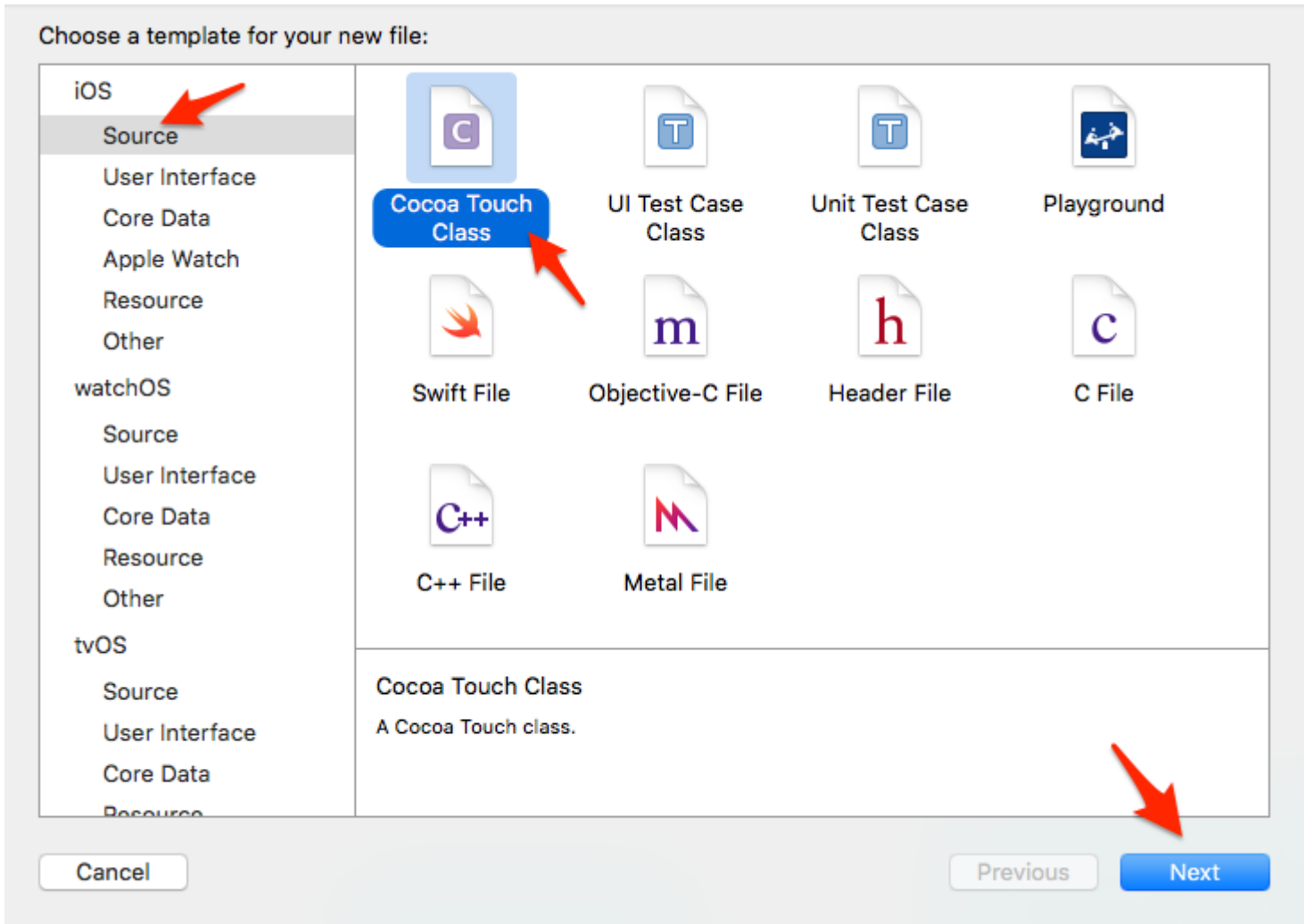


Hier finden wir ein **Bild Ansicht** (256x256) und einen **Schalter** hinzufügen.



Xcode-Menüleiste > Datei > Neu > Datei.

Wählen Sie iOS / Source / Cocoa Touch Class aus. Klicken Sie auf "Weiter".



Geben Sie der Klasse einen Namen, der mit der XIB-Datei (Pokemon) identisch sein muss. Wählen Sie UIView als Unterklassentyp aus und klicken Sie auf "Weiter".

Choose options for your new file:

Class:

Subclass of:

Also create XIB file

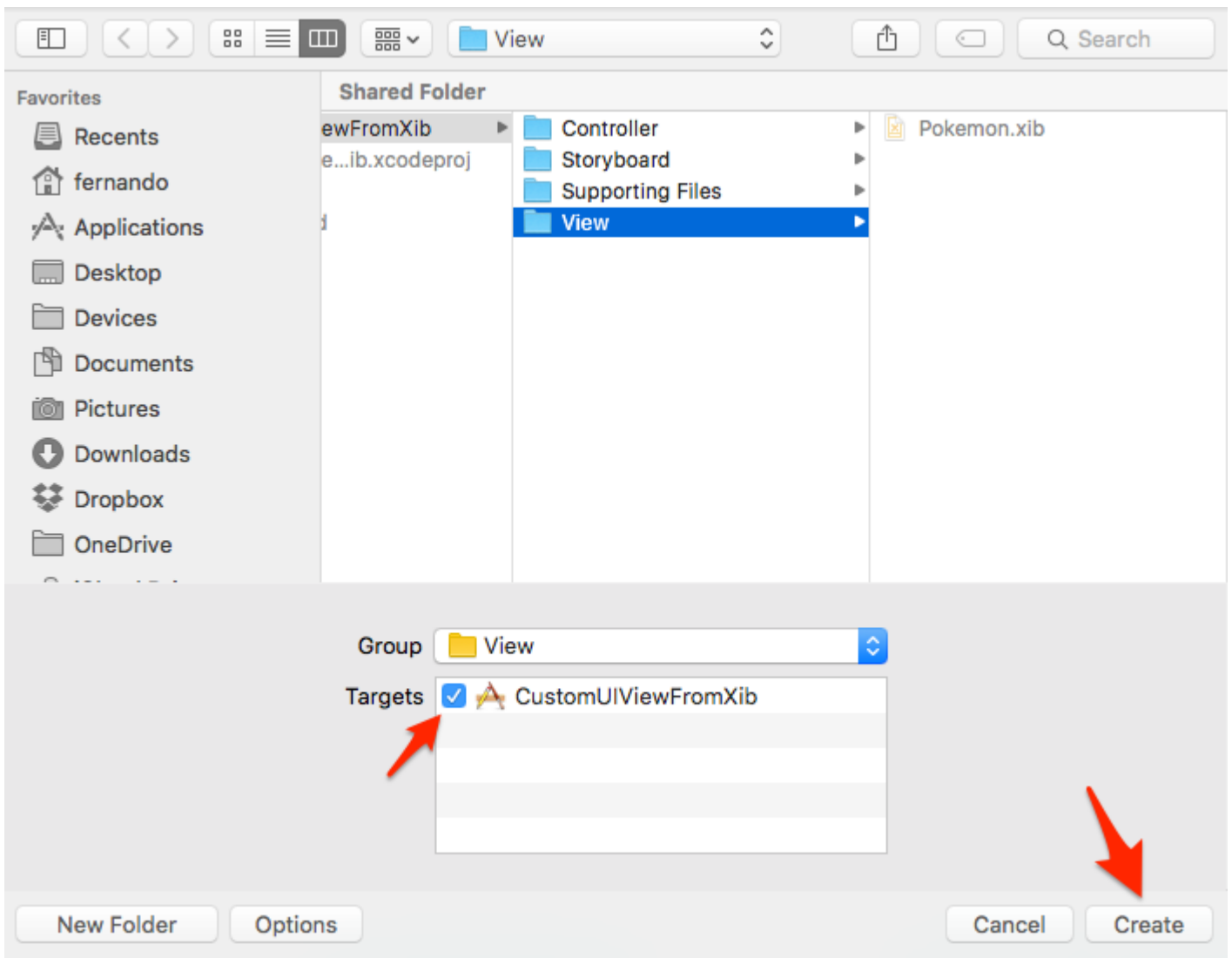
Language:

Cancel

Previous

Next

Wählen Sie im nächsten Fenster Ihr Ziel aus und klicken Sie auf "Erstellen".

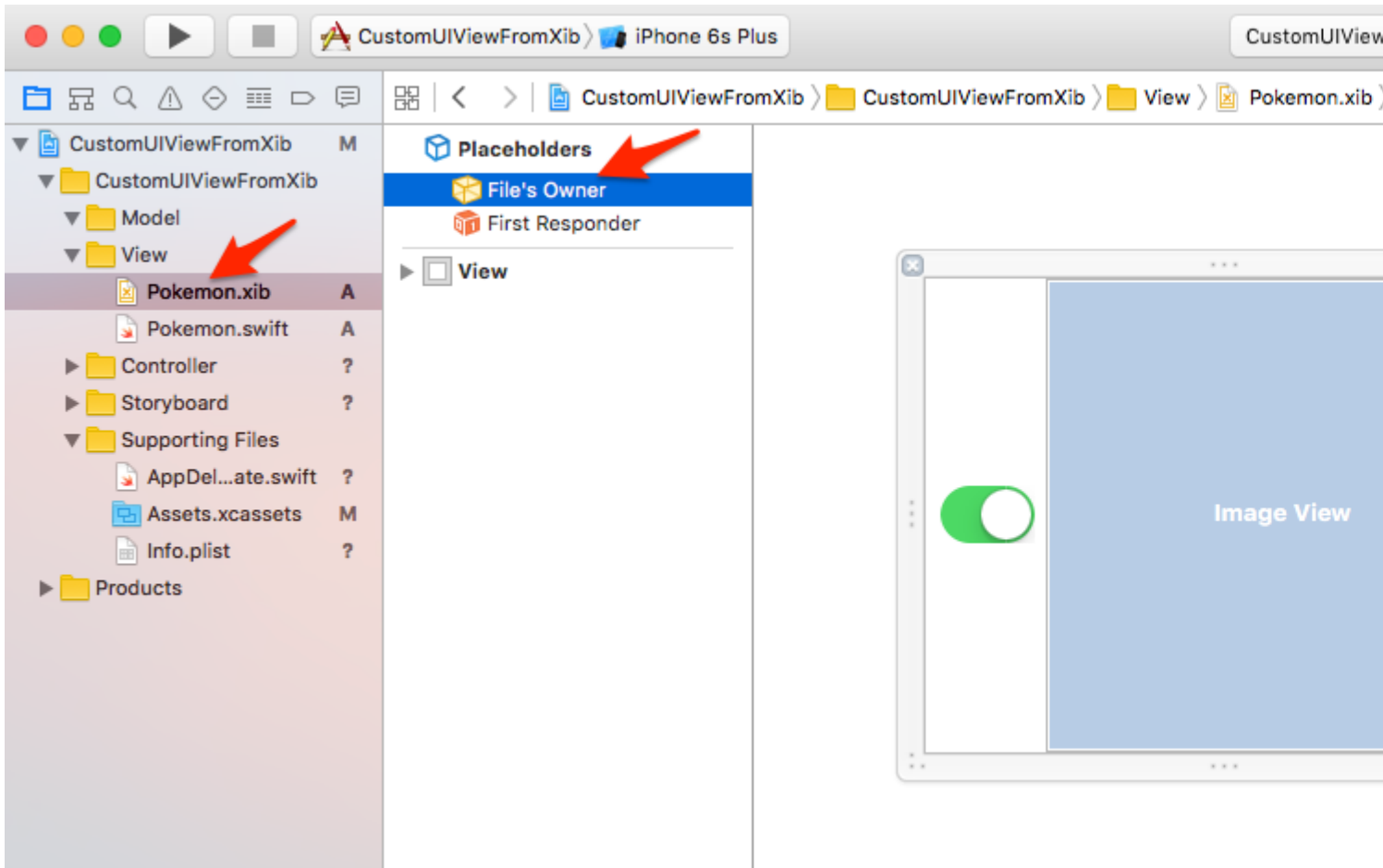


Verbinden Sie Pokemon.xib über das Attribut "Dateieigner" mit Pokemon.swift

Klicken Sie auf die Pokemon.xib-Datei in Xcode.

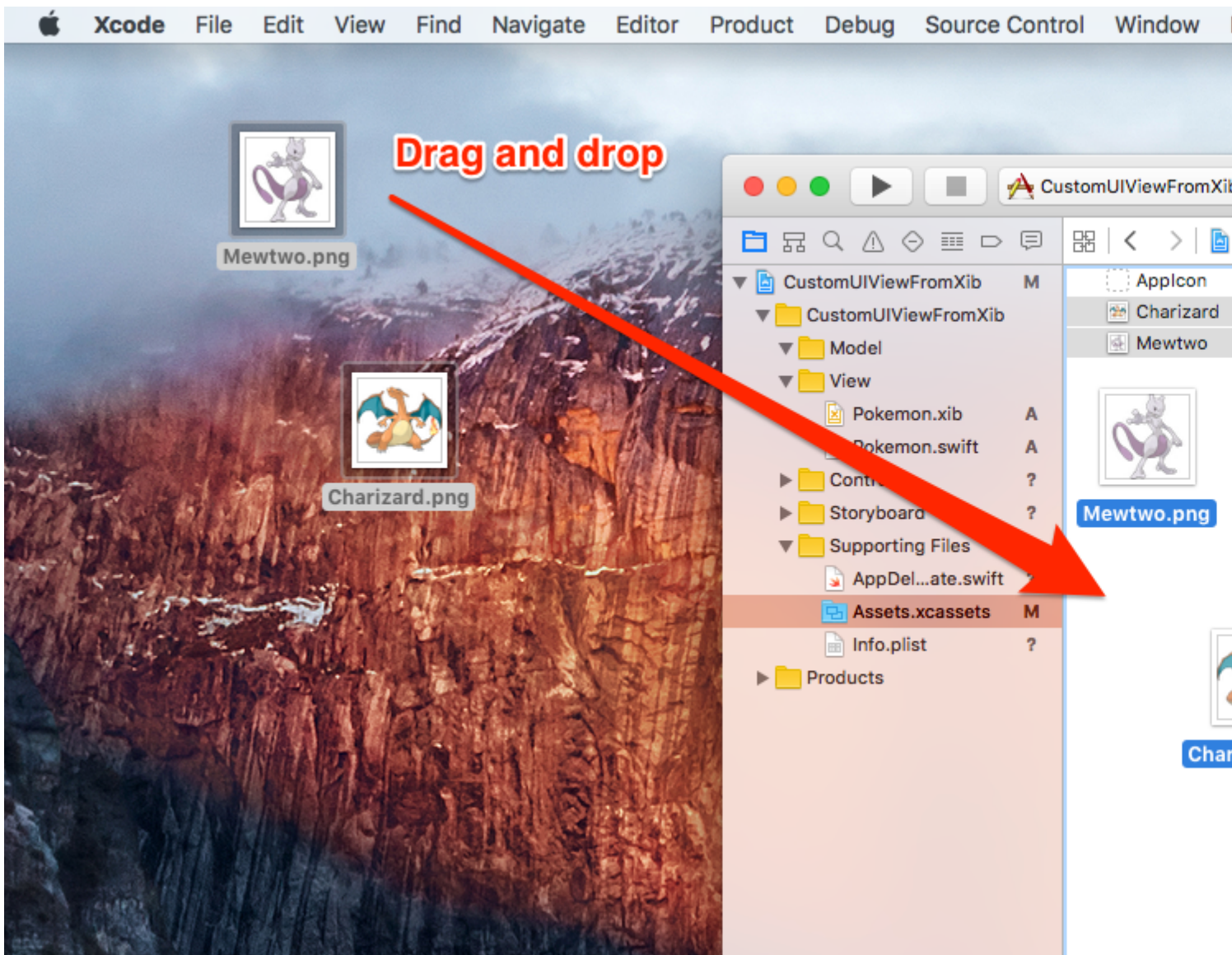
Klicken Sie auf den Ausgang "Dateieigner".

Legen Sie im "Identitätsinspektor" (oben rechts) die Klasse auf unsere kürzlich erstellte Datei "Pokemon.swift" fest.



POKEMONS !!!

Ja! Ziehen Sie einige Pokemons in Ihr Projekt, um unsere "Infrastruktur" zu beenden.
Hier fügen wir zwei PGN-Dateien, 256x256, transparent hinzu.



Zeigen Sie mir schon den Code.

Gut gut.

Zeit, um unserer Pokemon.swift-Klasse etwas Code hinzuzufügen.

Es ist eigentlich ziemlich einfach:

1. Implementieren Sie die erforderlichen Initialisierer
2. Laden Sie die XIB-Datei
3. Konfigurieren Sie die Ansicht, in der die XIB-Datei angezeigt wird
4. Zeigen Sie die obige Ansicht an

Fügen Sie der Pokemon.swift-Klasse den folgenden Code hinzu:

```
import UIKit

class Pokemon: UIView {

    // MARK: - Initializers

    override init(frame: CGRect) {
```



```

    super.init(frame: frame)
    setupView()
}

required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    setupView()
}

// MARK: - Private Helper Methods

// Performs the initial setup.
private func setupView() {
    let view = viewFromNibForClass()
    view.frame = bounds

    // Auto-layout stuff.
    view.autoresizingMask = [
        UIViewAutoresizing.flexibleWidth,
        UIViewAutoresizing.flexibleHeight
    ]

    // Show the view.
    addSubview(view)
}

// Loads a XIB file into a view and returns this view.
private func viewFromNibForClass() -> UIView {

    let bundle = Bundle(for: type(of: self))
    let nib = UINib(nibName: String(describing: type(of: self)), bundle: bundle)
    let view = nib.instantiate(withOwner: self, options: nil).first as! UIView

    /* Usage for swift < 3.x
    let bundle = NSBundle(forClass: self.dynamicType)
    let nib = UINib(nibName: String(self.dynamicType), bundle: bundle)
    let view = nib.instantiateWithOwner(self, options: nil)[0] as! UIView
    */

    return view
}
}

```

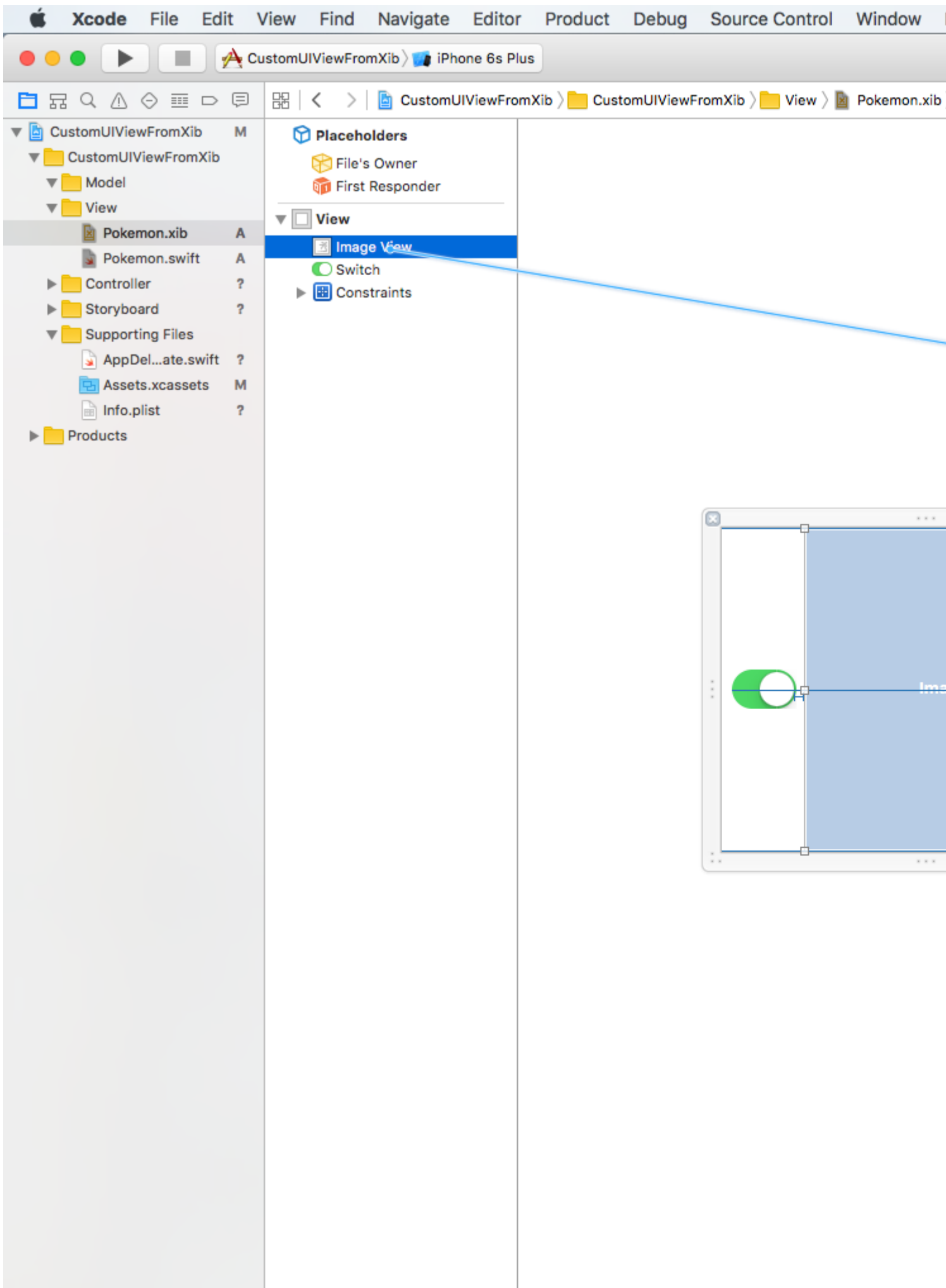
@IBDesignable und @IBInspectable

Durch das Hinzufügen von `@IBDesignable` zu Ihrer Klasse ermöglichen Sie das Live-Rendering im Interface Builder.

Durch Hinzufügen von `@IBInspectable` zu den Eigenschaften Ihrer Klasse können Sie sehen, dass sich Ihre benutzerdefinierten Ansichten im Interface Builder ändern, sobald Sie diese Eigenschaften ändern.

Lassen Sie uns die `Image View` unserer benutzerdefinierten Ansicht "Inspectable" machen.

Zuerst haken Sie die oben `Image View` von der `Pokemon.xib` Datei an die `Pokemon.swift` Klasse.



@IBDesignable vor dem Klassennamen):

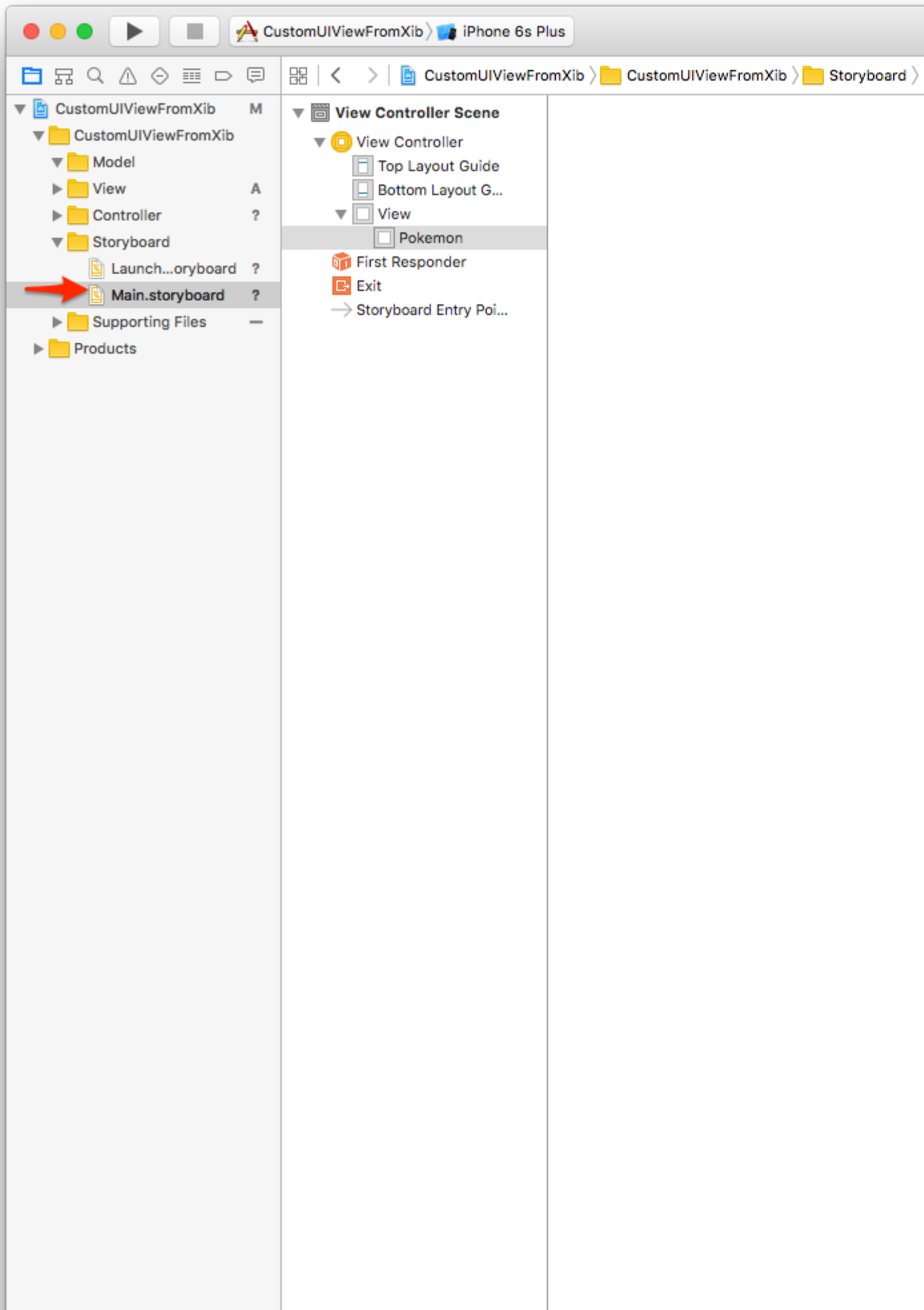
```
@IBDesignable class Pokemon: UIView {  
  
    // MARK: - Properties  
  
    @IBOutlet weak var imageView: UIImageView!  
  
    @IBInspectable var image: UIImage? {  
        get {  
            return imageView.image  
        }  
        set(image) {  
            imageView.image = image  
        }  
    }  
  
    // MARK: - Initializers  
    ...  
}
```

Eigene Ansichten verwenden

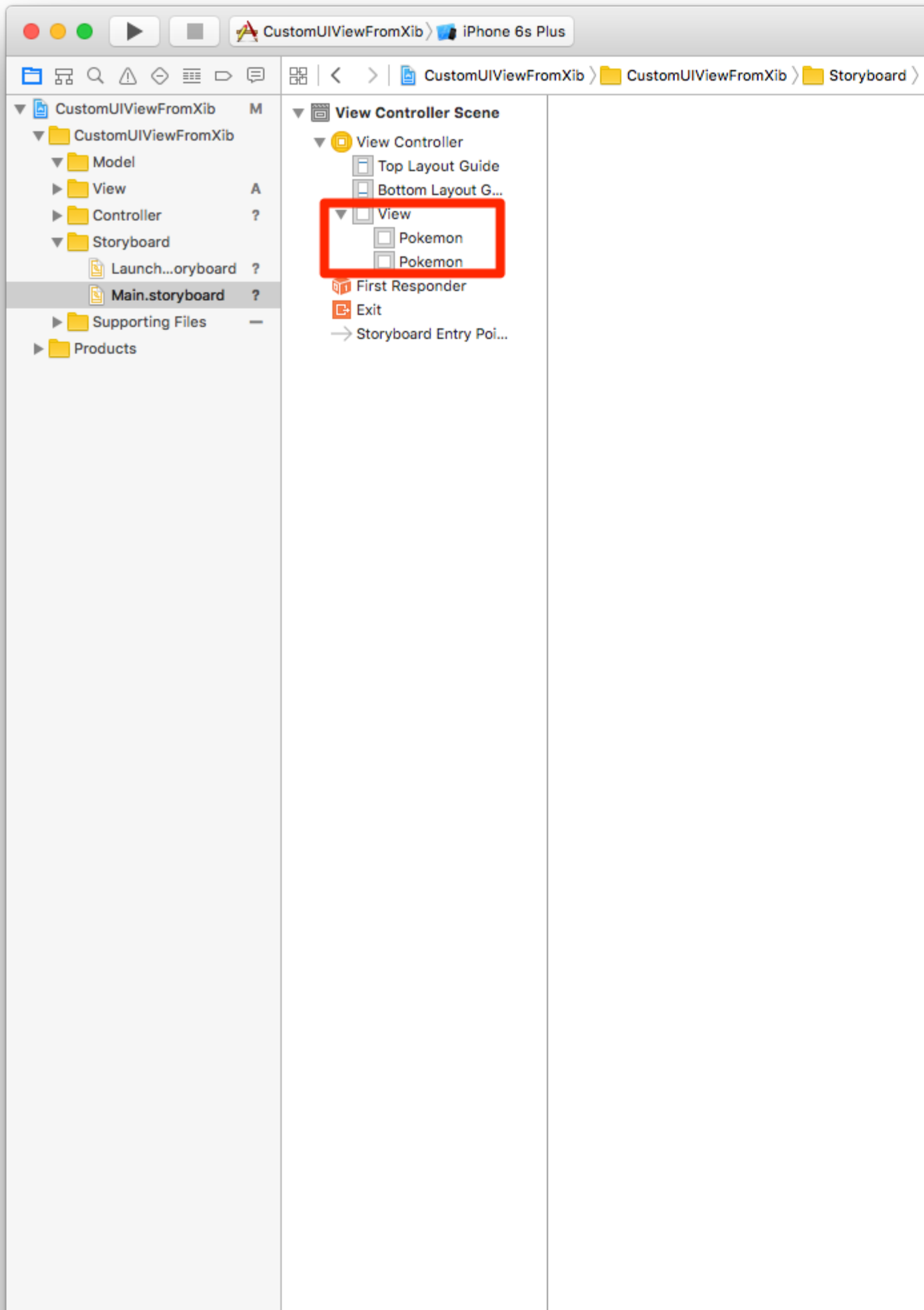
Gehen Sie zu Ihrer Haupt-Storyboard-Datei und ziehen Sie eine UIView hinein.

Ändern Sie die Größe der Ansicht auf 200x200. Zentrieren.

Gehen Sie zum Identitätsinspektor (oben rechts) und setzen Sie die Klasse auf Pokemon.



Geben Sie eine andere Größe an, sagen Sie 150x150.
Wähle ein anderes Pokemon-Bild aus und beachte:



Mit der Schaltfläche können Pokemons aktiviert / deaktiviert werden.

Erstellen Sie eine `IBAction` von der Schaltfläche Switch auf die `Pokemon.swift`-Klasse.

Rufen Sie die Aktion wie `switchTapped`.


Fügen Sie den folgenden Code hinzu:

```
// MARK: - Actions

@IBAction func switchTapped(sender: UISwitch) {
    imageView.alpha = sender.on ? 1.0 : 0.2
}

// MARK: - Initializers
...
```

Endergebnis:

Carrier 

3:54 PM



Game Center



Extras



Watch



CustomUIV...



Jetzt können Sie komplexe benutzerdefinierte Ansichten erstellen und sie an beliebiger Stelle wiederverwenden.

Dies erhöht die Produktivität und isoliert den Code in eigenständige Elemente der Benutzeroberfläche.

[Das endgültige Projekt kann in Github geklont werden.](#)

(**Aktualisiert zu Swift 3.1**)

Wie kann UIView mit XIB benutzerdefiniert wiederverwendbar werden?

Das folgende Beispiel zeigt die Schritte zum Initialisieren einer Ansicht aus XIB.

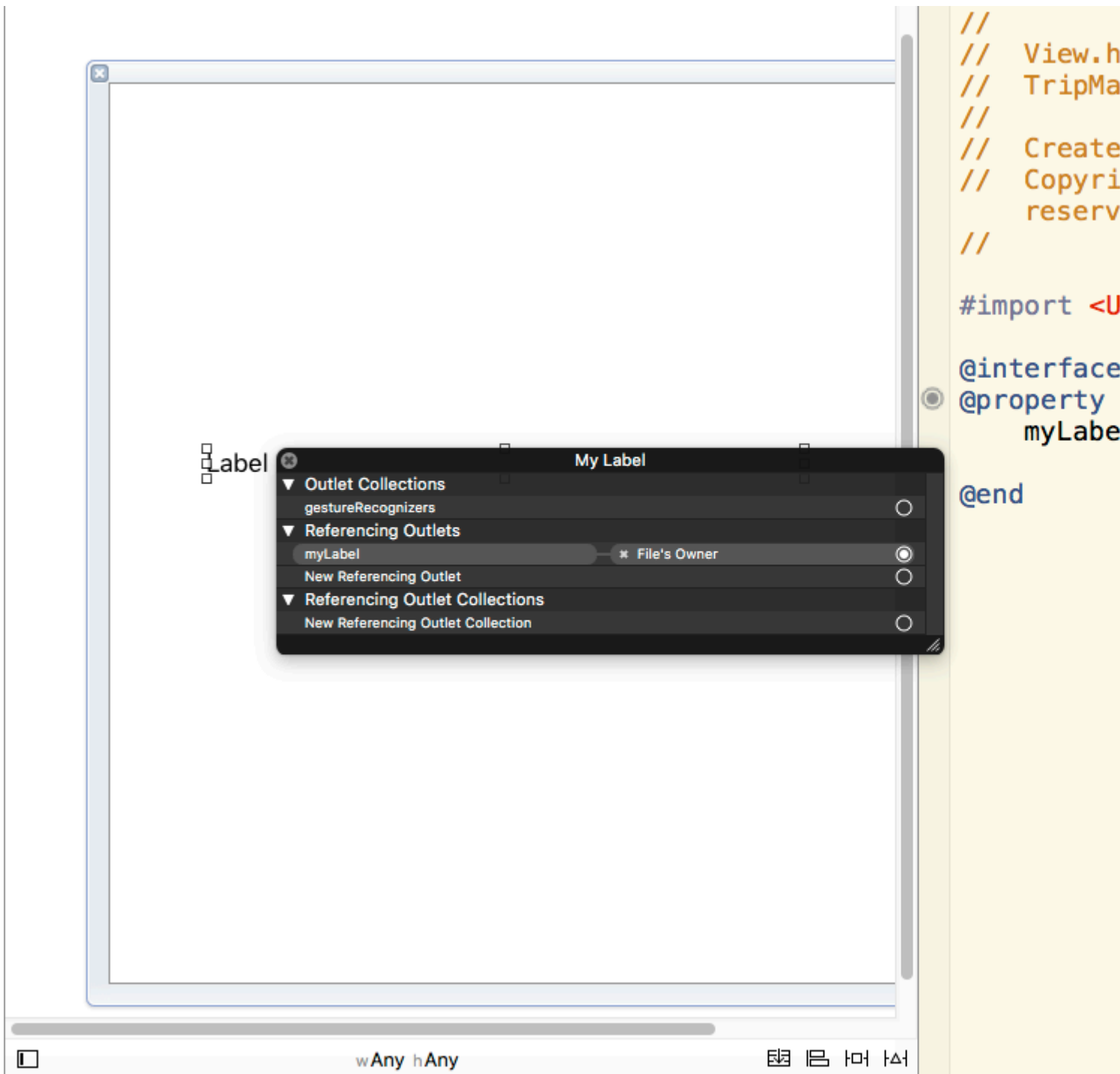
Dies ist keine komplexe Operation, aber es müssen genaue Schritte befolgt werden, um es beim ersten Mal richtig zu machen, wobei Ausnahmen vermieden werden.

[Wie funktioniert loadNibNamed?](#)

Hauptschritte sind:

1. XIB erstellen
2. Erstellen Sie die Klasse .h und .m
3. Definieren Sie die Auslässe in .h
4. Verbinden Sie die Ausgänge zwischen .h und XIB

Siehe angehängter Screenshot:



5. Rufen Sie `loadNibNamed` in der `initWithCoder`-Funktion der `.m`-Datei auf. Dies ist erforderlich, um sicherzustellen, dass Sie das `UIView`-Objekt direkt in die Storyboard- / übergeordnete `UIView`-XIB-Datei einfügen und als benutzerdefinierte Ansicht definieren können. Nach dem Laden des Storyboards / der übergeordneten XIB wird kein anderer Initialisierungscode benötigt. Ihre benutzerdefinierte Ansicht kann wie andere integrierte Objective C-Ansichtsobjekte, die in XCode angegeben sind, zu anderen Ansichten hinzugefügt werden.

Benutzerdefinierte UIViews aus XIB-Dateien online lesen:

<https://riptutorial.com/de/ios/topic/1362/benutzerdefinierte-uiviews-aus-xib-dateien>

Kapitel 28: Benutzerdefiniertes UITextField

Einführung

Mit benutzerdefiniertem UITextField können wir das Verhalten von Textfeldern beeinflussen!

Examples

Benutzerdefiniertes UITextField zum Filtern von Eingabetext

Hier ist ein Beispiel eines benutzerdefinierten UITextField, das nur numerischen Text übernimmt und alle anderen verwirft.

HINWEIS: Für das iPhone ist dies mit der Zifferntastatur problemlos möglich, für das iPad gibt es jedoch keine Tastatur nur mit Zahlen

```
class NumberTextField: UITextField {

    required init(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        registerForTextFieldNotifications()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
    }

    override func awakeFromNib() {
        super.awakeFromNib()
        keyboardType = .numberPad//useful for iPhone only
    }

    private func registerForTextFieldNotifications() {
        NotificationCenter.default.addObserver(self, selector:
        #selector(NumberTextField.textDidChange), name: NSNotification.Name(rawValue:
        "UITextFieldTextDidChangeNotification"), object: self)
    }

    deinit {
        NotificationCenter.default.removeObserver(self)
    }

    func textDidChange() {
        text = filteredText()
    }

    private func filteredText() -> String {
        let inverseSet = CharacterSet(charactersIn:"0123456789").inverted
        let components = text!.components(separatedBy: inverseSet)
        return components.joined(separator: "")
    }
}
```

Wo immer wir Textfelder wünschen, die nur Zahlen als Eingabetext verwenden, können wir dieses

benutzerdefinierte UITextField verwenden

Benutzerdefiniertes UITextField zum Deaktivieren aller Aktionen wie Kopieren, Einfügen usw

Wenn Sie alle Aktionen wie Kopieren, Einfügen, Ersetzen, Auswählen usw. von `UITextField` deaktivieren möchten, können wir das folgende benutzerdefinierte Textfeld verwenden:

```
class CustomTextField: UITextField {  
  
    var enableLongPressActions = false  
  
    required init(coder aDecoder: NSCoder) {  
        super.init(coder: aDecoder)!  
    }  
  
    override init(frame: CGRect) {  
        super.init(frame: frame)  
    }  
  
    override func canPerformAction(_ action: Selector, withSender sender: Any?) -> Bool {  
        return enableLongPressActions  
    }  
}
```

Mit der Eigenschaft `enableLongPressActions` können wir später bei Bedarf alle Aktionen aktivieren.

Benutzerdefiniertes UITextField online lesen:

<https://riptutorial.com/de/ios/topic/9997/benutzerdefiniertes-uitextfield>

Kapitel 29: Bilder asynchron laden

Examples

Einfachster Weg

Am einfachsten erstellen Sie dies, indem Sie [Alamofire](#) und seine [UIImageViewExtension](#) verwenden. Was wir brauchen, ist eine `UITableView` mit einer Zelle, in der sich `imageView` befindet, und diese `imageView` nennen kann.

In der Funktion `cellForRowAt:` des `tableView` würden wir das Bild herunterladen und folgendermaßen einstellen:

```
let url = URL(string: "https://httpbin.org/image/png")!
let placeholderImage = UIImage(named: "placeholder")!

imageView.af_setImage(withURL: url, placeholderImage: placeholderImage)
```

Die URL sollte auf das Bild zeigen, das Sie herunterladen möchten, und das `placeholder`-Bild sollte ein gespeichertes Bild sein. Wir rufen dann die `af_setImage` Methode in `imageView` die das Bild unter der angegebenen URL herunterlädt, und während des Downloads wird das Platzhalterbild angezeigt. Sobald das Bild heruntergeladen ist, wird das angeforderte Bild angezeigt

Überprüfen Sie, ob die Zelle nach dem Download noch sichtbar ist

Manchmal dauert der Download länger als die Zelle angezeigt wird. In diesem Fall kann es vorkommen, dass das heruntergeladene Bild in der falschen Zelle angezeigt wird. Um dies zu beheben, können wir die [UIImageView-Erweiterung](#) nicht verwenden.

Wir werden immer noch [Alamofire](#) verwenden, aber wir werden den Fertigstellungs-Handler verwenden, um das Bild anzuzeigen.

In diesem Szenario benötigen wir noch eine `tableView` mit einer Zelle, die eine `imageView` enthält. In der `cellForRowAt:`-Methode würden wir das Bild mit dem folgenden Code herunterladen:

```
let placeholderImage = UIImage(named: "placeholder")!
imageView.image = placeholderImage

let url = URL(string: "https://httpbin.org/image/png")!

Alamofire.request(url!, method: .get).responseImage { response in
    guard let image = response.result.value else { return }

    if let updateCell = tableView.cellForRow(at: indexPath) {
        updateCell.imageView.image = image
    }
}
```

In diesem Beispiel setzen wir das Bild zunächst auf das Platzhalterbild. Anschließend laden wir

das Bild mit der `request` von [Alamofire herunter](#) . Wir übergeben die URL als erstes Argument und da wir nur das Image erhalten wollen, verwenden wir die HTTP-Methode `.get` . Da wir ein Bild herunterladen, möchten wir, dass die Antwort ein Bild ist. Daher verwenden wir die `.responseImage` Methode.

Nachdem das Bild heruntergeladen wurde, wird die Schließung aufgerufen. Zunächst stellen wir sicher, dass das heruntergeladene Bild tatsächlich vorhanden ist. Dann stellen wir sicher, dass die Zelle noch sichtbar ist, indem wir überprüfen, dass `cellForRow` (at: `indexPath`) nicht null zurückgibt. Wenn dies nicht der Fall ist, wird das zuletzt heruntergeladene Bild zugewiesen.

Diese letzte if-Anweisung stellt sicher, dass die Zelle noch sichtbar ist, wenn der Benutzer bereits über die Zelle gescrollt wurde. Die `updateCell`-Instanz ist null und die if-Anweisung gibt null zurück. Dies hilft uns zu verhindern, dass das falsche Bild in einer Zelle angezeigt wird.

Bilder asynchron laden online lesen: <https://riptutorial.com/de/ios/topic/10793/bilder-asynchron-laden>

Kapitel 30: Block

Syntax

- Als Variable:

```
returnType (^ blockName) (parameterTypes) = ^ returnType (parameter) {...};
```

- Als Eigenschaft:

```
@ property (nonatomic, copy) returnType (^ blockName) (parameterTypes);
```

- Als Methodenparameter:

```
- (void) methodWithBlock: (returnType (^) (parameterTypes)) blockName;
```

- Als typedef:

```
typedef returnType (^ TypeName) (parameterTypes);
```

```
Typname blockName = ^ returnType (Parameter) {...};
```

Examples

UIView Animationen

```
[UIView animateWithDuration:1.0
 animations:^(
     someView.alpha = 0;
     otherView.alpha = 1;
 )
 completion:^(BOOL finished) {
     [someView removeFromSuperview];
 }];
```

Das Carat-Zeichen "^" definiert einen Block. Zum Beispiel ist `^{ ... }` ein Block. Genauer gesagt, ist dies ein Block, der "void" zurückgibt und keine Argumente akzeptiert. Es ist äquivalent zu einer Methode wie: `-(void) etwas;`, aber dem Codeblock ist kein inhärenter Name zugeordnet.

Definieren Sie einen Block, der Argumente akzeptieren kann, die sehr ähnlich funktionieren. Um einem Block ein Argument zu geben, definieren Sie den Block folgendermaßen: `^(BOOL someArg, NSString someStr) {...} *`. Wenn Sie API-Aufrufe verwenden, die Blöcke unterstützen, schreiben Sie Blöcke, die ähnlich aussehen, insbesondere für Animationsblöcke oder `NSURLConnection`-Blöcke, wie im obigen Beispiel gezeigt.

Benutzerdefinierter Abschlussblock für benutzerdefinierte Methoden

1- Definieren Sie Ihren eigenen benutzerdefinierten Block

```
typedef void(^myCustomCompletion) (BOOL);
```

2- Erstellen Sie eine benutzerdefinierte Methode, die Ihren benutzerdefinierten Abschlussblock als Parameter verwendet.

```
-(void) customMethodName:(myCustomCompletion) compblock{  
    //do stuff  
    // check if completion block exist; if we do not check it will throw an exception  
    if(compblock)  
        compblock(YES);  
}
```

3- Verwendung des Blocks in Ihrer Methode

```
[self customMethodName:^(BOOL finished) {  
    if(finished){  
        NSLog(@"success");  
    }  
}];
```

Erfasste Variable ändern

Block erfasst Variablen, die im gleichen lexikalischen Bereich angezeigt wurden. Normalerweise werden diese Variablen als "const" -Wert erfasst:

```
int val = 10;  
void (^blk)(void) = ^{  
    val = 20; // Error! val is a constant value and cannot be modified!  
};
```

Um die Variable zu ändern, müssen Sie den Modifizierer `__block` storage type verwenden.

```
__block int val = 10;  
void (^blk)(void) = ^{  
    val = 20; // Correct! val now can be modified as an ordinary variable.  
};
```

Block online lesen: <https://riptutorial.com/de/ios/topic/6888/block>

Kapitel 31: CAAnimation

Bemerkungen

CAAnimation ist eine abstrakte Animationsklasse. Es bietet die grundlegende Unterstützung für die Protokolle **CAMediaTiming** und **CAAction**. Erstellen Sie zum **Animieren von Core-Animationsebenen** oder Scene Kit-Objekten Instanzen der konkreten Unterklassen **CABasicAnimation**, **CAKeyframeAnimation**, **CAAnimationGroup** oder **CATransition**.

Examples

Animieren Sie eine Ansicht von einer Position zur anderen.

Ziel c

```
CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"position.x"];
animation.fromValue = @0;
animation.toValue = @320;
animation.duration = 1;

[_label.layer addAnimation:animation forKey:@"basic"];
```

Schnell

```
let animation = CABasicAnimation(keyPath: "position.x")
animation.fromValue = NSNumber(value: 0.0)
animation.toValue = NSNumber(value: 320.0)

_label.layer.addAnimation(animation, forKey: "basic")
```

Die Ansicht bewegt sich horizontal von 0 bis 320. Wenn Sie die Ansicht nach vertikal verschieben möchten, ersetzen Sie einfach den Schlüsselfad wie folgt:

```
"position.y"
```

Animierte Ansicht - Wurf

ZIEL C

```
CATransition* transition = [CATransition animation];
transition.startProgress = 0;
transition.endProgress = 1.0;
transition.type = @"flip";
```

```
transition.subtype = @"fromLeft";
transition.duration = 0.8;
transition.repeatCount = 5;
[_label.layer addAnimation:transition forKey:@"transition"];
```

SCHNELL

```
var transition = CATransition()
transition.startProgress = 0
transition.endProgress = 1.0
transition.type = "flip"
transition.subtype = "fromLeft"
transition.duration = 0.8
transition.repeatCount = 5
label.layer.addAnimation(transition, forKey: "transition")
```

Ansicht drehen

```
CGRect boundingRect = CGRectMake(-150, -150, 300, 300);

CAKeyframeAnimation *orbit = [CAKeyframeAnimation animation];
orbit.keyPath = @"position";
orbit.path = CFAutorelease(CGPathCreateWithEllipseInRect(boundingRect, NULL));
orbit.duration = 4;
orbit.additive = YES;
orbit.repeatCount = HUGE_VALF;
orbit.calculationMode = kCAAnimationPaced;
orbit.rotationMode = kCAAnimationRotateAuto;

[_label.layer addAnimation:orbit forKey:@"orbit"];
```

Blick schütteln

Ziel c

```
CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position.x"];
animation.values = @[ @0, @10, @-10, @10, @0 ];
animation.keyTimes = @[ @0, @(1 / 6.0), @(3 / 6.0), @(5 / 6.0), @1 ];
animation.duration = 0.4;
animation.additive = YES;
[_label.layer addAnimation:animation forKey:@"shake"];
```

Swift 3

```
let animation = CAKeyframeAnimation(keyPath: "position.x")
animation.values = [ 0, 10, -10, 10, 0 ]
animation.keyTimes = [ 0, NSNumber(value: (1 / 6.0)), NSNumber(value: (3 / 6.0)),
NSNumber(value: (5 / 6.0)), 1 ]
animation.duration = 0.4
animation.isAdditive = true
label.layer.add(animation, forKey: "shake")
```

Push-View-Animation

Ziel c

```
CATransition *animation = [CATransition animation];  
[animation setSubtype:kCATransitionFromRight]; //kCATransitionFromLeft  
[animation setDuration:0.5];  
[animation setType:kCATransitionPush];  
[animation setTimingFunction:[CAMediaTimingFunction  
functionWithName:kCAMediaTimingFunctionEaseInEaseOut]];  
[[yourView layer] addAnimation:animation forKey:@"SwitchToView1"];
```

Schnell

```
let animation = CATransition()  
animation.subtype = kCATransitionFromRight //kCATransitionFromLeft  
animation.duration = 0.5  
animation.type = kCATransitionPush  
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionEaseInEaseOut)  
yourView.layer.addAnimation(animation, forKey: "SwitchToView1")
```

CAAnimation online lesen: <https://riptutorial.com/de/ios/topic/981/caanimation>

Kapitel 32: CAGradientLayer

Syntax

- `CAGradientLayer ()` // Gibt ein initialisiertes `CALayer`-Objekt zurück.
- `CAGradientLayer (Layer: Layer)` // Überschreiben, um benutzerdefinierte Felder des angegebenen Layers zu kopieren oder zu initialisieren.

Parameter

Parameter	Einzelheiten
Farbe	Ein Array von <code>CGColorRef</code> Objekten, das die Farbe jedes Verlaufsstopps definiert. Animierbar
Standorte	Ein optionales Array von <code>NSNumber</code> Objekten, das die Position jedes Gradientenstopps definiert. Animierbar
endPoint	Der Endpunkt des Farbverlaufs, wenn er im Koordinatenraum der Ebene gezeichnet wird. Animierbar
Startpunkt	Der Startpunkt des Farbverlaufs, wenn er im Koordinatenraum der Ebene gezeichnet wird. Animierbar
Art	Stil des Verlaufs, der von der Ebene gezeichnet wird. Der <code>kCAGradientLayerAxial</code>

Bemerkungen

- Verwenden `startPoint` und `endPoint` die Ausrichtung des ändern `CAGradientLayer` .
- Verwenden Sie die `locations` , um die Verteilung / Position der Farben zu beeinflussen.

Examples

CAGradientLayer erstellen

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds
```

```
// Color at the top of the gradient.
let topColor: CGColor = UIColor.red.cgColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellow.cgColor

// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set locations of the colors.
gradientLayer.locations = [0.0, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

Ergebnis:



CGGradientLayer mit mehreren Farben erstellen.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.blue.cgColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.yellow.cgColor

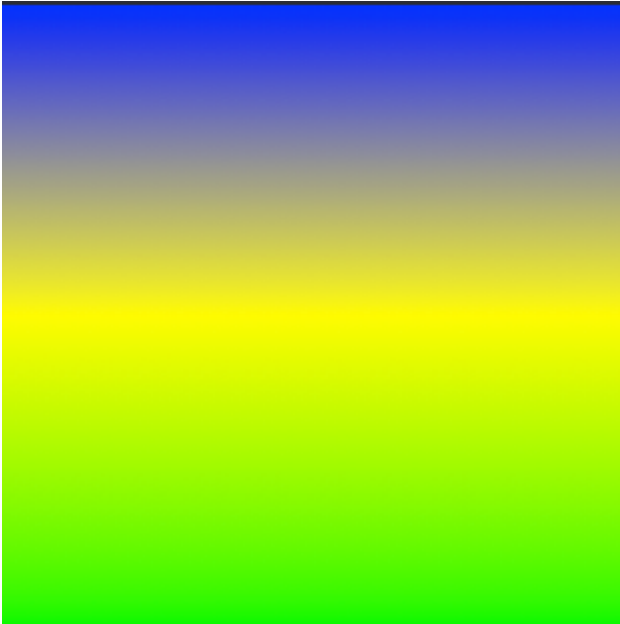
// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.green.cgColor

// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]
```

```
// Set locations of the colors.
gradientLayer.locations = [0.0, 0.5, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

Ergebnis:



Erstellen eines horizontalen CAGradientLayer.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.redColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellowColor().CGColor

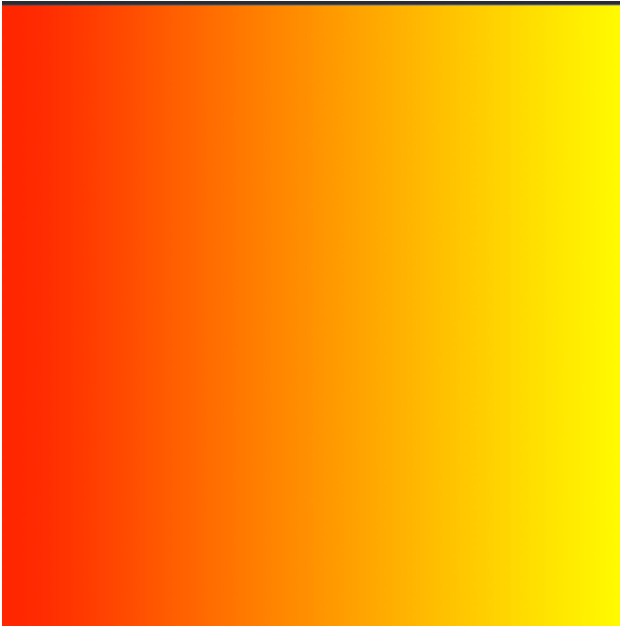
// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

Ergebnis:



Erstellen eines horizontalen CAGradientLayer mit mehreren Farben.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.greenColor().CGColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.blueColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.blackColor().CGColor

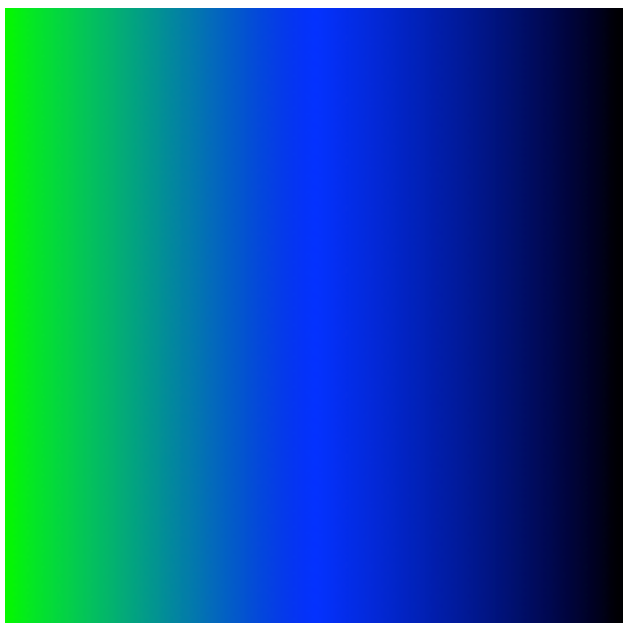
// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

Ergebnis:



Animieren einer Farbänderung in CAGradientLayer.

```
// Get the current colors of the gradient.
let oldColors = self.gradientLayer.colors

// Define the new colors for the gradient.
let newColors = [UIColor.red.cgColor, UIColor.yellow.cgColor]

// Set the new colors of the gradient.
self.gradientLayer.colors = newColors

// Initialize new animation for changing the colors of the gradient.
let animation: CABasicAnimation = CABasicAnimation(keyPath: "colors")

// Set current color value.
animation.fromValue = oldColors

// Set new color value.
animation.toValue = newColors

// Set duration of animation.
animation.duration = 0.3

// Set animation to remove once its completed.
animation.isRemovedOnCompletion = true

// Set receiver to remain visible in its final state when the animation is completed.
animation.fillMode = kCAFillModeForwards

// Set linear pacing, which causes an animation to occur evenly over its duration.
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)

// Set delegate of animation.
animation.delegate = self

// Add the animation.
self.gradientLayer.addAnimation(animation, forKey: "animateGradientColorChange")
```

Ergebnis:

Animate



CAGradientLayer online lesen: <https://riptutorial.com/de/ios/topic/1190/cagradientlayer>

Kapitel 33: CALayer

Examples

Einen CALayer erstellen

Sie können einen CALayer erstellen und seinen Rahmen folgendermaßen festlegen:

Schnell:

```
let layer = CALayer()
layer.frame = CGRect(x: 0, y: 0, width: 60, height: 80)
```

Ziel c:

```
CALayer *layer = [[CALayer alloc] init];
layer.frame = CGRectMake(0, 0, 60, 80);
```

Sie können es dann als Sublayer zu einem vorhandenen CALayer hinzufügen:

Schnell:

```
existingLayer.addSublayer(layer)
```

Ziel c:

```
[existingLayer addSublayer:layer];
```

Hinweis:

Dazu müssen Sie das QuartzCore-Framework einbeziehen.

Schnell:

```
@import QuartzCore
```

Ziel c

```
#import <QuartzCore/QuartzCore.h>
```

Partikel mit CAEmitterLayer erstellen

Die **CAEmitterLayer**- Klasse stellt ein Partikelemittersystem für Core-Animation bereit. Die Partikel werden durch **CAEmitterCell**- Instanzen **definiert** .

Die Partikel werden über der Hintergrundfarbe und dem Rand der Ebene gezeichnet.

```

var emitter = CAEmitterLayer()

emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
emitter.emitterShape = kCAEmitterLayerLine
emitter.emitterSize = CGSize(width: frame.size.width, height: 1)

emitter.emitterCells = cells
layer.addSublayer(emitter)

```

Emitter-Ansicht mit benutzerdefiniertem Bild

Zum Beispiel erstellen wir eine Ansicht, die die Emitter-Ebene enthält und Partikel animiert.

```

import QuartzCore

class ConfettiView: UIView {
    // main emitter layer
    var emitter: CAEmitterLayer!

    // array of color to emit
    var colors: [UIColor]!

    // intensity of appearance
    var intensity: Float!

    private var active :Bool!

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    func setup() {
        // initialization
        colors = [UIColor.redColor(),
                 UIColor.greenColor(),
                 UIColor.blueColor()
                ]
        intensity = 0.2

        active = false
    }

    func startConfetti() {
        emitter = CAEmitterLayer()

        emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
        emitter.emitterShape = kCAEmitterLayerLine
        emitter.emitterSize = CGSize(width: frame.size.width, height: 1)

        var cells = [CAEmitterCell]()
        for color in colors {
            cells.append(confettiWithColor(color))
        }
    }
}

```

```

    emitter.emitterCells = cells
    layer.addSublayer(emitter)
    active = true
}

func stopConfetti() {
    emitter?.birthRate = 0
    active = false
}

func confettiWithColor(color: UIColor) -> CAEmitterCell {
    let confetti = CAEmitterCell()

    confetti.birthRate = 10.0 * intensity
    confetti.lifetime = 180.0 * intensity
    confetti.lifetimeRange = 0
    confetti.color = color.CGColor
    confetti.velocity = CGFloat(350.0 * intensity)
    confetti.velocityRange = CGFloat(40.0 * intensity)
    confetti.emissionLongitude = CGFloat(M_PI)
    confetti.emissionRange = CGFloat(M_PI_4)
    confetti.spin = CGFloat(3.5 * intensity)
    confetti.spinRange = CGFloat(4.0 * intensity)

    // WARNING: A layer can set this property to a CGImageRef to display the image as its
    contents.
    confetti.contents = UIImage(named: "confetti")?.CGImage
    return confetti
}

internal func isActive() -> Bool {
    return self.active
}
}

```

Sie müssen ein "confetti" -Bild hinzufügen oder mit **confetti.contentsRect** "rect" definieren

So fügen Sie einen UIImage zu einem CALayer hinzu

Sie können der `layer` einer Ansicht einfach ein Bild hinzufügen, indem Sie dessen `contents` :

```
myView.layer.contents = UIImage(named: "star")?.CGImage
```

- Beachten Sie, dass der `UIImage` in ein `CGImage` konvertiert werden `CGImage` .

Wenn Sie das Bild in einer eigenen Ebene hinzufügen möchten, können Sie dies folgendermaßen tun:

```

let myLayer = CALayer()
let myImage = UIImage(named: "star")?.CGImage
myLayer.frame = myView.bounds
myLayer.contents = myImage
myView.layer.addSublayer(myLayer)

```

Erscheinungsbild ändern

Der obige Code erzeugt eine Ansicht wie diese. Das helle Blau ist die `UIView` und der dunkelblaue Stern ist der `UIImage`.



Wie Sie sehen, sieht es jedoch pixelig aus. Dies liegt daran, dass der `UIImage` kleiner ist als die `UIView` so dass er so skaliert wird, dass er die Ansicht ausfüllt. Dies ist der Standard, wenn Sie nichts anderes angeben.

Die folgenden Beispiele zeigen Variationen der Eigenschaft `contentsGravity` der Ebene. Der Code sieht so aus:

```
myView.layer.contents = UIImage(named: "star").CGImage
myView.layer.contentsGravity = kCAGravityTop
myView.layer.geometryFlipped = true
```

In iOS möchten Sie möglicherweise die Eigenschaft `geometryFlipped` auf `true` wenn Sie mit der oberen oder unteren Schwerkraft arbeiten. Andernfalls ist dies das Gegenteil von dem, was Sie erwarten. (Nur die Schwerkraft wird vertikal umgedreht, nicht das Rendern von Inhalten. Wenn Sie Probleme mit dem umgedrehten Inhalt haben, lesen Sie [diese Stack Overflow-Antwort](#).)

Es gibt zwei `UIView` Beispiele unten für jede `contentsGravity` Einstellung ist eine Ansicht größer als die `UIImage` und der andere kleiner. Auf diese Weise können Sie die Auswirkungen der Skalierung und der Schwerkraft erkennen.

`kCAGravityResize`

Dies ist die Standardeinstellung.



`kCAGravityResizeAspect`



`kCAGravityResizeAspectFill`



`kCAGravityCenter`



`kCAGravityTop`



`kCAGravityBottom`



`kCAGravityLeft`



`kCAGravityRight`



`kCAGravityTopLeft`



`kCAGravityTopRight`



`kCAGravityBottomLeft`



`kCAGravityBottomRight`



verbunden

- [Eigenschaft des Inhaltsmodus einer Ansicht](#)
- [Zeichnen eines UIImage in drawRect mit CGContextDrawImage](#)
- [CALayer-Tutorial: Erste Schritte](#)

Anmerkungen

- Dieses Beispiel stammt ursprünglich aus [dieser Stack Overflow-Antwort](#) .

Transformationen zu einem CALayer hinzufügen (übersetzen, drehen, skalieren)

Grundlagen

Es gibt eine Reihe verschiedener Transformationen, die Sie auf einer Ebene ausführen können, aber die grundlegenden sind dies

- übersetzen (bewegen)
- Rahmen
- drehen



Um Transformationen für einen `CALayer`, setzen Sie die `transform` des Layers auf einen `CATransform3D` Typ. Um eine Ebene zu übersetzen, würden Sie beispielsweise Folgendes tun:

```
myLayer.transform = CATransform3DMakeTranslation(20, 30, 0)
```

Das Wort `Make` wird im Namen zum Erstellen der ursprünglichen Transformation verwendet: `CATransform3D Make Translation`. Nachfolgende Transformationen, die angewendet werden, lassen das `Make`. Siehe zum Beispiel diese Rotation gefolgt von einer Übersetzung:

```
let rotation = CATransform3DMakeRotation(CGFloat(30.0 * M_PI / 180.0), 20, 20, 0)
myLayer.transform = CATransform3DTranslate(rotation, 20, 30, 0)
```

Da wir nun die Grundlage dafür haben, wie eine Transformation durchgeführt werden kann, wollen wir uns einige Beispiele ansehen, wie sie jeweils ausgeführt werden können. Zuerst werde ich jedoch zeigen, wie ich das Projekt aufbaue, falls Sie auch damit herumspielen wollen.

Konfiguration

Für die folgenden Beispiele habe ich eine `UIView` Anwendung eingerichtet und dem Storyboard eine `UIView` mit hellblauem Hintergrund hinzugefügt. Ich habe die Ansicht mit dem folgenden Code an den View-Controller angeschlossen:

```
import UIKit

class ViewController: UIViewController {

    var myLayer = CATextLayer()
```

```

@IBOutlet weak var myView: UIView!

override func viewDidLoad() {
    super.viewDidLoad()

    // setup the sublayer
    addSubLayer()

    // do the transform
    transformExample()
}

func addSubLayer() {
    myLayer.frame = CGRect(x: 0, y: 0, width: 100, height: 40)
    myLayer.backgroundColor = UIColor.blueColor().CGColor
    myLayer.string = "Hello"
    myView.layer.addSublayer(myLayer)
}

//***** Replace this function with the examples below *****/

func transformExample() {

    // add transform code here ...

}
}

```

Es gibt viele verschiedene Arten von `CALayer`, aber ich entschied mich für `CATextLayer` damit die Transformationen visuell klarer werden.

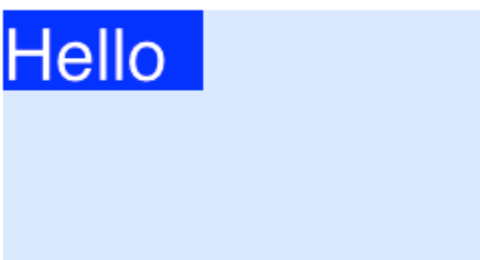
Übersetzen

Die Übersetzungstransformation verschiebt die Ebene. Die grundlegende Syntax lautet

```
CATransform3DMakeTranslation(tx: CGFloat, ty: CGFloat, tz: CGFloat)
```

Dabei ist t_x die Änderung in den x-Koordinaten, t_y ist die Änderung in y und t_z ist die Änderung in z.

Beispiel



Bei iOS befindet sich der Ursprung des Koordinatensystems oben links. Wenn wir also die Ebene um 90 Punkte nach rechts und 50 Punkte nach unten verschieben möchten, würden wir Folgendes tun:

```
myLayer.transform = CATransform3DMakeTranslation(90, 50, 0)
```

Anmerkungen

- Denken Sie daran, dass Sie dies im obigen Projektcode in die `transformExample()` Methode einfügen können.
- Da wir uns hier nur mit zwei Dimensionen befassen, wird `tz` auf `0`.
- Die rote Linie im obigen Bild verläuft von der Mitte des ursprünglichen Orts bis zur Mitte des neuen Orts. Das liegt daran, dass Transformationen in Bezug auf den Ankerpunkt durchgeführt werden und der Ankerpunkt standardmäßig in der Mitte der Ebene liegt.

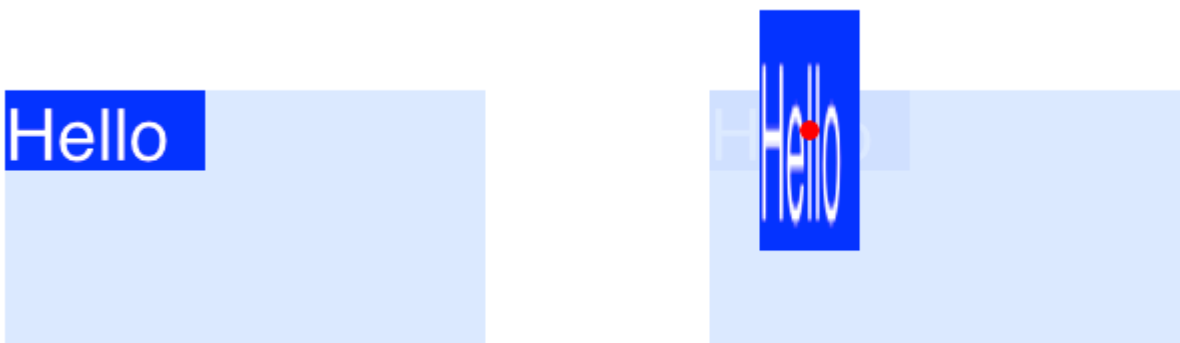
Rahmen

Die Skalentransformation dehnt oder zerquetscht die Ebene. Die grundlegende Syntax lautet

```
CATransform3DMakeScale(sx: CGFloat, sy: CGFloat, sz: CGFloat)
```

Dabei sind `sx`, `sy` und `sz` die Zahlen, mit denen die x-, y- bzw. z-Koordinaten skaliert (multipliziert) werden sollen.

Beispiel



Wenn wir die Breite halbieren und die Höhe verdreifachen wollten, würden wir Folgendes tun

```
myLayer.transform = CATransform3DMakeScale(0.5, 3.0, 1.0)
```

Anmerkungen

- Da wir nur in zwei Dimensionen arbeiten, multiplizieren wir einfach die z-Koordinaten mit `1,0`, um sie nicht zu beeinflussen.
- Der rote Punkt im Bild oben repräsentiert den Ankerpunkt. Beachten Sie, wie die Skalierung

im Verhältnis zum Ankerpunkt erfolgt. Das heißt, alles ist entweder zum Ankerpunkt hin oder von ihm weg gestreckt.

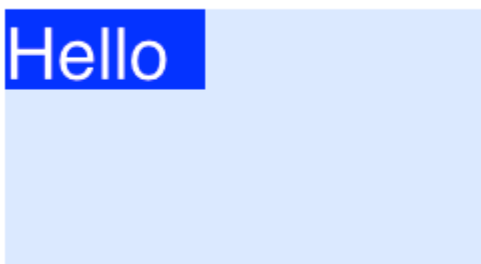
Drehen

Die Rotationstransformation dreht die Ebene um den Ankerpunkt (standardmäßig die Mitte der Ebene). Die grundlegende Syntax lautet

```
CATransform3DMakeRotation(angle: CGFloat, x: CGFloat, y: CGFloat, z: CGFloat)
```

Dabei ist `angle` der Winkel, um den die Ebene gedreht werden soll, und `x`, `y` und `z` sind die Achsen, um die gedreht werden soll. Wenn Sie eine Achse auf 0 setzen, wird die Drehung um diese bestimmte Achse abgebrochen.

Beispiel



Wenn wir eine Ebene um 30 Grad im Uhrzeigersinn drehen möchten, würden wir Folgendes tun:

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)
myLayer.transform = CATransform3DMakeRotation(radians, 0.0, 0.0, 1.0)
```

Anmerkungen

- Da wir in zwei Dimensionen arbeiten, möchten wir nur, dass die `xy`-Ebene um die `z`-Achse gedreht wird. Also setzen wir `x` und `y` auf `0.0` und `z` auf `1.0`.
- Dadurch wurde die Ebene im Uhrzeigersinn gedreht. Wir hätten uns gegen den Uhrzeigersinn drehen können, indem wir `z` auf `-1.0`.
- Der rote Punkt zeigt, wo sich der Ankerpunkt befindet. Die Drehung erfolgt um den Ankerpunkt.

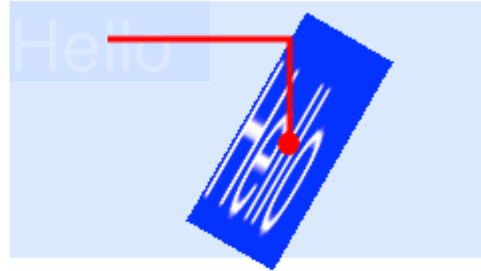
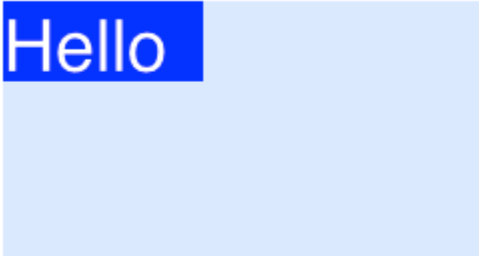
Mehrere Transformationen

Um mehrere Transformationen zu kombinieren, können wir die Concatination wie folgt verwenden

```
CATransform3DConcat(a: CATransform3D, b: CATransform3D)
```

Wir werden jedoch nur einen nach dem anderen machen. Die erste Transformation verwendet das `Make` im Namen. Bei den folgenden Transformationen wird `Make` nicht verwendet, die vorherige Transformation wird jedoch als Parameter verwendet.

Beispiel



Diesmal kombinieren wir alle drei der vorherigen Transformationen.

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)

// translate
var transform = CATransform3DMakeTranslation(90, 50, 0)

// rotate
transform = CATransform3DRotate(transform, radians, 0.0, 0.0, 1.0)

// scale
transform = CATransform3DScale(transform, 0.5, 3.0, 1.0)

// apply the transforms
myLayer.transform = transform
```

Anmerkungen

- Die Reihenfolge, in der die Transformationen durchgeführt werden.
- Alles wurde in Bezug auf den Ankerpunkt (roter Punkt) gemacht.

Ein Hinweis zu Ankerpunkt und Position

Wir haben alle unsere Transformationen oben durchgeführt, ohne den Ankerpunkt zu ändern. Manchmal ist es jedoch notwendig, sie zu ändern, als ob Sie sich um einen anderen Punkt außer der Mitte drehen möchten. Dies kann jedoch etwas schwierig sein.

Der Ankerpunkt und die Position befinden sich an derselben Stelle. Der Ankerpunkt wird als Einheit des Koordinatensystems der Ebene angegeben (Standardeinstellung ist `0.5, 0.5`), und die Position wird im Koordinatensystem der Überlagerung angegeben. Sie können so eingestellt werden

```
myLayer.anchorPoint = CGPoint(x: 0.0, y: 1.0)
myLayer.position = CGPoint(x: 50, y: 50)
```

Wenn Sie nur den Ankerpunkt festlegen, ohne die Position zu ändern, ändert sich der Rahmen, sodass die Position an der richtigen Stelle ist. Oder genauer, der Rahmen wird basierend auf dem neuen Ankerpunkt und der alten Position neu berechnet. Dies führt normalerweise zu unerwarteten Ergebnissen. Die folgenden zwei Artikel haben eine ausgezeichnete Diskussion darüber.

- [Über den Ankerpunkt](#)
- [Übersetzen drehen übersetzen?](#)

Siehe auch

- [CALayer , abgerundete Ecken und Schatten auf einem CALayer](#)
- [Verwenden eines Rahmens mit einem Bezier-Pfad für eine Ebene](#)

Dieses Beispiel stammt ursprünglich aus [diesem Stack Overflow-Beispiel](#) .

Animationen deaktivieren

`CALayer` Eigenschaftsanimationen sind standardmäßig aktiviert. Wenn dies unerwünscht ist, können sie wie folgt deaktiviert werden.

Schnell

```
CATransaction.begin()
CATransaction.setDisableActions(true)

// change layer properties that you don't want to animate

CATransaction.commit()
```

Ziel c

```
[CATransaction begin];
[CATransaction setDisableActions:YES];

// change layer properties that you don't want to animate

[CATransaction commit];
```

Abgerundete Ecken

```
layer.masksToBounds = true;
layer.cornerRadius = 8;
```

Schatten

Sie können 5 Eigenschaften auf jeder Ebene verwenden, um Ihre Schatten zu konfigurieren:

- `shadowOffset` - Diese Eigenschaft bewegt Ihren Schatten nach links / rechts oder nach oben / unten

```
self.layer.shadowOffset = CGSizeMake(-1, -1); // 1px left and up
self.layer.shadowOffset = CGSizeMake(1, 1); // 1px down and right
```

- `shadowColor` - legt die Farbe Ihres Schattens fest

```
self.layer.shadowColor = [UIColor blackColor].CGColor;
```

- `shadowOpacity` - Dies ist die Deckkraft des Schattens von 0 bis 1

```
self.layer.shadowOpacity = 0.2;
```

- `shadowRadius` - Dies ist der Unschärfekreis (entspricht der Unschärfe-Eigenschaft in Sketch oder Photoshop).

```
self.layer.shadowRadius = 6;
```

- `shadowPath` - Dies ist eine wichtige Eigenschaft für die Leistung, wenn nicht festgelegtes iOS den Schatten auf dem Alphakanal der Ansicht `shadowPath` Dies kann bei einer komplexen PNG mit Alpha die Leistung beeinträchtigen. Mit dieser Eigenschaft können Sie eine Form für Ihren Schatten erzwingen und dadurch leistungsfähiger werden.

Ziel c

```
self.layer.shadowPath = [UIBezierPath bezierPathWithOvalInRect:CGRectMake(0,0,100,100)];
//this does a circular shadow
```

Swift 3

```
self.layer.shadowPath = UIBezierPath(ovalIn: CGRect(x: 0, y: 0, width: 100, height: 100)).cgPath
```

CALayer online lesen: <https://riptutorial.com/de/ios/topic/1462/calayer>

Kapitel 34: Carthage iOS-Setup

Examples

Carthage Installation Mac

Karthago-Setup

Laden Sie die neueste Version von Carthage unter dem angegebenen Link [Download-Link](#) herunter

Laden Sie die Datei **Carthage.pkg** im Download-Bereich herunter.

Wenn der Download abgeschlossen ist, installieren Sie ihn durch Doppelklick auf die Download-Pkg-Datei.

Um zu überprüfen, ob der Download erfolgreich ist, führen Sie den folgenden Befehl in Ihrer Terminal- `0.18-19-g743fa0f` Version aus. Dies sollte die installierte Version wie `0.18-19-g743fa0f`

Carthage iOS-Setup online lesen: <https://riptutorial.com/de/ios/topic/7404/carthage-ios-setup>

Kapitel 35: CAShapeLayer

Syntax

1. shapeLayer.fillColor
2. shapeLayer.fillRule
3. shapeLayer.lineCap
4. shapeLayer.lineDashPattern
5. shapeLayer.lineDashPhase
6. shapeLayer.lineJoin

Bemerkungen

Die CAShapeLayer-Klasse zeichnet in ihrem Koordinatenraum einen kubischen Bezier-Spline. Die Form wird zwischen dem Inhalt der Ebene und ihrer ersten Unterschicht zusammengesetzt.

Examples

Grundlegende CAShapeLayer-Operation

UIBezierPath zum Erstellen eines ShapeLayer mit Kreispfad

```
CAShapeLayer *circleLayer = [CAShapeLayer layer];
[circleLayer setPath:[UIBezierPath bezierPathWithOvalInRect:
CGRectMake(50, 50, 100, 100)] CGPath]];
circleLayer.lineWidth = 2.0;
[circleLayer setStrokeColor:[UIColor redColor] CGColor]];
[circleLayer setFillColor:[UIColor clearColor] CGColor]];
circleLayer.lineJoin = kCALineJoinRound; //4 types are available to create a line style
circleLayer.lineDashPattern = [NSArray arrayWithObjects:
[NSNumber numberWithInt:2],[NSNumber numberWithInt:3 ], nil];
// self.origImage is parentView
[[self.view layer] addSublayer:circleLayer];
self.currentShapeLayer = circleLayer; // public value using to keep that reference of the
shape Layer
self.view.layer.borderWidth = 1.0f;
self.view.layer.borderColor = [[UIColor blueColor]CGColor]; // that will plotted in the
mainview
```

Entfernen Sie ShapeLayer

Behalten Sie einen Verweis auf diese Formebene. Sie haben beispielsweise eine Eigenschaft currentShapeLayer: Nachdem Sie nun eine Referenz haben, können Sie die Ebene problemlos entfernen:

Typ 1:

```
[self.currentShapeLayer removeFromSuperlayer];
```

Typ 2:

```
self.view.layer.sublayers = nil ; //removed all earlier shapes
```

Andere Bedienung

```
//Draw Square Shape

CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 20, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = nil;
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Draw Circle Shape

CAShapeLayer *circleShape = [CAShapeLayer layer];
circleShape.frame = CGRectMake(160, 20, 120, 120);
circleShape.lineWidth = 2.0;
circleShape.fillColor = nil;
circleShape.strokeColor = [[UIColor redColor] CGColor];
circleShape.path = [UIBezierPath bezierPathWithOvalInRect:circleShape.bounds].CGPath;
[[self.view layer] addSublayer:circleShape];

//Subpaths
//UIBezierPath can have any number of "path segments" (or subpaths) so you can effectively
draw as many shapes or lines as you want in a single path object

CAShapeLayer *shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(20, 140, 200, 200);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

CGMutablePathRef combinedPath= CGPathCreateMutableCopy(circleShape.path);
CGPathAddPath(combinedPath, NULL, squareLayer.path);

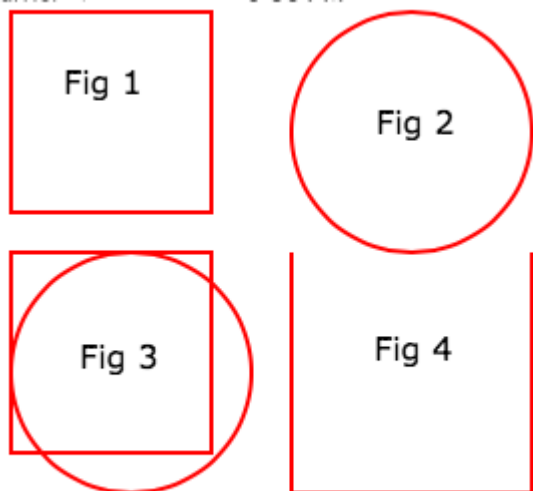
shapeLayer.path = combinedPath;
[[self.view layer] addSublayer:shapeLayer];

//Open Path
// Paths do not need to connect their end points back to their starting points. A path that
connects back to its starting point is called a closed path, and one that does not is called
an open path.

shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(160, 140, 300, 300);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

UIBezierPath *linePath=[UIBezierPath bezierPath];
[linePath moveToPoint:CGPointZero];
[linePath addLineToPoint:CGPointMake(0 , 120)];
[linePath addLineToPoint:CGPointMake(120 , 120)];
```

```
[linePath addLineToPoint:CGPointMake(120 , 0)];
shapeLayer.path = linePath.CGPath;
[[self.view layer] addSublayer:shapeLayer];
```



Füllkonzepte // Füllfarbe

```
CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Fill Pattern Color
//images.jpeg

squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor colorWithPatternImage:[UIImage
imageName:@"images.jpeg"]]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Fill Rule

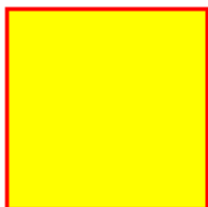
//Type 1: kCAFillRuleNonZero
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(0, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleNonZero; // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
UIBezierPath *outerPath = [UIBezierPath bezierPathWithRect:CGRectInset (squareLayer.bounds,
20.0, 20.0)];
UIBezierPath *innerPath = [UIBezierPath bezierPathWithRect:CGRectInset (squareLayer.bounds,
50.0, 50.0)];
CGMutablePathRef combinedPath= CGPathCreateMutableCopy (outerPath.CGPath);
```

```

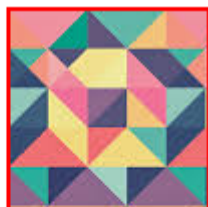
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];

//Type 2: kCAFillRuleEvenOdd
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleEvenOdd; // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
outerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds, 20.0, 20.0)];
innerPath = [UIBezierPath bezierPathWithRect:CGRectInset(squareLayer.bounds, 50.0, 50.0)];
combinedPath= CGPathCreateMutableCopy(outerPath.CGPath);
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];

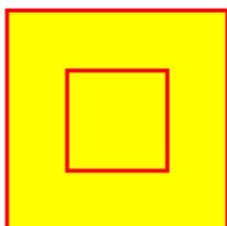
```



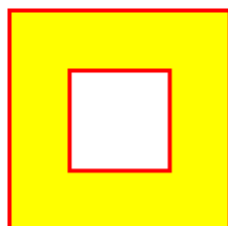
Fill color



Fill Pattern



Fill Rule Zero



Fill Rule Even

Listete die aufrufenden Stileigenschaften auf

```

fillColor
    Fill the color based on the drawn shape.

fillRule
    Fill Rule the there are two rule is applied to draw the shape.
    1. kCAFillRuleNonZero
    2. kCAFillRuleEvenOdd

lineCap
    Below type used to change the style of the line.
    1. kCALineCapButt
    2. kCALineCapRound
    3. kCALineCapSquare

lineDashPattern
    The dash pattern applied to the shape's path when stroked.
    Create DashStyle while you will stroke the line.

lineDashPhase
    The dash phase applied to the shape's path when stroked. Animatable.

```

lineJoin
Line join style for the shape path. Below style use to draw the line join style.

1. kCALineJoinMiter
2. kCALineJoinRound
3. kCALineJoinBevel

lineWidth
Which using to set the line width.

miterLimit
The miter limit used when stroking the shape's path. Animatable.

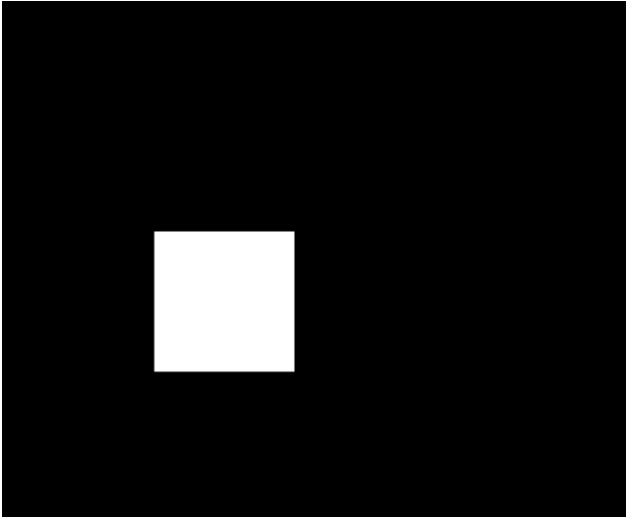
strokeColor
Set the stroke color based on the path of the line.

strokeStart
When the stroke will start.

strokeEnd
When the stroke will end.

Rechteck zeichnen

```
CAShapeLayer *mask = [[CAShapeLayer alloc] init];  
mask.frame = CGRectMake(50, 50, 100, 100);  
CGFloat width = 100;  
CGFloat height = 100;  
CGMutablePathRef path = CGPathCreateMutable();  
CGPathMoveToPoint(path, nil, 30, 30);  
CGPathAddLineToPoint(path, nil, width, 30);  
CGPathAddLineToPoint(path, nil, width, height);  
CGPathAddLineToPoint(path, nil, 30, height);  
CGPathAddLineToPoint(path, nil, 30, 30);  
CGPathCloseSubpath(path);  
  
mask.path = path;  
CGPathRelease(path);  
  
self.view.layer.mask = mask;
```



Kreis zeichnen

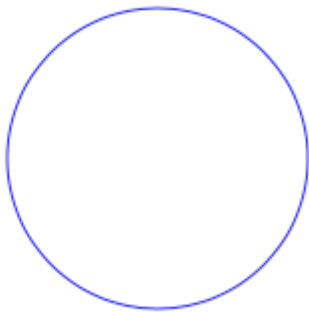
```
CAShapeLayer *circle = [CAShapeLayer layer];

[circle setPath:[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

[circle setStrokeColor:[UIColor blueColor] CGColor]];

[circle setFillColor:[UIColor clearColor] CGColor]];

[[self.view layer] addSublayer:circle];
```



CAShapeLayer-Animation

```
CAShapeLayer *circle = [CAShapeLayer layer];

[circle setPath:[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

[circle setStrokeColor:[UIColor blueColor] CGColor]];

[circle setFillColor:[UIColor clearColor] CGColor]];

[[self.view layer] addSublayer:circle];
```

```
CABasicAnimation *pathAnimation = [CABasicAnimation animationWithKeyPath:@"strokeEnd"];
pathAnimation.duration = 1.5f;
pathAnimation.fromValue = [NSNumber numberWithFloat:0.0f];
pathAnimation.toValue = [NSNumber numberWithFloat:1.0f];
pathAnimation.repeatCount = 10;
pathAnimation.autoreverses = YES;

[circle addAnimation:pathAnimation
      forKey:@"strokeEnd"];
```



CAShapeLayer online lesen: <https://riptutorial.com/de/ios/topic/3575/cashapelayer>

Kapitel 36: CGContext-Referenz

Bemerkungen

Der undurchsichtige **CGContextRef**- Typ steht für ein 2D-Zeichnungsziel. Ein Grafikkontext enthält Zeichnungsparameter und alle gerätespezifischen Informationen, die erforderlich sind, um die Farbe auf einer Seite an das Ziel zu übertragen, unabhängig davon, ob es sich bei dem Ziel um ein Fenster in einer Anwendung, ein Bitmap-Bild, ein PDF-Dokument oder einen Drucker handelt.

Examples

Linie zeichnen

```
CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetLineWidth(context, 5.0);
CGColorSpaceRef colorspace = CGColorSpaceCreateDeviceRGB();
CGContextMoveToPoint(context, 200, 400);
CGContextAddLineToPoint(context, 100, 100);
CGContextStrokePath(context);
CGColorSpaceRelease(colorspace);
```



Text zeichnen

Draw To erfordert, dass das **Core Text Framework** in der Erstellungsphase hinzugefügt wird

```
[NSString* textToDraw = @"Welcome to the world of IOS";

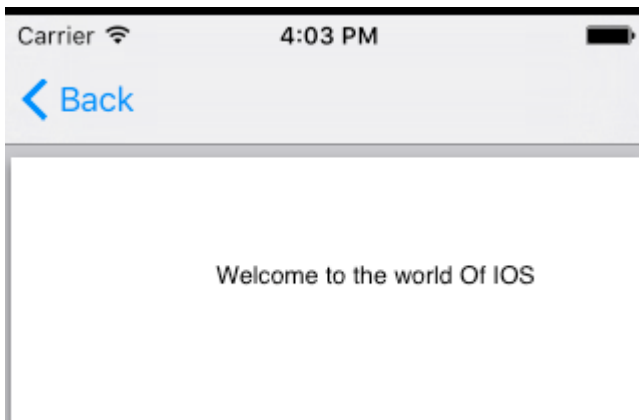
CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);
CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);
CGRect frameRect = CGRectMake(0, 0, 300, 100);
CGMutablePathRef framePath = CGPathCreateMutable();
CGPathAddRect(framePath, NULL, frameRect);

CFRange currentRange = CFRangeMake(0, 0);
CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
```



```
NULL);  
    CGPathRelease(framePath);  
    CGContextRef currentContext = UIGraphicsGetCurrentContext();  
  
    CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);  
    CGContextTranslateCTM(currentContext, 200, 300);  
    CGContextScaleCTM(currentContext, 2, -2);  
    CTFrameDraw(frameRef, currentContext);  
  
    CFRelease(frameRef);  
    CFRelease(stringRef);  
    CFRelease(framesetter);
```



CGContext-Referenz online lesen: <https://riptutorial.com/de/ios/topic/2664/cgcontext-referenz>

Kapitel 37: CLLocation

Examples

Entfernungsfilter mit

Beispiel:

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.desiredAccuracy = kCLLocationAccuracyBest;
locationManager.distanceFilter = 5;
```

Im obigen Beispielcode werden beispielsweise Standortänderungen von weniger als 5 Metern nicht an den Rückruf gesendet, sondern ignoriert.

Abrufen des Benutzerstandorts mithilfe von CLLocationManager

1 - Fügen Sie das CoreLocation.framework in Ihr Projekt ein. Dies wird durch Klicken auf erreicht:

```
root directory -> build phases -> Link Binary With Libraries
```

Klicken Sie auf die Schaltfläche (+), suchen Sie nach CoreLocation.framework und klicken Sie auf Hinzufügen.

2- Ändern Sie die Datei info.plist, um die Erlaubnis zur Verwendung des Benutzerstandorts zu erfragen, indem Sie sie als Quellcode öffnen. Fügen Sie einen der folgenden Schlüssel ein: Wertepaar unter dem Tag, um die Verwendung des Benutzerstandorts während der Verwendung der Anwendung abzufragen:

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>message to display when asking for permission</string>
```

3- Importieren Sie CoreLocation in den ViewController, der es verwendet.

```
import CoreLocation
```

4- Stellen Sie sicher, dass Ihr ViewController dem CLLocationManagerDelegate-Protokoll entspricht

```
class ViewController: UIViewController, CLLocationManagerDelegate {}
```

Nach diesen Schritten können wir ein CLLocationManager-Objekt als Instanzvariable erstellen und im ViewController verwenden.

```
var manager:CLLocationManager!
```

Wir verwenden hier nicht "let", da wir den Manager ändern, um seinen Delegierten, die Mindestentfernung vor dem Aktualisierungsereignis und seine Genauigkeit anzugeben

```
//initialize the manager
manager = CLLocationManager()

//specify delegate
manager.delegate = self

//set the minimum distance the phone needs to move before an update event is triggered (for
example: 100 meters)
manager.distanceFilter = 100

//set Accuracy to any of the following depending on your use case

//let kCLLocationAccuracyBestForNavigation: CLLocationAccuracy
//let kCLLocationAccuracyBest: CLLocationAccuracy
//let kCLLocationAccuracyNearestTenMeters: CLLocationAccuracy
//let kCLLocationAccuracyHundredMeters: CLLocationAccuracy
//let kCLLocationAccuracyKilometer: CLLocationAccuracy
//let kCLLocationAccuracyThreeKilometers: CLLocationAccuracy

manager.desiredAccuracy = kCLLocationAccuracyBest

//ask the user for permission
manager.requestWhenInUseAuthorization()

//Start collecting location information
if #available(iOS 9.0, *) {

    manager.requestLocation()

} else {

    manager.startUpdatingLocation()

}
```

Um nun auf die Standortaktualisierungen zuzugreifen, können wir die Funktion implementieren, unter der Überstunden genannt werden, in denen der distanceFilter erreicht wird.

```
func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
{}
```

Der Parameter locations ist ein Array von CLLocation-Objekten, die den tatsächlichen Standort des Geräts darstellen. Von diesen Objekten aus können Sie auf die folgenden Attribute zugreifen: coordinate, altitude, floor, horizontalAccuracy, verticalAccuracy, timestamp, description, course, speed **Genauigkeit** coordinate, altitude, floor, horizontalAccuracy, verticalAccuracy, timestamp, description, course, speed **und eine Funktionsentfernung distance(from:), die die Entfernung zwischen zwei Orten misst.**

Hinweis: Beim Anfordern der Berechtigung für den Standort gibt es zwei verschiedene Arten der Autorisierung.

Die Berechtigung "Bei Verwendung" erteilt der App nur die Berechtigung, Ihren Standort zu empfangen, wenn die App verwendet wird oder sich im Vordergrund

befindet.

Die Berechtigung "Immer" gibt der App Hintergrundberechtigungen, die dazu führen können, dass die Akkulaufzeit verringert wird, falls Ihre App geschlossen wird.

Plist-Datei sollte nach Bedarf angepasst werden.

CLLocation online lesen: <https://riptutorial.com/de/ios/topic/2002/cllocation>

Kapitel 38: CloudKit

Bemerkungen

Unterstützte Typen

- NSData
- NSDate (Datum)
- NSNumber (Int / Double)
- NSString (String)
- NSArray (Array)
- CLLocation
- CKReferenz
- CKAsset

[Mehr Details](#)

[CloudKit-Dashboard](#)

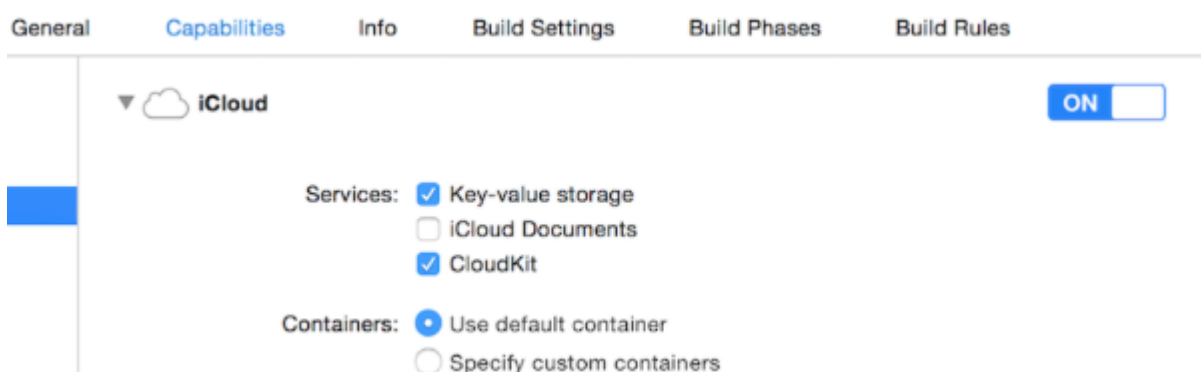
Examples

App zur Verwendung mit CloudKit registrieren

Sie benötigen eine Berechtigungsdatei, damit die App mithilfe von CloudKit auf Ihre iCloud zugreifen und Datensätze schreiben kann.

Befolgen Sie die Schritte, um von Ihrer App aus Zugriff auf iCloud zu gewähren:

- 1- Wählen Sie das Projekt im Projektnavigator aus und öffnen Sie die Registerkarte Allgemein.
- 2- Setzen Sie im Abschnitt Identität Ihre Entwickler-Apple-ID auf das Team-Dropdown-Menü. (Wenn es nicht verfügbar ist, fügen Sie es im Xcode-Menü hinzu -> Voreinstellungen -> Konten.)
- 3- Wechseln Sie in den Projekteigenschaften zur Registerkarte Funktionen und aktivieren Sie iCloud. Wählen Sie dann "Schlüsselwertspeicher" und "CloudKit" aus.



4- Stellen Sie sicher, dass diese Punkte markiert sind:

-
- Steps: ✓ Add the "iCloud" entitlement to your App ID
✓ Add the "iCloud containers" entitlement to your App ID
✓ Add the "iCloud" entitlement to your entitlements file
✓ Link CloudKit.framework

Wenn alle Elemente markiert sind, kann Ihre App CloudKit verwenden.

CloudKit Dashboard verwenden

Alle mit CloudKit-Code erstellten Datensätze können in CloudKit Dashboard in der Vorschau angezeigt, bearbeitet und sogar entfernt werden. Um auf das CloudKit Dashboard zuzugreifen, klicken Sie [hier](#).

Das Dashboard enthält mehrere Teile:

- Datensatztypen (auf die später noch eingegangen wird)
- Sicherheitsrollen (Hier können Sie Datenbanken als öffentlich oder privat festlegen)
- Abonnementtypen (die Ihre App für [Apple Push Notifications \(APNs\)](#) registrieren könnte, um Sie zu benachrichtigen, wenn ein Datensatz geändert wird)

Datensatztypen

Hier erhalten Sie eine Liste aller in der App vorhandenen Datensatztypen. Wenn Sie das CloudKit Dashboard zum ersten Mal für eine App öffnen, gibt es dort einen Datensatztyp namens Benutzer, den Sie verwenden können oder einfach löschen und einen eigenen verwenden können.

Auf dieser Seite können Sie Ihre Daten manuell eingeben. In den meisten Fällen ist dies natürlich sinnlos, da iOS SDK damit weitaus besser umgehen kann als das Dashboard. Die Funktionalität ist jedoch auch vorhanden, wenn Sie dies vorziehen. Diese Seite wird am häufigsten für die Vorschau von Typen verwendet.

Daten in CloudKit speichern

Um ein Datum in CloudKit zu speichern, müssen wir Folgendes vornehmen:

- Eine `CKRecordID` (der Schlüssel Ihres eindeutigen Datensatzes)
- Ein `CKRecord` (der Daten enthält)

Einen Aufnahmeschlüssel erstellen

Um sicherzustellen, dass jede neue Datensatzkennung eindeutig ist, verwenden wir den aktuellen *Zeitstempel*, der eindeutig ist. Wir erhalten den Zeitstempel mit der `NSDate`-Methode `timeIntervalSinceReferenceDate()`. Es ist in der Form von `###.###` (`#` sind Zahlen), wobei wir den ganzzahligen Teil verwenden werden. Dazu teilen wir die Zeichenfolge auf:

Schnell

```
let timestamp = String(format: "%f", NSDate.timeIntervalSinceReferenceDate())
let timestampParts = timestamp.componentsSeparatedByString(".")
let recordID = CKRecordID(recordName: timestampParts[0])
```

Aufnahme machen

Um den Datensatz zu erstellen, sollten wir den Datensatztyp (wie in Verwenden von CloudKit Dashboard erläutert) als Benutzer, die ID als das, was wir gerade gemacht haben, und die Daten angeben. Hier fügen wir dem Datensatz einen Beispieltext, ein Bild und das aktuelle Datum hinzu:

Schnell

```
let record = CKRecord(recordType: "Users", recordID: recordID)
record.setObject("Some Text", forKey: "text")
record.setObject(CKAsset(fileURL: someValidImageURL), forKey: "image")
record.setObject(NSDate(), forKey: "date")
```

Ziel c

```
CKRecord *record = [[CKRecord alloc] initWithRecordType: "Users" recordID: recordID];
[record setObject: "Some Text" forKey: "text"];
[record setObject: [CKAsset assetWithURL: someValidImageURL] forKey: "image"];
[record setObject: [[NSDate alloc] init] forKey: "date"];
```

Hinweis

Hier haben wir den `UIImage` direkt zum Datensatz hinzugefügt, da das `UIImage` in `CKAsset` nicht direkt unterstützt wird. Daher haben wir `UIImage` in `CKAsset` konvertiert.

Zugriff auf den Container

Schnell

```
let container = CKContainer.defaultContainer()
let database = container.privateCloudDatabase // or container.publicCloudDatabase
```

Speichern der Datensätze in der CloudKit-

Datenbank

Schnell

```
database.saveRecord(record, completionHandler: { (_, error) -> Void in
    print(error ?? "")
})
```

CloudKit online lesen: <https://riptutorial.com/de/ios/topic/4946/cloudkit>

Kapitel 39: Code-Signatur

Examples

Bereitstellungsprofile

Um eine IPA-Datei in XCode erstellen zu können, müssen Sie Ihre Anwendung mit einem Zertifikat und einem Bereitstellungsprofil signieren. Diese können unter <https://developer.apple.com/account/ios/profile/create> erstellt werden

Bereitstellungstypen

Bereitstellungsprofile sind in zwei Typen unterteilt: Entwicklung und Verteilung:

Entwicklung

- iOS-App-Entwicklung / tvOS-App-Entwicklung: Wird bei der Entwicklung verwendet, um Ihre App auf einem Testgerät zu installieren.

Verteilung

- App Store / tvOS App Store - Zum Signieren Ihrer Anwendung für den App Store-Upload.
- In House - Wird für die Verteilung Ihrer App im Unternehmen an Geräte innerhalb Ihres Unternehmens verwendet.
- Ad Hoc / tvOS Ad Hoc - Wird verwendet, um Ihre App auf eine begrenzte Anzahl bestimmter Geräte zu verteilen (z. B. müssen Sie die UDIDs der Geräte kennen, auf denen Sie Ihre App installieren möchten).

Code-Signatur online lesen: <https://riptutorial.com/de/ios/topic/6055/code-signatur>

Kapitel 40: Codierbar

Einführung

Codable wird mit Xcode 9, iOS 11 und Swift 4 hinzugefügt. Codable wird verwendet, um Ihre Datentypen für die Kompatibilität mit externen Repräsentationen wie JSON codierbar und decodierbar zu machen.

Codable verwenden, um sowohl das Codieren als auch das Decodieren zu unterstützen. Deklarieren Sie die Konformität mit Codable, die die Protokolle Encodable und Decodable kombiniert. Dieser Vorgang ist dafür bekannt, Ihre Typen codierbar zu machen.

Examples

Verwendung von Codable mit JSONEncoder und JSONDecoder in Swift 4

Nehmen wir ein Beispiel mit der Struktur des Films. Hier haben wir die Struktur als codierbar definiert. So können wir es leicht codieren und decodieren.

```
struct Movie: Codable {
    enum MovieGenre: String, Codable {
        case horror, skifi, comedy, adventure, animation
    }

    var name : String
    var moviesGenre : [MovieGenre]
    var rating : Int
}
```

Wir können ein Objekt aus einem Film erstellen wie:

```
let upMovie = Movie(name: "Up", moviesGenre: [.comedy , .adventure, .animation], rating : 4)
```

Der upMovie enthält den Namen "Up" und sein movieGenre ist Comedy, Abenteuer und Animation, die 4 von 5 Punkten enthält.

Kodieren

JSONEncoder ist ein Objekt, das Instanzen eines Datentyps als JSON-Objekte codiert. JSONEncoder unterstützt das Codable-Objekt.

```
// Encode data
let jsonEncoder = JSONEncoder()
do {
    let jsonData = try jsonEncoder.encode(upMovie)
    let jsonString = String(data: jsonData, encoding: .utf8)
    print("JSON String : " + jsonString!)
}
```

```
catch {  
}
```

JSONEncoder gibt uns die JSON-Daten an, die zum Abrufen der JSON-Zeichenfolge verwendet werden.

Ausgabestring wird wie folgt aussehen:

```
{  
  "name": "Up",  
  "moviesGenere": [  
    "comedy",  
    "adventure",  
    "animation"  
  ],  
  "rating": 4  
}
```

Dekodieren

JSONDecoder ist ein Objekt, das Instanzen eines Datentyps von JSON-Objekten dekodiert. Wir können das Objekt von der JSON-Zeichenfolge zurückholen.

```
do {  
  // Decode data to object  
  
  let jsonDecoder = JSONDecoder()  
  let upMovie = try jsonDecoder.decode(Movie.self, from: jsonData)  
  print("Rating : \(upMovie.name)")  
  print("Rating : \(upMovie.rating)")  
}  
catch {  
}
```

Durch die Dekodierung der JSONData erhalten wir das Movie-Objekt zurück. So können wir alle Werte erhalten, die in diesem Objekt gespeichert sind.

Ausgabe wird wie folgt sein:

```
Name : Up  
Rating : 4
```

Codierbar online lesen: <https://riptutorial.com/de/ios/topic/10639/codierbar>

Kapitel 41: Content Hugging / Inhaltskomprimierung im Autolayout

Bemerkungen

Priorität der Inhaltskomprimierungsresistenz

Dieser Wert legt fest, wie widerstandsfähig eine Ansicht gegen das Komprimieren oder Verkleinern ist. Ein höherer Wert bedeutet, dass die Ansicht weniger komprimiert wird und eher gleich bleibt.

Priorität für Inhalte

Dieser Wert bestimmt, wie widerstandsfähig eine Ansicht beim Erweitern ist. Sie können sich vorstellen, hier "umarmen" zu bedeuten, "Größe passend" - die Grenzen der Ansicht werden "umarmen" oder nahe an der inhärenten Inhaltsgröße liegen. Ein höherer Wert bedeutet, dass die Sicht weniger wahrscheinlich wächst und eher gleich bleibt.

Examples

Definition: Intrinsische Inhaltsgröße

Vor dem automatischen Layout mussten Sie den Schaltflächen und anderen Steuerelementen immer mitteilen, wie groß sie sein sollten, indem Sie entweder ihre Rahmen- oder Begrenzungseigenschaften festlegen oder ihre Größe im Interface Builder ändern. Es stellt sich jedoch heraus, dass die meisten Steuerelemente in der Lage sind, anhand ihres Inhalts den für sie benötigten Platz zu bestimmen.

Ein **Etikett** weiß, wie groß und groß es ist, weil es die Länge des darauf eingestellten Textes sowie die Schriftgröße für diesen Text kennt. Ebenso für eine **Schaltfläche**, die den Text mit einem Hintergrundbild und etwas Auffüllung kombiniert.

Dasselbe gilt für segmentierte Steuerelemente, Fortschrittsbalken und die meisten anderen Steuerelemente, obwohl einige nur eine vorbestimmte Höhe, aber eine unbekannte Breite haben.

Dies ist als die intrinsische Inhaltsgröße bekannt, und es ist ein wichtiges Konzept in Auto Layout. Beim automatischen Layout werden die Steuerelemente nach der Größe gefragt, die sie benötigen, und der Bildschirm wird basierend auf diesen Informationen angezeigt.

Normalerweise möchten Sie die `intrinsic content size`, aber in manchen Fällen möchten Sie dies möglicherweise nicht. Sie können dies verhindern, indem Sie eine explizite Breite oder Höhe für ein Steuerelement festlegen.

Stellen Sie sich vor, was passiert, wenn Sie ein Bild in einem UIImageView festlegen, wenn dieses

Bild viel größer als der Bildschirm ist. Normalerweise möchten Sie Bildansichten eine feste Breite und Höhe zuweisen und den Inhalt skalieren, es sei denn, Sie möchten die Größe der Ansicht an die Abmessungen des Bildes anpassen.

Referenz: <https://www.raywenderlich.com/115444/auto-layout-tutorial-in-ios-9-part-2-beschränkungen>

Content Hugging / Inhaltskomprimierung im Autolayout online lesen:

<https://riptutorial.com/de/ios/topic/6899/content-hugging---inhaltskomprimierung-im-autolayout>

Kapitel 42: Core Graphics

Examples

Erstellen eines Kerngrafikkontexts

Core Graphics-Kontext

Ein Core Graphics-Kontext ist eine Leinwand, die wir darin zeichnen und einige Eigenschaften wie die Linienstärke festlegen können.

Einen Kontext machen

Um einen Kontext zu `UIGraphicsBeginImageContextWithOptions()`, verwenden wir die C-Funktion `UIGraphicsBeginImageContextWithOptions()`. Wenn wir mit dem Zeichnen fertig sind, rufen wir einfach `UIGraphicsEndImageContext()` auf, um den Kontext zu beenden:

Schnell

```
let size = CGSize(width: 256, height: 256)

UIGraphicsBeginImageContextWithOptions(size, false, 0)

let context = UIGraphicsGetCurrentContext()

// drawing code here

UIGraphicsEndImageContext()
```

Ziel c

```
CGSize size = [CGSize width:256 height:256];

UIGraphicsBeginImageContextWithOptions(size, NO, 0);

CGContext *context = UIGraphicsGetCurrentContext();

// drawing code here

UIGraphicsEndImageContext();
```

Im obigen Code haben wir 3 Parameter an die `UIGraphicsBeginImageContextWithOptions()` Funktion übergeben:

1. Ein `CGSize` Objekt, das die gesamte Größe des Kontexts (der Leinwand) speichert.

2. Ein boolescher Wert, bei dem der Kontext undurchsichtig ist, wenn er wahr ist
3. Ein ganzzahliger Wert, der die Skala festlegt (1 für Nicht-Retina, 2 für Retina und 3 für Retina-HD-Bildschirme). Wenn der Wert auf 0 gesetzt ist, behandelt das System die Waage automatisch basierend auf dem Zielgerät.

Präsentieren der gezeichneten Leinwand für den Benutzer

Schnell

```
let image = UIGraphicsGetImageFromCurrentImageContext()  
imageView.image = image //assuming imageView is a valid UIImageView object
```

Ziel c

```
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();  
imageView.image = image; //assuming imageView is a valid UIImageView object
```

Core Graphics online lesen: <https://riptutorial.com/de/ios/topic/5530/core-graphics>

Kapitel 43: Core Spotlight in iOS

Examples

Core-Spotlight

Ziel c

1. Erstellen Sie ein neues iOS-Projekt und fügen Sie Ihrem Projekt das *CoreSpotlight*- und *MobileCoreServices*- Framework hinzu.



General

Capabilities

PROJECT



CoreSpotlighSample

TARGETS



CoreSpotlighSample



CoreSpotlighSampl...



CoreSpotlighSampl...



▶ **Target Dependencies (0 it**

▶ **Compile Sources (3 item...**

▼ **Link Binary With Libraries**

Name



CoreSp



Mobile



▶ **Copy Bundle Resources (4**

2. Erstellen Sie das eigentliche `CSSearchableItem` und verknüpfen Sie den `uniqueIdentifier`, den `domainIdentifier` und das `AttributSet`. Indizieren Sie schließlich das `CSSearchableItem` mit `[[CSSearchableIndex defaultSearchableIndex] ...]` wie unten gezeigt.

```

4 //
5 // Created by Mayqiyue on 7/10/15.
6 // Copyright © 2015 mayqiyue. All rights reserved.
7 //
8
9 #import "ViewController.h"
10 #import <CoreSpotlight/CoreSpotlight.h>
11 #import <MobileCoreServices/MobileCoreServices.h>
12
13 @interface ViewController ()
14
15 @end
16
17 @implementation ViewController
18
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     [self setupCoreSpotlightSearch];
22 }
23
24 - (void)setupCoreSpotlightSearch {
25     CSSearchableItemAttributeSet *attributeSet =
26         [CSSearchableItemAttributeSet new];
27     (NSString *)kUTTypeImage;
28     attributeSet.title = @"My First Spotlight Search";
29     attributeSet.contentDescription = @"A sample Spotlight search item";
30     attributeSet.keywords = [NSArray arrayWithObjects:@"keyword1", @"keyword2", nil];
31     UIImage *image = [UIImage imageNamed:@"sampleImage"];
32     NSData *imageData = [NSData dataWithBytes:[image CGImage] length:[image CGImage].bytes];
33     attributeSet.thumbnailData = imageData;
34
35     CSSearchableItem *item = [[CSSearchableItem alloc] initWithAttributeSet:attributeSet
36                             domainIdentifier:@"spotlight.sampleItem"
37                             error);
38     if (!error)
39         [CSSearchableIndex defaultSearchableIndex addItem:item];
40 }

```

3. OK, teste den Index!

Kapitel 44: CoreImage-Filter

Examples

Beispiel für den Kernbildfilter

Ziel c

Melden Sie sich einfach an, wie Sie einen bestimmten Filter verwenden

```
NSArray *properties = [CIFilter filterNamesInCategory:kCICategoryBuiltIn];

for (NSString *filterName in properties)
{
    CIFilter *fltr = [CIFilter filterWithName:filterName];
    NSLog(@"%@", [fltr attributes]);
}
```

Im Falle von CISepiaTone lautet das Systemprotokoll wie folgt

```
CIAttributeFilterDisplayName = "Sepia Tone";
CIAttributeFilterName = CISepiaTone;
CIAttributeReferenceDocumentation = "http://developer.apple.com/cgi-
bin/apple_ref.cgi?apple_ref=//apple_ref/doc/filter/ci/CISepiaTone";
inputImage =
{
    CIAttributeClass = CIImage;
    CIAttributeDescription = "The image to use as an input image. For filters that also
use a background image, this is the foreground image.";
    CIAttributeDisplayName = Image;
    CIAttributeType = CIAttributeTypeImage;
};
inputIntensity =
{
    CIAttributeClass = NSNumber;
    CIAttributeDefault = 1;
    CIAttributeDescription = "The intensity of the sepia effect. A value of 1.0 creates a
monochrome sepia image. A value of 0.0 has no effect on the image.";
    CIAttributeDisplayName = Intensity;
    CIAttributeIdentity = 0;
    CIAttributeMin = 0;
    CIAttributeSliderMax = 1;
    CIAttributeSliderMin = 0;
    CIAttributeType = CIAttributeTypeScalar;
};
}
```

Unter Verwendung des obigen Systemprotokolls richten wir den Filter wie folgt ein:

```
CIImage *beginImage = [CIImage imageWithCGImage:[myImageView.image CGImage]];
CIContext *context = [CIContext contextWithOptions:nil];
//select Filter Name and Intensity
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
kCIInputImageKey, beginImage, @"inputIntensity", [NSNumber numberWithInt:0.8], nil];
CIImage *outputImage = [filter outputImage];
```

```

CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
UIImage *newImg = [UIImage imageWithCGImage:cgimg];

[myImageView1 setImage:newImg];

CGImageRelease(cgimg);

```

Mit dem obigen Code erzeugtes Bild



Eine alternative Methode zum Einrichten eines Filters

```

UIImageView *imageView1=[[UIImageView alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, self.view.frame.size.height/2)];
UIImageView *imageView2=[[UIImageView alloc] initWithFrame:CGRectMake(0,
self.view.frame.size.height/2, self.view.frame.size.width, self.view.frame.size.height/2)];
imageView1.image=[UIImage imageNamed:@"image.png"];

CIImage *beginImage = [CIImage imageWithCGImage:[imageView1.image CGImage]];
CIText *context = [CIText contextWithOptions:nil];
//select Filter Name and Intensity

CIFilter *filter = [CIFilter filterWithName:@"CIColorPosterize"];

```

```

[filter setValue:beginImage forKey:kCIInputImageKey];
[filter setValue:[NSNumber numberWithFloat:8.0] forKey:@"inputLevels"];
CIImage *outputImage = [filter outputImage];

CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
UIImage *newImg = [UIImage imageWithCGImage:cgimg];

[imageView2 setImage:newImg];

CGImageRelease(cgimg);
[self.view addSubview:imageView1];
[self.view addSubview:imageView2];

```

Bild aus diesem Code generiert



Alle verfügbaren Filter sind wie folgt

```

/* CIAccordionFoldTransition,
   CIAdditionCompositing,
   CIAffineClamp,
   CIAffineTile,
   CIAffineTransform,
   CIColorAverage,
   CIColorHistogram,
   CIColorMaximum,
   CIColorMaximumAlpha,
   CIColorMinimum,
   CIColorMinimumAlpha,
   CIAztecCodeGenerator,

```

CIBarsSwipeTransition,
CIBlendWithAlphaMask,
CIBlendWithMask,
CIBloom,
CIBoxBlur,
CIBumpDistortion,
CIBumpDistortionLinear,
CICheckerboardGenerator,
CICircleSplashDistortion,
CICircularScreen,
CICircularWrap,
CICMYKHalftone,
CICode128BarcodeGenerator,
CIColorBlendMode,
CIColorBurnBlendMode,
CIColorClamp,
CIColorControls,
CIColorCrossPolynomial,
CIColorCube,
CIColorCubeWithColorSpace,
CIColorDodgeBlendMode,
CIColorInvert,
CIColorMap,
CIColorMatrix,
CIColorMonochrome,
CIColorPolynomial,
CIColorPosterize,
CIColumnAverage,
CIComicEffect,
CIConstantColorGenerator,
CIConvolution3X3,
CIConvolution5X5,
CIConvolution7X7,
CIConvolution9Horizontal,
CIConvolution9Vertical,
CICopyMachineTransition,
CICrop,
CICrystallize,
CIDarkenBlendMode,
CIDepthOfField,
CIDifferenceBlendMode,
CIDiscBlur,
CIDisintegrateWithMaskTransition,
CIDisplacementDistortion,
CIDissolveTransition,
CIDivideBlendMode,
CIDotScreen,
CIDroste,
CIEdges,
CIEdgeWork,
CIEightfoldReflectedTile,
CIExclusionBlendMode,
CIExposureAdjust,
CIFalseColor,
CIFlashTransition,
CIFourfoldReflectedTile,
CIFourfoldRotatedTile,
CIFourfoldTranslatedTile,
CIGammaAdjust,
CIGaussianBlur,
CIGaussianGradient,

CIGlassDistortion,
CIGlassLozenge,
CIGlideReflectedTile,
CIGloom,
CIHardLightBlendMode,
CIHatchedScreen,
CIHeightFieldFromMask,
CIHexagonalPixellate,
CIHighlightShadowAdjust,
CIHistogramDisplayFilter,
CIHoleDistortion,
CIHueAdjust,
CIHueBlendMode,
CIKaleidoscope,
CILanczosScaleTransform,
CILenticularHaloGenerator,
CILightenBlendMode,
CILightTunnel,
CILinearBurnBlendMode,
CILinearDodgeBlendMode,
CILinearGradient,
CILinearToSRGBToneCurve,
CILineOverlay,
CILineScreen,
CILuminosityBlendMode,
CIMaskedVariableBlur,
CIMaskToAlpha,
CIMaximumComponent,
CIMaximumCompositing,
CIMedianFilter,
CIMinimumComponent,
CIMinimumCompositing,
CIModTransition,
CIMotionBlur,
CIMultiplyBlendMode,
CIMultiplyCompositing,
CINoiseReduction,
CIOpTile,
CIOverlayBlendMode,
CIPageCurlTransition,
CIPageCurlWithShadowTransition,
CIParallelogramTile,
CIPDF417BarcodeGenerator,
CIPerspectiveCorrection,
CIPerspectiveTile,
CIPerspectiveTransform,
CIPerspectiveTransformWithExtent,
CIPhotoEffectChrome,
CIPhotoEffectFade,
CIPhotoEffectInstant,
CIPhotoEffectMono,
CIPhotoEffectNoir,
CIPhotoEffectProcess,
CIPhotoEffectTonal,
CIPhotoEffectTransfer,
CIPinchDistortion,
CIPinLightBlendMode,
CIPixellate,
CIPointillize,
CIQRCodeGenerator,
CIRadialGradient,

```
CIRandomGenerator,  
CIRippleTransition,  
CIRowAverage,  
CISaturationBlendMode,  
CIScreenBlendMode,  
CISepiaTone,  
CIShadedMaterial,  
CISharpenLuminance,  
CISixfoldReflectedTile,  
CISixfoldRotatedTile,  
CISmoothLinearGradient,  
CISoftLightBlendMode,  
CISourceAtopCompositing,  
CISourceInCompositing,  
CISourceOutCompositing,  
CISourceOverCompositing,  
CISpotColor,  
CISpotLight,  
CISRGBToneCurveToLinear,  
CIStarShineGenerator,  
CIStraightenFilter,  
CISretchCrop,  
CIStripesGenerator,  
CISubtractBlendMode,  
CISunbeamsGenerator,  
CISwipeTransition,  
CITemperatureAndTint,  
CIToneCurve,  
CITorusLensDistortion,  
CITriangleKaleidoscope,  
CITriangleTile,  
CITwelvefoldReflectedTile,  
CITwirlDistortion,  
CIUnsharpMask,  
CIVibrance,  
CIVignette,  
CIVignetteEffect,  
CIVortexDistortion,  
CIWhitePointAdjust,  
CIZoomBlur*/
```

CoreImage-Filter online lesen: <https://riptutorial.com/de/ios/topic/7278/coreimage-filter>

Kapitel 45: CTCallCenter

Examples

Fangen Sie Anrufe von Ihrer App auch aus dem Hintergrund ab

Aus der Apple-Dokumentation:

Verwenden Sie die CTCallCenter-Klasse, um eine Liste der aktuellen Mobiltelefonanrufe abzurufen und auf Statusänderungen für Anrufe zu reagieren, z. B. von einem Wählzustand in einen verbundenen Zustand. Solche Zustandsänderungen werden als zellulare Anrufereignisse bezeichnet.

Der Zweck von CTCallCenter besteht darin, dem Entwickler die Möglichkeit zu geben, den Status seiner App während eines Anrufs zu unterbrechen, um dem Benutzer die beste Benutzererfahrung zu bieten.

Ziel c:

Zuerst definieren wir einen neuen Klassenmitglied innerhalb der Klasse, mit der wir die Interceptions behandeln möchten:

```
@property (atomic, strong) CTCallCenter *callCenter;
```

Innerhalb unserer Klasse init (Konstruktor) werden wir unserem Klassenmitglied neuen Speicher zuweisen:

```
[self setCallCenter:[CTCallCenter new]];
```

Anschließend rufen wir unsere neue Methode auf, die die Interceptions tatsächlich behandelt:

```
- (void)registerPhoneCallListener
{
    [[self callCenter] setCallEventHandler:^(CTCall * _Nonnull call) {
        NSLog(@"CallEventHandler called - interception in progress");

        if ([call.callState isEqualToString: CTCallStateConnected])
        {
            NSLog(@"Connected");
        }
        else if ([call.callState isEqualToString: CTCallStateDialing])
        {
            NSLog(@"Dialing");
        }
        else if ([call.callState isEqualToString: CTCallStateDisconnected])
        {
            NSLog(@"Disconnected");
        }
        } else if ([call.callState isEqualToString: CTCallStateIncoming])
        {

```



```
        NSLog(@"Incomming");
    }
}];
}
```

Wenn der Benutzer Ihre App verwendet und einen Anruf erhält, können Sie diesen Anruf unterbrechen und Ihre App für einen Sicherungsstatus bearbeiten.

Es ist erwähnenswert, dass es 4 Anrufzustände gibt, die Sie abfangen können:

```
CTCallStateDialing
CTCallStateIncoming
CTCallStateConnected
CTCallStateDisconnected
```

Schnell:

Definieren Sie Ihren Teilnehmer in der betreffenden Klasse und definieren Sie ihn:

```
self.callCenter = CTCallCenter()
self.callCenter.callEventHandler = { call in
    // Handle your interception
    if call.callState == CTCallStateConnected
    {
    }
}
```

Was passiert, wenn sich Ihre App im Hintergrund befindet und Sie Anrufe abfangen müssen, während sich die App im Hintergrund befindet?

Wenn Sie beispielsweise eine **Unternehmensanwendung entwickeln**, können Sie auf der Registerkarte "Funktionen" im Wesentlichen nur zwei Funktionen (VOIP und Hintergrundabruf) hinzufügen:

Ihr Projektziel -> Funktionen -> Hintergrundmodi -> Markieren Sie Voice over IP und Hintergrundabruf

CallKit - ios 10

```
//Header File
<CallKit/CXCallObserver.h>
CXCallObserver *callObserver = [[CXCallObserver alloc] init];
// If queue is nil, then callbacks will be performed on main queue
[callObserver setDelegate:self queue:nil];
// Don't forget to store reference to callObserver, to prevent it from being released
self.callObserver = callObserver;
```

```
// get call status
- (void)callObserver:(CXCallObserver *)callObserver callChanged:(CXCall *)call {
    if (call.hasConnected) {
        // perform necessary actions
    }
}
```

CTCallCenter online lesen: <https://riptutorial.com/de/ios/topic/3007/ctcallcenter>

Kapitel 46: CydiaSubstrate Tweak

Einführung

Erfahren Sie, wie Sie Cydia-Substrat-Anpassungen für jailbroken iPhones erstellen.

Mit diesen Optimierungen können Sie das Verhalten des Betriebssystems ändern, um so zu arbeiten, wie Sie es möchten.

Bemerkungen

Theos installieren

<https://github.com/theos/theos/wiki/Installation>

Examples

Erstellen Sie einen neuen Tweak mit Theos

Verwenden Sie nic, um ein neues Projekt zu erstellen

Geben Sie diesen Befehl in Ihr Terminal ein

```
$THEOS/bin/nic.pl
```

```
NIC 2.0 - New Instance Creator
-----
[1.] iphone/activator_event
[2.] iphone/application_modern
[3.] iphone/cydget
[4.] iphone/flipswitch_switch
[5.] iphone/framework
[6.] iphone/ios7_notification_center_widget
[7.] iphone/library
[8.] iphone/notification_center_widget
[9.] iphone/preference_bundle_modern
[10.] iphone/tool
[11.] iphone/tweak
[12.] iphone/xpc_service
Choose a Template (required):
```

Wählen Sie die Vorlage [11.] iphone/tweak

Füllen Sie die Details aus und Sie erhalten folgende Dateien:

```
-rw-r--r--@ 1 gkpln3 staff 214B Jun 12 15:09 Makefile
-rw-r--r--@ 1 gkpln3 staff 89B Jun 11 22:58 TorchonFocus.plist
-rw-r--r-- 1 gkpln3 staff 2.7K Jun 12 16:10 Tweak.xm
-rw-r--r-- 1 gkpln3 staff 224B Jun 11 16:17 control
drwxr-xr-x 3 gkpln3 staff 102B Jun 11 16:18 obj
drwxr-xr-x 16 gkpln3 staff 544B Jun 12 16:12 packages
```

Überschreibe die Methode zum Speichern von iOS-Screenshots

Öffnen Sie die Datei `Tweak.xm` mit Ihrem bevorzugten Code-Editor.

haken Sie sich an eine bestimmte Methode aus dem Betriebssystem.

```
%hook SBScreenShotter
- (void)saveScreenshot:(BOOL)screenshot
{
    %orig;
    NSLog(@"saveScreenshot: is called");
}
%end
```

Beachten Sie, dass Sie wählen können, ob die ursprüngliche Funktion aufgerufen werden soll oder nicht, zum Beispiel:

```
%hook SBScreenShotter
- (void)saveScreenshot:(BOOL)screenshot
{
    NSLog(@"saveScreenshot: is called");
}
%end
```

überschreibt die Funktion, ohne die ursprüngliche aufzurufen, sodass die Screenshots des Gehäuses nicht gespeichert werden.

CydiaSubstrate Tweak online lesen: <https://riptutorial.com/de/ios/topic/10533/cydiastrate-tweak>

Kapitel 47: Daten zwischen View Controllern übergeben (mit MessageBox-Konzept)

Einführung

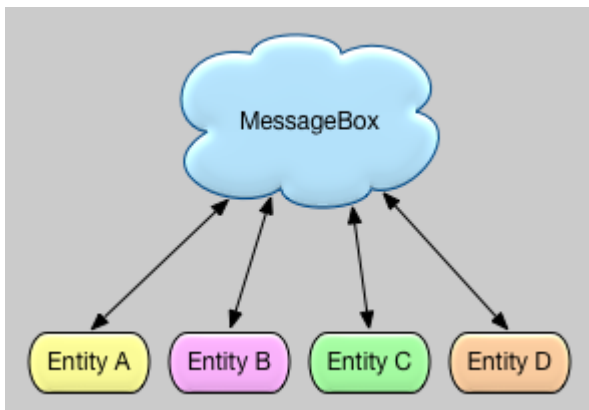
MessageBox ist ein einfaches Konzept zur Entkopplung von Entitäten.

Entität A kann zum Beispiel eine Nachricht platzieren, die Entität B lesen kann, wann immer dies geeignet ist.

Ein View-Controller möchte mit einem anderen View-Controller sprechen, aber Sie möchten keine starke oder schwache Beziehung herstellen.

Examples

Einfache Beispielnutzung



```
let messageBox:MessageBox = MessageBox ()

// set
messageBox.setObject ("TestObject1", forKey:"TestKey1")

// get
// but don't remove it, keep it stored, so that it can still be retrieved later
let someObject:String = messageBox.getObject (forKey:"TestKey1", removeIfFound:false)

// get
// and remove it
let someObject:String = messageBox.getObject (forKey:"TestKey1", removeIfFound:true)
```

Daten zwischen View Controllern übergeben (mit MessageBox-Konzept) online lesen:
<https://riptutorial.com/de/ios/topic/9118/daten-zwischen-view-controllern-ubergeben--mit-messagebox-konzept->

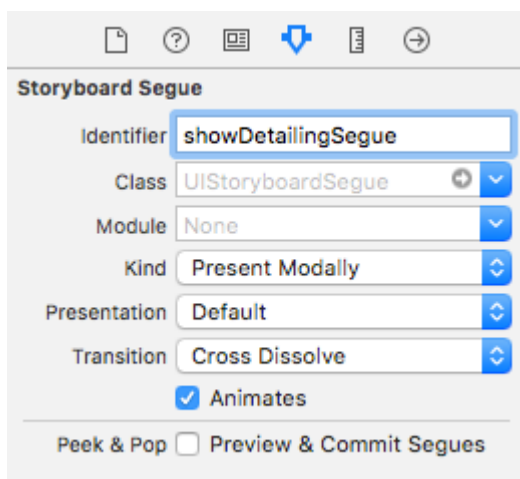
Kapitel 48: Daten zwischen View-Controllern übergeben

Examples

Segmente verwenden (Daten weiterleiten)

Um Daten vom aktuellen View-Controller über Segues an den nächsten neuen View-Controller (nicht einen vorherigen View-Controller) zu übergeben, erstellen Sie zunächst ein Segment mit einem Bezeichner im entsprechenden Storyboard. Überschreiben Sie die `prepareForSegue` Methode Ihres aktuellen View-Controllers. Überprüfen Sie innerhalb der Methode nach dem Segment, das Sie gerade anhand des Bezeichners erstellt haben. Konvertieren Sie den Ziel-View-Controller und übergeben Sie Daten an ihn, indem Sie die Eigenschaften für den Downcast-View-Controller festlegen.

Festlegen eines Bezeichners für ein Segment:



Segmente können programmgesteuert oder mithilfe von Schaltflächenaktionsereignissen ausgeführt werden, die im Storyboard durch Drücken von Strg + Ziehen zum Zielansicht-Controller festgelegt werden. Sie können bei Bedarf ein Segment programmatisch aufrufen, indem Sie die Segment-ID im View-Controller verwenden:

Ziel c

```
- (void)showDetail {
    [self performSegueWithIdentifier:@"showDetailingSegue" sender:self];
}
```

Schnell

```
func showDetail() {
    self.performSegue(withIdentifier: "showDetailingSegue", sender: self)
}
```

In der überschriebenen Version der `prepareForSegue` Methode können Sie die `prepareForSegue` . Sie können die erforderlichen Eigenschaften festlegen, bevor der Zielansicht-Controller geladen wird.

Ziel c

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"showDetailingSegue"]){
        DetailViewController *controller = (DetailViewController
*)segue.destinationViewController;
        controller.isDetailingEnabled = YES;
    }
}
```

Schnell

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showDetailingSegue" {
        let controller = segue.destinationViewController as! DetailViewController
        controller.isDetailingEnabled = true
    }
}
```

`DetailViewController` ist der Name des zweiten View-Controllers und `isDetailingEnabled` ist eine öffentliche Variable in diesem View-Controller.

Um dieses Muster zu erweitern, können Sie eine öffentliche Methode in `DetailViewController` als Pseudo-Initialisierer behandeln, um die erforderlichen Variablen zu initialisieren. Dies dokumentiert Variablen, die in `DetailViewController` werden müssen, ohne dass der Quellcode gelesen werden muss. Es ist auch ein praktischer Ort, um Standardwerte festzulegen.

Ziel c

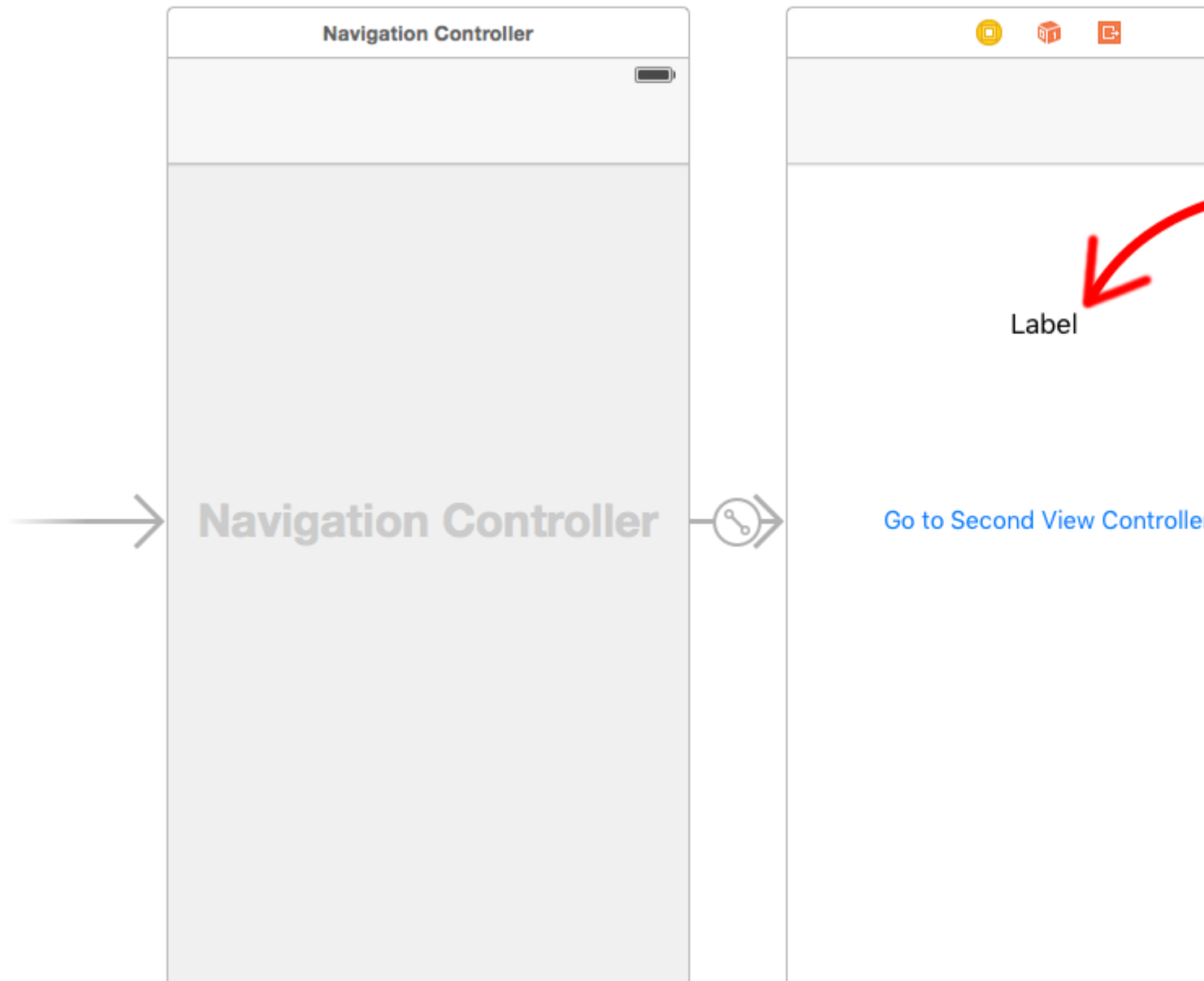
```
- (void)initVC:(BOOL *)isDetailingEnabled {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Schnell

```
func initVC(isDetailingEnabled: Bool) {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Verwenden des Delegatenmusters (Zurückgeben von Daten)

Um Daten vom aktuellen View-Controller zurück an den vorherigen View-Controller zu übergeben, können Sie das Delegatenmuster verwenden.



In diesem Beispiel wird davon `showSecondViewController` dass Sie im Interface Builder ein `showSecondViewController` und dass Sie die `showSecondViewController` auf `showSecondViewController`. Die Ausgänge und Aktionen müssen auch mit den Namen im folgenden Code verbunden sein.

Erste Ansicht Controller

Der Code für den First View Controller lautet

Schnell

```
class FirstViewController: UIViewController, DataEnteredDelegate {

    @IBOutlet weak var label: UILabel!

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "showSecondViewController", let secondViewController =
        segue.destinationViewController as? SecondViewController {
            secondViewController.delegate = self
        }
    }
}
```



```

    }
}

// required method of our custom DataEnteredDelegate protocol
func userDidEnterInformation(info: String) {
    label.text = info
    navigationController?.popViewControllerAnimated(true)
}
}

```

Ziel c

```

@interface FirstViewController : UIViewController <DataEnteredDelegate>
@property (weak, nonatomic) IBOutlet UILabel *label;
@end

@implementation FirstViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    SecondViewController *secondViewController = segue.destinationViewController;
    secondViewController.delegate = self;
}
-(void)userDidEnterInformation:(NSString *)info {
    _label.text = info
    [self.navigationController popViewControllerAnimated:YES];
}
@end

```

Beachten Sie die Verwendung unseres benutzerdefinierten `DataEnteredDelegate` Protokolls.

Second View Controller und Protokoll

Der Code für den zweiten View-Controller lautet

Schnell

```

// protocol used for sending data back
protocol DataEnteredDelegate: class {
    func userDidEnterInformation(info: String)
}

class SecondViewController: UIViewController {

    // making this a weak variable so that it won't create a strong reference cycle
    weak var delegate: DataEnteredDelegate?

    @IBOutlet weak var textField: UITextField!

    @IBAction func sendTextBackButton(sender: AnyObject) {

        // call this method on whichever class implements our delegate protocol (the first
        view controller)
    }
}

```

```
        delegate?.userDidEnterInformation(textField.text ?? "")
    }
}
```

Ziel c

```
@protocol DataEnteredDelegate <NSObject>
- (void)userDidEnterInformation:(NSString *)info;
@end

@interface SecondViewController : UIViewController
@property (nonatomic) id <DataEnteredDelegate> delegate;
@property (weak, nonatomic) IBOutlet UITextField *textField;
@end

@implementation SecondViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}

- (IBAction) sendTextBackButton:(id)sender{
    [_delegate userDidEnterInformation:textField.text];
}
@end
```

Beachten Sie, dass sich das `protocol` außerhalb der View Controller-Klasse befindet.

Daten rückwärts mit "Abwickeln" umleiten

Im Gegensatz zu segue können Sie Daten vom aktuellen View-Controller an den Ziel-View-Controller "weiterleiten":

(VC1) -> (VC2)

Mit "Abwickeln" können Sie das Gegenteil tun, Daten vom Ziel- oder aktuellen Ansichts-Controller an den präsentierenden Ansichts-Controller übergeben:

(VC1) <- (VC2)

ANMERKUNG : Achten Sie darauf, dass Sie mit der Funktion "Abwickeln" die Daten zuerst weitergeben können. Danach wird der aktuelle View Controller (VC2) freigegeben.

So geht's:

Zuerst müssen Sie die folgende Deklaration bei dem Presenting View Controller (VC1) hinzufügen. Dies ist der View Controller, an den die Daten übergeben werden sollen:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
```

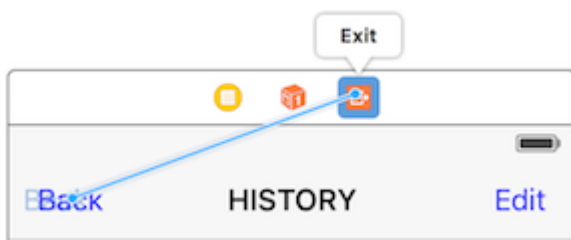
Das Wichtigste ist , das Präfix zu verwenden , `unwind` , um diesen „informiert“ Xcode , dass dies eine Abroll - Methode Ihnen die Möglichkeit, es auch in Storyboard zu verwenden.

Danach müssen Sie die Methode implementieren, sie sieht fast wie ein tatsächliches Segment aus:

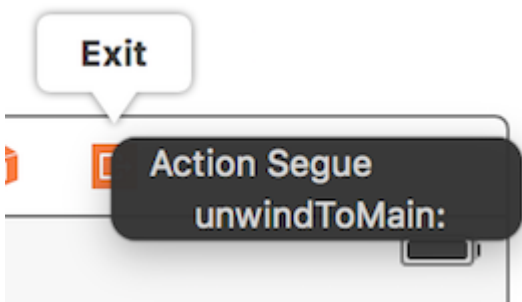
```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
{
    if segue.identifier == "YourCustomIdentifier"
    {
        if let VC2 = segue.sourceViewController as? VC2
        {
            // Your custom code in here to access VC2 class member
        }
    }
}
```

Jetzt haben Sie zwei Möglichkeiten, die Abwicklungsaufrufe aufzurufen:

1. Sie können "hard code" aufrufen: `self.performSegueWithIdentifier("YourCustomIdentifier", sender: self)` der die Abwicklung für Sie `performSegueWithIdentifier` wenn `performSegueWithIdentifier` .
2. Sie können die Abwickelmethode mithilfe des `storyboard` mit dem Objekt "Beenden" verknüpfen: Halten Sie die Taste gedrückt, und ziehen Sie die Schaltfläche, die Sie die Abwickelmethode aufrufen möchten, auf das Objekt "Beenden".



Wenn Sie loslassen, können Sie Ihre benutzerdefinierte Abwickelmethode wählen:



Weitergabe von Daten mithilfe von Closures (Zurückgeben von Daten)

Anstatt das **Delegatenmuster zu verwenden** , das die Implementierung in verschiedene Teile der `UIViewController` Klasse `UIViewController` , können Sie sogar `closures` , um Daten vor und zurück zu übergeben. Wenn Sie davon ausgehen, dass Sie `UIStoryboardSegue` , können Sie in der `prepareForSegue` Methode den neuen Controller einfach in einem Schritt `prepareForSegue`

```
final class DestinationViewController: UIViewController {
    var onComplete: ((success: Bool) -> ())?

    @IBAction func someButtonTapped(sender: AnyObject?) {
        onComplete?(success: true)
    }
}
```

```

    }
}

final class MyViewController: UIViewController {
    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

        guard let destinationController = segue.destinationViewController as?
        DestinationViewController else { return }

        destinationController.onCompletion = { success in
            // this will be executed when `someButtonTapped(_)` will be called
            print(success)
        }
    }
}
}

```

Dies ist ein Beispiel für die Verwendung und es ist besser für Swift zu verwenden. Die Syntax des Objective-C-Blocks ist nicht so einfach, um den Code lesbarer zu machen

Callback-Schließung (Block) verwenden, um Daten zurückzuleiten

Dieses Thema ist ein klassisches Thema in der iOS-Entwicklung, und seine Lösung unterscheidet sich von anderen bereits gezeigten Beispielen. In diesem Beispiel werde ich zeige eine weitere tägliche gemeinsame Nutzung ein: Übergabe von Daten unter Verwendung von `closure` durch Anpassung `delegate pattern` auf dieser Seite in `Callback - Beispiel closure` !

eine Sache , diese Methode überlegen ist `delegate pattern` ist anstelle von Split die Einrichtung Code in zwei anderen Ort (auf dieser Seite an Delegationen Beispiel aussehen, `prepareForSegue` , `userDidEnterInformation`) eher sie zusammen zu sammeln (nur in `prepareForSegue` , ich werde es zeigen)

Starten Sie vom Second View Controller aus

Wir müssen herausfinden, wie Callback verwendet wird. Dann können wir es schreiben. Deshalb starten wir vom Second-View-Controller, da wir Callback verwenden. Wenn wir die neue Texteingabe erhalten haben, rufen wir unseren Callback auf und **verwenden den Callback-Parameter** als Medium Um Daten an den ersten ViewController zurückzuschicken, muss ich feststellen, dass ich mit dem Callback-Parameter gesagt habe. Dies ist sehr wichtig. Neulinge (wie ich es waren) übersehen dies immer und wissen nicht, wo sie den Callback-Abschluss richtig schreiben sollen

In diesem Fall wissen wir also, dass unser Rückruf nur einen Parameter enthält: Text und dessen Typ ist `String` . Lassen Sie uns ihn deklarieren und als Eigenschaft definieren, da wir den ersten View-Controller mit Daten füllen müssen

Ich kommentiere einfach alle `delegate` und halte es für Vergleiche

```

class SecondViewController: UIViewController {

    //weak var delegate: DataEnteredDelegate? = nil
    var callback: ((String?)->())?

```

```

@IBOutlet weak var textField: UITextField!

@IBAction func sendTextBackButton(sender: AnyObject) {

    //delegate?.userDidEnterInformation(textField.text!)
    callback?(input.text)

    self.navigationController?.popViewControllerAnimated(true)
}
}

```

Beenden Sie den Controller für die erste Ansicht

Sie müssen nur noch den Callback-Abschluss übergeben, und wir sind fertig, der Abschluss wird die zukünftige Arbeit für uns erledigen, da wir ihn bereits im Second-View-Controller eingerichtet haben

Sehen Sie, wie unser Code im Vergleich zum `delegate pattern` **kürzer** `delegate pattern`

```

//no more DataEnteredDelegate
class FirstViewController: UIViewController {

    @IBOutlet weak var label: UILabel!

    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        if segue.identifier == "showSecondViewController" {
            let secondViewController = segue.destinationViewController as!
SecondViewController
            //secondViewController.delegate = self
            secondViewController.callback = { text in self.label.text = text }
        }
    }

    // required method of our custom DataEnteredDelegate protocol
    //func userDidEnterInformation(info: String) {
    //    label.text = info
    //}
}

```

und in letzter Konsequenz wird vielleicht jemand von Ihnen verwirrt, wenn wir nur die Daten (Schließung in diesem Fall) nur auf eine Weise weitergeben, vom ersten Ansichts-Controller zum zweiten, kein direktes Zurückkommen vom zweiten Ansichts-Controller, wie können wir das tun betrachten es als Kommunikationsmittel? Vielleicht sollten Sie es wirklich ausführen und es selbst beweisen. Alles, was ich sagen werde, ist es **Parameter**, es ist **der Parameter** des **Callback-Abschlusses**, der Daten zurückgibt!

Durch Zuweisen von Eigenschaften (Weiterleiten von Daten)

Sie können Daten direkt übergeben, indem Sie die Eigenschaft des nächsten View-Controllers zuweisen, bevor Sie sie drücken oder präsentieren.

```

class FirstViewController: UIViewController {

    func openSecondViewController() {

```

```
// Here we initialize SecondViewController and set the id property to 492
let secondViewController = SecondViewController()
secondViewController.id = 492

// Once it was assign we now push or present the view controller
present(secondViewController, animated: true, completion: nil)
}

}

class SecondViewController: UIViewController {

    var id: Int?

    override func viewDidLoad() {
        super.viewDidLoad()

        // Here we unwrapped the id and will get the data from the previous view controller.
        if let id = id {
            print("Id was set: \(id)")
        }
    }
}
```

Daten zwischen View-Controllern übergeben online lesen:

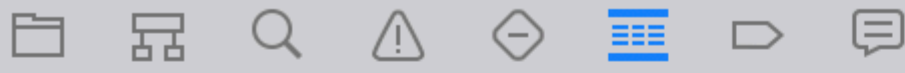
<https://riptutorial.com/de/ios/topic/434/daten-zwischen-view-controllern-ubergeben>

Kapitel 49: Debuggen von Abstürzen

Examples

Informationen zu einem Absturz suchen

Wenn Ihre App abstürzt, öffnet Xcode den Debugger und zeigt Ihnen weitere Informationen zum Absturz:



test PID 12093

- CPU 0%
- Memory 1.3 MB
- Energy Impact Zero
- Disk Zero KB/s
- Network Zero KB/s
- Thread 1 Queue: com....hread (serial)
 - 0 __pthread_kill
 - 10 +[NSArray arrayWithObject:]
 - 11 main**
 - 12 start
 - 13 start



```

1 //
2 // main.m
3 // test
4 //
5 // Created
6 // Copyright
7 //
8
9 #import <Fou
10
11 int main(int
12 {
13     @autorel
14         id o
15     NSLo
16 }
17     return 0
18 }
19
20

```


Eingabetaste, werden Sie eine Textdarstellung der Stack - Trace, die Sie kopieren und einfügen können:

```
(lldb) bt
* thread #1: tid = 0x3aaec5, 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10,
queue = 'com.apple.main-thread', stop reason = signal SIGABRT
  frame #0: 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10
  frame #1: 0x000000010008142d libsystem_pthread.dylib`pthread_kill + 90
  frame #2: 0x00007fff96dc76e7 libsystem_c.dylib`abort + 129
  frame #3: 0x00007fff8973bf81 libc++abi.dylib`abort_message + 257
  frame #4: 0x00007fff89761a47 libc++abi.dylib`default_terminate_handler() + 267
  frame #5: 0x00007fff94f636ae libobjc.A.dylib`_objc_terminate() + 103
  frame #6: 0x00007fff8975f19e libc++abi.dylib`std::__terminate(void (*)()) + 8
  frame #7: 0x00007fff8975ec12 libc++abi.dylib`__cxa_throw + 121
  frame #8: 0x00007fff94f6108c libobjc.A.dylib`objc_exception_throw + 318
  frame #9: 0x00007fff8d067372 CoreFoundation`-[__NSPlaceholderArray initWithObjects:count:]
+ 290
  frame #10: 0x00007fff8d0eaa1f CoreFoundation`+[NSArray arrayWithObject:] + 47
* frame #11: 0x0000000100001b54 test`main(argc=1, argv=0x00007fff5fbff808) + 68 at main.m:15
  frame #12: 0x00007fff8bea05ad libdyld.dylib`start + 1
  frame #13: 0x00007fff8bea05ad libdyld.dylib`start + 1
```

Das Debuggen von SIGABRT und EXC_BAD_INSTRUCTION stürzt ab

Ein SIGABRT oder eine EXC_BAD_INSTRUCTION bedeutet normalerweise, dass die App absichtlich abgestürzt ist, da einige Prüfungen fehlgeschlagen sind. Diese sollten eine Nachricht mit weiteren Informationen an die Debugger-Konsole protokollieren. Weitere Informationen finden Sie dort.

Viele SIGABRT werden durch nicht SIGABRT Objective-C-Ausnahmen verursacht. Es gibt *vielen* Gründe, aus denen Ausnahmen ausgelöst werden können, und es werden *immer* viele hilfreiche Informationen auf der Konsole protokolliert.

- `NSInvalidArgumentException`, was bedeutet, dass die App ein ungültiges Argument an eine Methode übergeben hat
- `NSRangeException`, was bedeutet, dass die App versucht hat, auf einen Index außerhalb des `NSArray NSString` eines Objekts `NSArray NSString`
- `NSInternalInconsistencyException` bedeutet, dass ein Objekt festgestellt hat, dass es sich in einem unerwarteten Zustand `NSInternalInconsistencyException`.
- `NSUnknownKeyException` bedeutet normalerweise, dass in einer XIB eine schlechte Verbindung besteht. Versuchen Sie einige Antworten auf [diese Frage](#).

Debuggen von EXC_BAD_ACCESS

EXC_BAD_ACCESS bedeutet, dass der Prozess versucht hat, auf einen ungültigen Zugriff auf den Speicher zuzugreifen, z. B. der Referenzierung eines `NULL` Zeigers oder das Schreiben in den Nur-Lese-Speicher. Dies ist die schwierigste Art des Absturzes beim Debuggen, da normalerweise keine Fehlermeldung angezeigt wird und einige Abstürze *sehr* schwer reproduzierbar sind und / oder in Code auftreten, der nicht mit dem Problem zusammenhängt. Dieser Fehler tritt in Swift sehr selten auf. Wenn er jedoch auftritt, können Abstürze durch das Reduzieren der Compiler-Optimierungen oft leichter zu debuggen sein.

Die meisten `EXC_BAD_ACCESS` Fehler werden durch den Versuch verursacht, einen `NULL` Zeiger zu dereferenzieren. Wenn dies der Fall ist, ist die im roten Pfeil angegebene Adresse normalerweise eine Hexadezimalzahl, die niedriger als eine normale Speicheradresse ist, häufig `0x0`. Setzen Sie Haltepunkte im Debugger oder fügen Sie gelegentlich `printf` / `NSLog` Anweisungen hinzu, um herauszufinden, warum der Zeiger `NULL`.

Ein `EXC_BAD_ACCESS`, der weniger zuverlässig auftritt oder überhaupt keinen Sinn `EXC_BAD_ACCESS`, könnte das Ergebnis eines Speicherverwaltungsproblems sein. Häufige Probleme, die dies verursachen können, sind:

- Verwendung von Speicher, der freigegeben wurde
- Der Versuch, über das Ende eines C-Arrays oder eines anderen Puffers zu schreiben
- Verwendung eines Zeigers, der nicht initialisiert wurde

Xcode enthält im Diagnoseabschnitt des Schema-Editors einige nützliche Tools zum Beheben von Speicherproblemen:



test > My Mac



test > My Mac

Build
1 target

Run
Debug

Test
Debug

Profile
Release

Analyze
Debug

Archive
Release

No Debug Session

<https://riptutorial.com/de/ios/topic/4745/debuggen-von-absturzen>

Kapitel 50: Deep Linking in iOS

Bemerkungen

Nützliche [Apple-Dokumentation](#) mit Beispielen und Erläuterungen.

Examples

Öffnen einer App basierend auf ihrem URL-Schema

So öffnen Sie eine App mit definiertem URL-Schema `todolist://` :

Ziel c

```
NSURL *myURL = [NSURL URLWithString:@"todolist://there/is/something/to/do"];
[[UIApplication sharedApplication] openURL:myURL];
```

Schnell

```
let urlString = "todolist://there/is/something/to/do"
if let url = NSURL(string: urlString) {
    UIApplication.shared().openURL(url)
}
```

HTML

```
<a href="todolist://there/is/something/to/do">New SMS Message</a>
```

Hinweis: Es ist hilfreich zu prüfen, ob der Link geöffnet werden kann, um dem Benutzer eine entsprechende Meldung anzuzeigen. Dies kann mit der Methode `canOpenURL:` werden.

Hinzufügen eines URL-Schemas zu Ihrer eigenen App

`MyTasks` , Sie arbeiten an einer App namens `MyTasks` und möchten, dass eingehende URLs eine neue Aufgabe mit einem Titel und einem `MyTasks` erstellen. Die von Ihnen gestaltete URL könnte etwa so aussehen:

```
mytasks://create?title=hello&body=world
```

(Natürlich ist der `text` und `body` werden Parameter verwendet , um unsere Aufgabe zu füllen , die wir schaffen!)

Hier sind die großen Schritte zum Hinzufügen dieses URL-Schemas zu Ihrem Projekt:

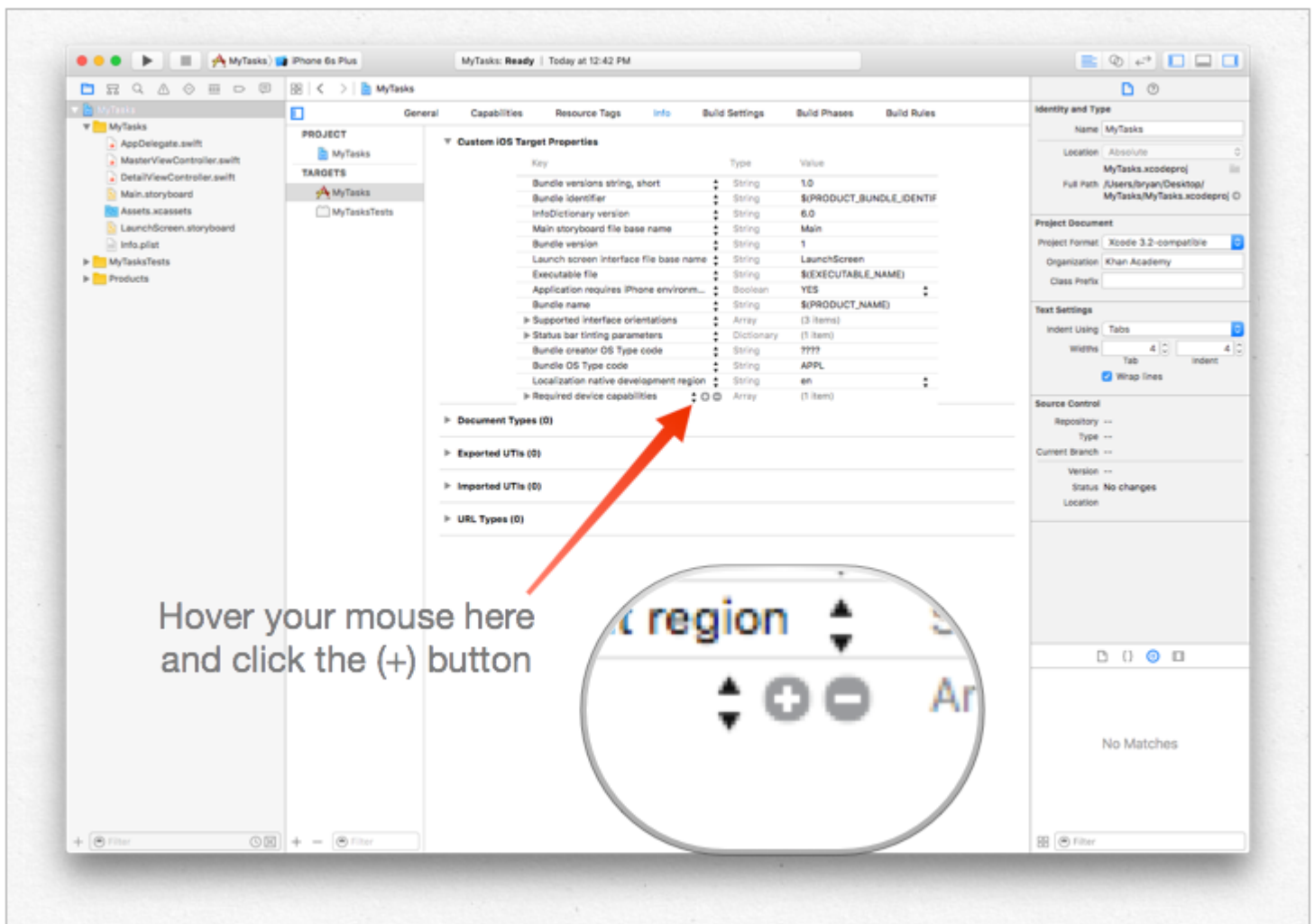
1. Registrieren Sie ein URL-Schema in der `Info.plist` Datei Ihrer App, damit das System weiß,

wann eine URL an Ihre App weitergeleitet wird.

2. Fügen Sie Ihrem `UIApplicationDelegate` eine Funktion `UIApplicationDelegate`, die eingehende URLs akzeptiert und verarbeitet.
3. Führen Sie die Aufgaben aus, die beim Öffnen dieser URL erforderlich sind.

Schritt 1: Registrieren Sie ein URL-Schema in Info.plist:

Zuerst müssen wir unserer Info.plist-Datei einen Eintrag "URL Types" hinzufügen. Klicken Sie hier auf die Schaltfläche (+):



Geben Sie dann eine eindeutige Kennung für Ihre App sowie das gewünschte URL-Schema ein. Sei präzise! Sie möchten nicht, dass das URL-Schema mit der Implementierung einer anderen App in Konflikt steht. Lieber hier zu lang sein als zu kurz!

▼ URL types	▲	Array	(5 items)
▼ Item 0	▲	Dictionary	(2 items)
URL identifier	▲	String	com.mycompany
▼ URL Schemes	▲	Array	(1 item)
Item 0	⊕ ⊖	String	mytasks
▼ Item 1	▲	Dictionary	(1 item)

Schritt 2: Behandeln Sie die URL in der UIApplicationDelegate

Wir müssen `application:openURL:options:` auf unserer `UIApplicationDelegate`. Wir prüfen die eingehende `URL` und sehen, ob es eine Aktion gibt, die wir ergreifen können!

Eine Implementierung wäre dies:

```
func application(app: UIApplication, openURL url: NSURL, options: [String : AnyObject]) -> Bool {
    if url.scheme == "mytasks" && url.host == "create" {
        let title = // get the title out of the URL's query using a method of your choice
        let body = // get the title out of the URL's query using a method of your choice
        self.rootViewController.createTaskWithTitle(title, body: body)
        return true
    }

    return false
}
```

Schritt 3: Führen Sie je nach URL eine Aufgabe aus.

Wenn ein Benutzer Ihre App über eine URL öffnet, hat er wahrscheinlich erwartet, dass *etwas* passiert. Vielleicht navigieren Sie zu einem Inhalt, vielleicht erstellen Sie einen neuen Artikel - in diesem Beispiel erstellen wir eine neue Aufgabe in der App!

Im obigen Code können wir einen Aufruf an `self.rootViewController.createTaskWithTitle(:body:)` sehen. `self.rootViewController.createTaskWithTitle(:body:)` also davon ausgehen, dass Ihr `AppDelegate` einen Zeiger auf den Root-View-Controller hat, der die Funktion ordnungsgemäß implementiert, sind Sie `AppDelegate` !

Einrichten eines Deeplinks für Ihre App

Das Einrichten von Deep-Links für Ihre App ist einfach. Sie benötigen lediglich eine kleine URL, über die Sie Ihre App öffnen möchten.

Folgen Sie den Schritten, um eine Verknüpfung für Ihre App einzurichten.

1. Ermöglicht das Erstellen eines Projekts und den Namen `DeepLinkPOC`.
2. Wählen Sie nun Ihr Projektziel aus.
3. Wählen Sie nach der Auswahl des Ziels die Registerkarte "Info" aus.
4. Scrollen Sie nach unten, bis eine Option für **URL-Typen angezeigt wird**

5. Klicken Sie auf die Option "+".

6. **URL-Schemata** fügen eine Zeichenfolge hinzu, mit der Sie Ihre App öffnen möchten. **Fügen Sie** in URL-Schemata " **DeepLinking** " hinzu.

Um Ihre App zu öffnen, können Sie sie starten, indem Sie "**DeepLinking: //**" in Ihre Safari eingeben. Ihre Deep-Linking-Zeichenfolge hat folgendes Format.

```
[scheme]://[host]/[path] --> DeepLinking://path/Page1
```

wo, Schema: "DeepLinking" Host: "Pfad" Pfad: "Seite1"

Hinweis : Auch wenn Host und Pfad nicht hinzugefügt werden, wird die App gestartet, sodass Sie sich keine Sorgen machen müssen. Sie können jedoch Host und Pfad hinzufügen, um nach dem Start der Anwendung zusätzlich zu einer bestimmten Seite umzuleiten.

7. Fügen Sie nun Ihrer Appdelegate folgende Methode hinzu.

Schnell:

```
func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?, annotation: AnyObject) -> Bool
```

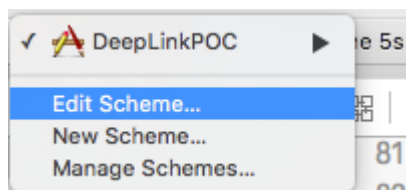
Ziel c:

```
-(BOOL)application:(UIApplication *)application  
    openURL:(NSURL *)url  
    sourceApplication:(NSString *)sourceApplication  
    annotation:(id)annotation
```

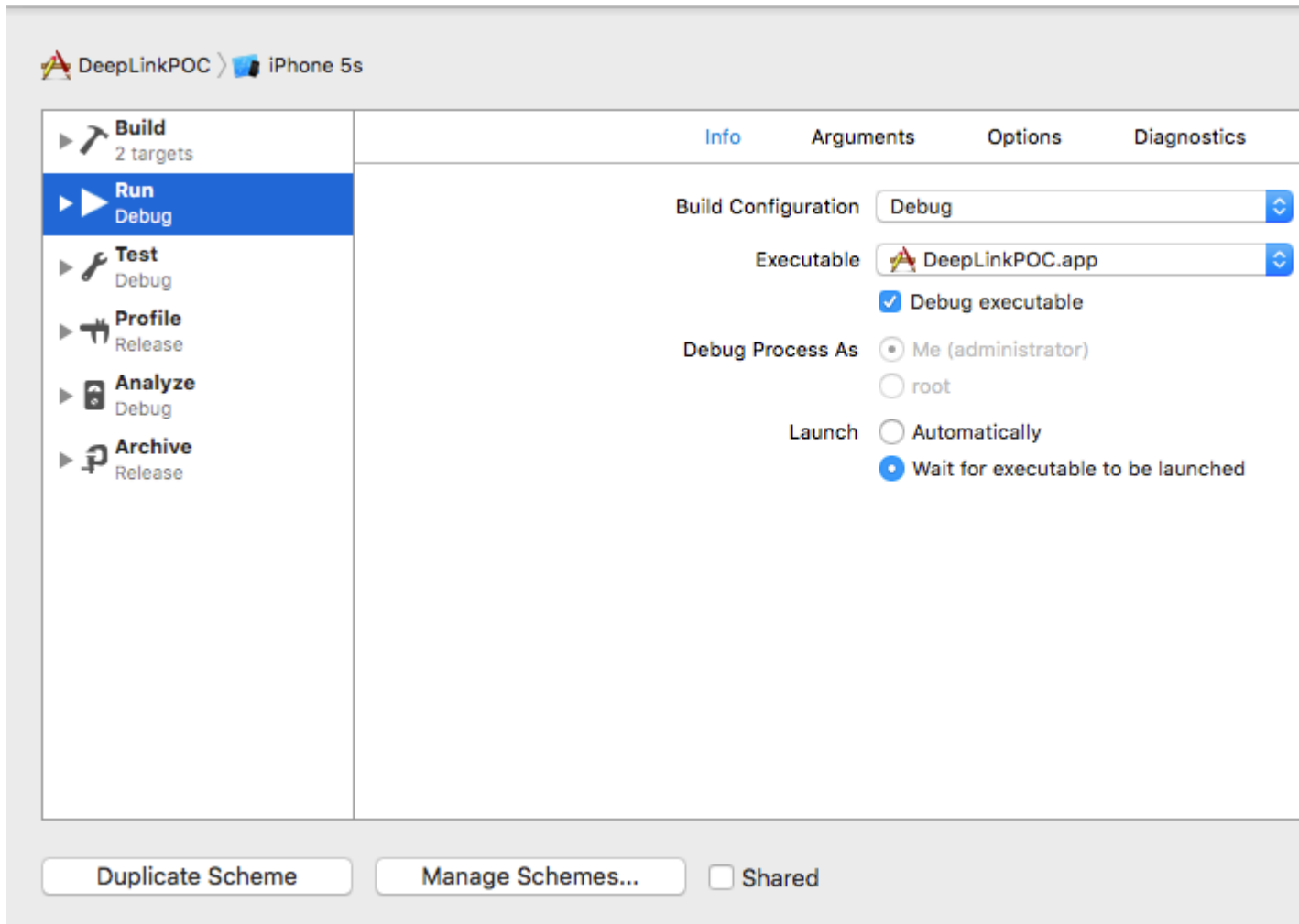
Die obige Methode wird aufgerufen, wenn Ihre App mit einer für Ihre App festgelegten Deep-Linking-Zeichenfolge gestartet wird.

8. Nun ist es an der Zeit, Ihre App zu installieren, aber warten Sie, bevor Sie direkt zur Schaltfläche "Ausführen" springen. Lassen Sie die App-Startmethode des Schemas geringfügig ändern.

- Wählen Sie Ihr Schema aus und bearbeiten Sie es als



- Startart ändern und schließen



9. Klicken Sie nun auf die Schaltfläche Ausführen (wenn Sie möchten, können Sie Ihrem `didFinishLaunchingWithOptions`- und `openURL`-Verfahren einen Haltepunkt hinzufügen, um die Werte zu überwachen.)
10. Sie erhalten eine Nachricht "Wartet auf den Start von DeepLinkPOC (oder den Namen Ihrer App)".
11. Öffnen Sie die Safari und geben Sie " **DeepLinking: //** " in die Suchleiste ein. **Daraufhin** erscheint die Aufforderung "Diese Seite in DeepLinkPOC öffnen". Klicken Sie auf Öffnen, um Ihre App zu starten.

Ich hoffe, Sie haben gelernt, wie Sie die Verknüpfung für Ihre App einrichten können :)

Deep Linking in iOS online lesen: <https://riptutorial.com/de/ios/topic/5173/deep-linking-in-ios>

Kapitel 51: DispatchGroup

Einführung

Verwandte Themen:

[Grand Central Dispatch](#)

[Parallelität](#)

Examples

Einführung

Angenommen, Sie haben mehrere Threads, die ausgeführt werden. Jeder Thread erledigt eine Aufgabe. Sie möchten entweder im mainThread ODER einem anderen Thread benachrichtigt werden, wenn alle Task-Threads abgeschlossen sind.

Die einfachste Lösung für ein solches Problem ist eine `DispatchGroup` .

Bei der Verwendung von `DispatchGroup` , für jede Anforderung, Sie `enter` die Gruppe und für jede abgeschlossene Anforderung, Sie `leave` die Gruppe.

Wenn keine Anforderungen mehr in der Gruppe vorhanden sind, werden Sie `notify` (benachrichtigt).

Verwendungszweck:

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let dispatchGroup = DispatchGroup() //Create a group for the tasks.
        let session: URLSession = URLSession.shared

        dispatchGroup.enter() //Enter the group for the first task.

        let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com"!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave() //Leave the group for the first task.
        }
    }
}
```

```

        dispatchGroup.enter() //Enter the group for the second task.

        let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca"!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave() //Leave the group for the second task.
        }

        //Get notified on the main thread/queue.. when ALL of the tasks above has been
completed.
        dispatchGroup.notify(queue: DispatchQueue.main) {

            print("Every task is complete")

        }

        //Start the tasks.
        firstTask.resume()
        secondTask.resume()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}

```

Mit dem oben genannten müssen Sie nicht unendlich lange `wait` bis alle Aufgaben abgeschlossen sind. Sie können einen Loader anzeigen, **BEVOR** alle Tasks gestartet sind und den Loader **BEENDEN**, nachdem alle Tasks abgeschlossen sind. Auf diese Weise wird Ihr Haupt-Thread nicht blockiert und Ihr Code bleibt sauber.

Nehmen Sie nun an, Sie möchten auch, dass die Aufgaben `ordered` oder fügen Sie die Antworten nacheinander in ein Array ein. Sie könnten folgendes tun:

```

import UIKit

//Locking mechanism..
func synchronized(_ lock: AnyObject, closure: () -> Void) {
    objc_sync_enter(lock)
    closure()
    objc_sync_exit(lock)
}

class ViewController: UIViewController {

    let lock = NSObject() //Object to lock on.
    var responseArray = Array<Data?>() //Array of responses.

    override func viewDidLoad() {
        super.viewDidLoad()

        let dispatchGroup = DispatchGroup()
        let session: URLSession = URLSession.shared
    }
}

```

```

    dispatchGroup.enter() //Enter the group for the first task.

    let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com"!)) { (data, response, error) in

        //Process Response..

        synchronized(self.lock, closure: { () -> Void in
            self.responseArray[0] = data ?? nil
        })

        dispatchGroup.leave() //Leave the group for the first task.
    })

    dispatchGroup.enter() //Enter the group for the second task.

    let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca"!)) { (data, response, error) in

        //Process Response..

        synchronized(self.lock, closure: { () -> Void in
            self.responseArray[1] = data ?? nil
        })

        dispatchGroup.leave() //Leave the group for the second task.
    })

//Get notified on the main thread.. when ALL of the requests above has been completed.
dispatchGroup.notify(queue: DispatchQueue.main) {

    print("Every task is complete..")

    for i in 0..

```

Anmerkungen

Jeder Eintrag muss einen Ausgang in einer `DispatchGroup` . Wenn Sie nach dem `entering` vergessen zu `leave` , richten Sie sich ein. Sie werden NIEMALS benachrichtigt, wenn die Aufgaben abgeschlossen sind.

Der Betrag für die `enter` muss der Höhe des `leave` .

DispatchGroup online lesen: <https://riptutorial.com/de/ios/topic/4624/dispatchgroup>

Kapitel 52: Dynamischer Typ

Bemerkungen

```
// Content size category constants
UIContentSizeCategoryExtraSmall
UIContentSizeCategorySmall
UIContentSizeCategoryMedium
UIContentSizeCategoryLarge
UIContentSizeCategoryExtraLarge
UIContentSizeCategoryExtraExtraLarge
UIContentSizeCategoryExtraExtraExtraLarge

// Accessibility sizes
UIContentSizeCategoryAccessibilityMedium
UIContentSizeCategoryAccessibilityLarge
UIContentSizeCategoryAccessibilityExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraExtraLarge
```

Examples

Rufen Sie die aktuelle Inhaltsgröße ab

Schnell

```
UIApplication.sharedApplication().preferredContentSizeCategory
```

Ziel c

```
[UIApplication sharedApplication].preferredContentSizeCategory;
```

Dies gibt eine konstante Inhaltsgrößenkategorie oder eine konstante Inhaltsgrößenkategorie für Eingabehilfen zurück.

Benachrichtigung zur Änderung der Textgröße

Sie können sich für Benachrichtigungen registrieren, wenn die Gerätetextgröße geändert wird.

Schnell

```
NSNotificationCenter.defaultCenter().addObserver(self, selector: #selector(updateFont), name:
name:UIContentSizeCategoryDidChangeNotification, object: nil)
```

Ziel c

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(updateFont)
name:UIContentSizeCategoryDidChangeNotification object:nil];
```

Das `userInfo` Benachrichtigungsobjekt enthält die neue Größe unter

`UIContentSizeCategoryNewValueKey`.

Passende dynamische Schriftgröße in WKWebView

WKWebView passt die Schriftarten für Webinhalte so an, dass eine Webseite in voller Größe in den Formfaktor des Geräts passt. Wenn Sie möchten, dass der Webtext sowohl im Hoch- als auch im Querformat in der Größe der bevorzugten Lesegröße des Benutzers entspricht, müssen Sie ihn explizit festlegen.

Schnell

```
// build HTML header for dynamic type and responsive design
func buildHTMLHeader() -> String {

    // Get preferred dynamic type font sizes for html styles
    let bodySize = UIFont.preferredFont(forTextStyle: UIFontTextStyle.body).pointSize
    let h1Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title1).pointSize
    let h2Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title2).pointSize
    let h3Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title3).pointSize

    // On iPad, landscape text is larger than preferred font size
    var portraitMultiplier = CGFloat(1.0)
    var landscapeMultiplier = CGFloat(0.5)

    // iPhone text is shrunken
    if UIDevice.current.model.range(of: "iPhone") != nil {
        portraitMultiplier = CGFloat(3.0)
        landscapeMultiplier = CGFloat(1.5)
    }

    // Start HTML header text
    let patternText = "<html> <head> <style> "

    // Match Dynamic Type for this page.
    + "body { background-color: \(backgroundColor); } "
    + "@media all and (orientation:portrait) {img {max-width: 90%; height: auto;} "
    + "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * portraitMultiplier)px + 1.0vw); font-weight: normal; color:
\(\fontColor) } "
    + "h1 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
    + "h2 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h2Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
    + "h3, h4 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
```

```

size:calc(\(h3Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } } "
+ "@media all and (orientation:landscape) {img {max-width: 65%; height: auto;}"
+ "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * landscapeMultiplier)px + 1.0vw); font-weight: normal; color:
\(\fontColor) }"
+ "h1 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
+ "h2 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h2Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
+ "h3, h4 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h3Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } } </style>"
+ "</head><body>"
+ "<meta name=\"viewport\" content=\"width: device-width\">"

return patternText
}

```

Umgang mit der bevorzugten Änderung der Textgröße ohne Benachrichtigungen unter iOS 10

UILabel Klassen UILabel , UITextField und UITextView verfügen ab iOS 10 über eine neue Eigenschaft, deren Schriftgröße automatisch geändert wird, wenn ein Benutzer seine bevorzugte Lesegröße namens `adjustsFontForContentSizeCategory` .

Schnell

```

@IBOutlet var label:UILabel!

if #available(iOS 10.0, *) {
    label.adjustsFontForContentSizeCategory = true
} else {
    // Observe for UIContentSizeCategoryDidChangeNotification and handle it manually
    // since the adjustsFontForContentSizeCategory property isn't available.
}

```

Dynamischer Typ online lesen: <https://riptutorial.com/de/ios/topic/4466/dynamischer-typ>

Kapitel 53: Erstellen Sie ein benutzerdefiniertes Framework in iOS

Examples

Erstellen Sie ein Framework in Swift

Führen Sie die folgenden Schritte aus, um ein benutzerdefiniertes Framework in Swift-IOS zu erstellen:

1. Erstellen Sie ein neues Projekt. In Xcode
2. Wählen Sie iOS / Framework & Library / Cocoa Touch Framework, um ein neues Framework zu erstellen
3. Klicken Sie auf Weiter und legen Sie den Produktnamen fest
4. Klicken Sie auf Weiter und wählen Sie das Verzeichnis, um das Projekt dort zu erstellen
5. Fügen Sie dem erstellten Projekt Code und Ressourcen hinzu

Framework erfolgreich erstellt

Um ein erstelltes Framework zu einem anderen Projekt hinzuzufügen, sollten Sie zuerst einen Arbeitsbereich erstellen

Fügen Sie dem Arbeitsbereich "Zielprojekt" und "Rahmenprojekt" hinzu.

1. Gehen Sie zur Registerkarte Allgemein des Zielprojekts
2. Ziehen Sie die Datei "*.framework" im Produktordner des Rahmenprojekts in den Abschnitt "Embedded Binaries"
3. In jedem ViewController oder in jeder Klasse verwenden, importieren Sie einfach das Framework in jeder Datei

Erstellen Sie ein benutzerdefiniertes Framework in iOS online lesen:

<https://riptutorial.com/de/ios/topic/7331/erstellen-sie-ein-benutzerdefiniertes-framework-in-ios>

Kapitel 54: Erstellen Sie ein Video aus Bildern

Einführung

Erstellen Sie ein Video aus Bildern mit AVFoundation

Examples

Erstellen Sie ein Video aus Ullimages

Zunächst müssen Sie `AVAssetWriter` erstellen

```
NSError *error = nil;
NSURL *outputURL = <#NSURL object representing the URL where you want to save the video#>;
AVAssetWriter *assetWriter = [AVAssetWriter assetWriterWithURL:outputURL
                             fileType:AVFileTypeQuickTimeMovie error:&error];
if (!assetWriter) {
    // handle error
}
```

`AVAssetWriter` benötigt mindestens einen Asset-Writer-Eingang.

```
NSDictionary *writerInputParams = [NSDictionary dictionaryWithObjectsAndKeys:
                                   AVVideoCodecH264, AVVideoCodecKey,
                                   [NSNumber numberWithInt:renderSize.width],
                                   AVVideoWidthKey,
                                   [NSNumber numberWithInt:renderSize.height],
                                   AVVideoHeightKey,
                                   AVVideoScalingModeResizeAspectFill,
                                   AVVideoScalingModeKey,
                                   nil];

AVAssetWriterInput *assetWriterInput = [AVAssetWriterInput
assetWriterInputWithMediaType:AVMediaTypeVideo outputSettings:writerInputParams];
if ([assetWriter canAddInput:assetWriterInput]) {
    [assetWriter addInput:assetWriterInput];
} else {
    // show error message
}
```

Anhängen `CVPixelBufferRef`, `AVAssetWriterInput` müssen wir schaffen

`AVAssetWriterInputPixelBufferAdaptor`

```
NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
                            [NSNumber numberWithInt:kCVPixelFormatType_32ARGB],
                            (NSString*)kCVPixelBufferPixelFormatKey,
                            [NSNumber numberWithBool:YES], (NSString*)kCVPixelBufferCGImageCompatibilityKey,
                            [NSNumber numberWithBool:YES], (NSString*)kCVPixelBufferCGBitmapContextCompatibilityKey,
                            nil];
AVAssetWriterInputPixelBufferAdaptor *writerAdaptor = [AVAssetWriterInputPixelBufferAdaptor
```

```
assetWriterInputPixelBufferAdaptorWithAssetWriterInput:assetWriterInput
sourcePixelBufferAttributes:attributes];
```

Jetzt können wir anfangen zu schreiben

```
[assetWriter startWriting];
[assetWriter startSessionAtSourceTime:kCMTIMEZERO];
[assetWriterInput requestMediaDataWhenReadyOnQueue:exportingQueue usingBlock:^(
    for (int i = 0; i < images.count; ++i) {
        while (![assetWriterInput isReadyForMoreMediaData]) {
            [NSThread sleepForTimeInterval:0.01];
            // can check for attempts not to create an infinite loop
        }

        UIImage *uIImage = images[i];

        CVPixelBufferRef buffer = NULL;
        CVReturn err = PixelBufferCreateFromImage(uIImage.CGImage, &buffer);
        if (err) {
            // handle error
        }

        // frame duration is duration of single image in seconds
        CMTIME presentationTime = CMTIMEMakeWithSeconds(i * frameDuration, 1000000);

        [writerAdaptor appendPixelBuffer:buffer withPresentationTime:presentationTime];

        CVPixelBufferRelease(buffer);
    }

[assetWriterInput markAsFinished];
[assetWriter finishWritingWithCompletionHandler:^(
    if (assetWriter.error) {
        // show error message
    } else {
        // outputURL
    }
}]];
}];
```

Hier ist eine Funktion, um CVPixelBufferRef von CGImageRef

```
CVReturn PixelBufferCreateFromImage(CGImageRef imageRef, CVPixelBufferRef *outBuffer) {
    CIContext *context = [CIContext context];
    CIImage *ciImage = [CIImage imageWithCGImage:imageRef];

    NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
        [NSNumber numberWithInt:YES], (NSString
*)kCVPixelBufferCGBitmapContextCompatibilityKey,
        [NSNumber numberWithInt:YES], (NSString
*)kCVPixelBufferCGImageCompatibilityKey
        , nil];

    CVReturn err = CVPixelBufferCreate(kCFAllocatorDefault, CGImageGetWidth(imageRef),
    CGImageGetHeight(imageRef), kCVPixelFormatType_32ARGB, (__bridge CFDictionaryRef
    _Nullable)(attributes), outBuffer);
    if (err) {
        return err;
    }
}
```

```
if (outBuffer) {
    [context render:ciImage toCVPixelBuffer:*outBuffer];
}

return kCVReturnSuccess;
}
```

Erstellen Sie ein Video aus Bildern online lesen: <https://riptutorial.com/de/ios/topic/10607/erstellen-sie-ein-video-aus-bildern>

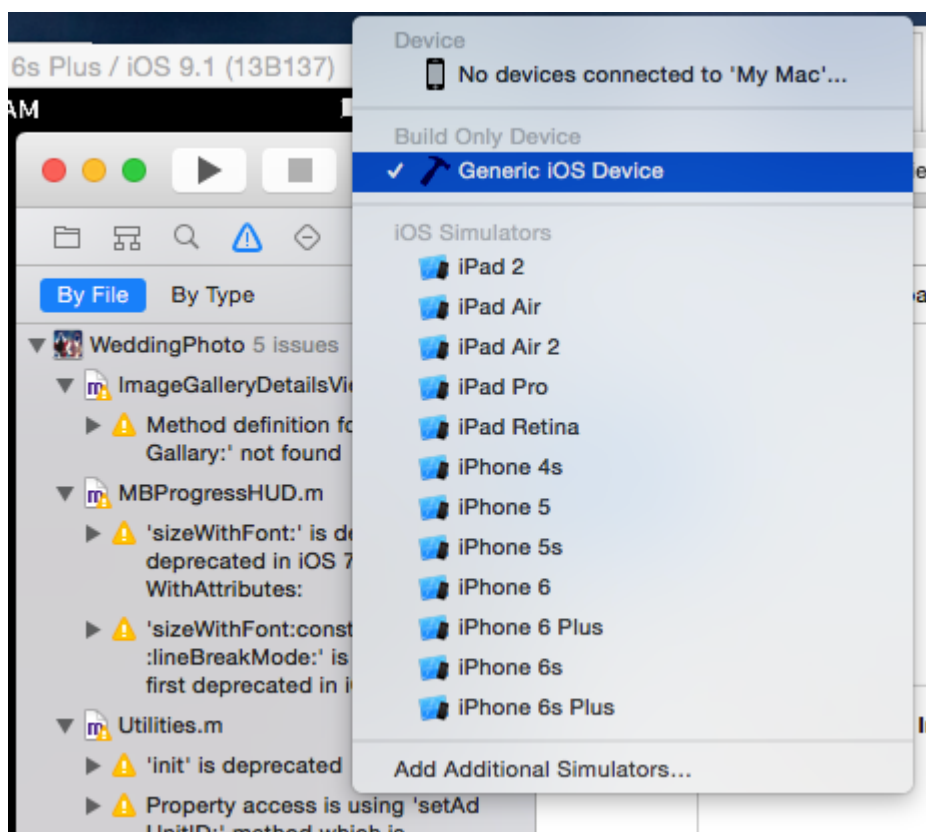
Kapitel 55: Erstellen Sie eine .ipa-Datei, die mit Apploadloader im Appstore hochgeladen wird

Examples

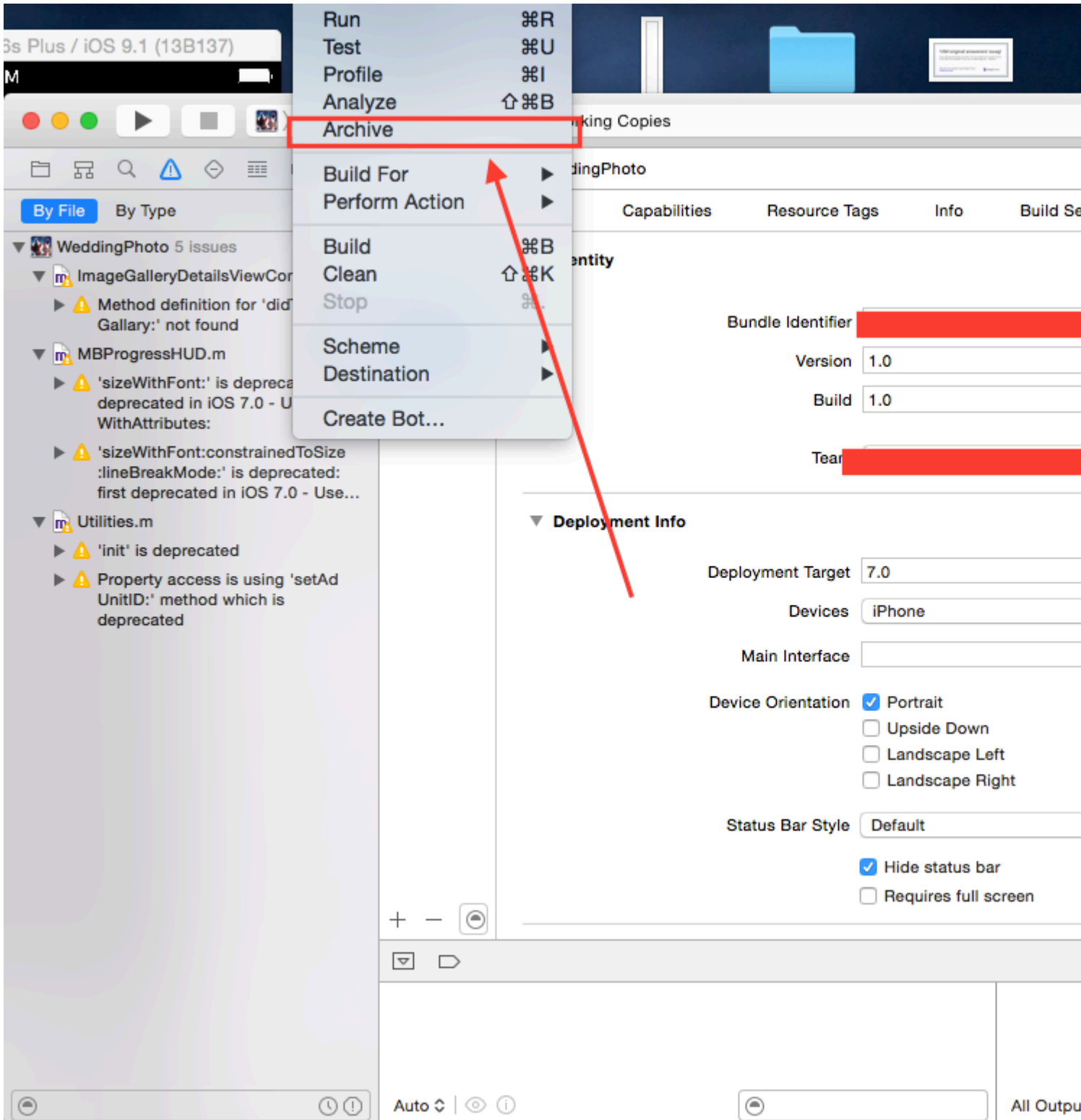
Erstellen Sie eine .ipa-Datei, um die App mit dem Application Loader in den Appstore zu laden

Wenn Sie eine .ipa-Datei auf itunesconnect hochladen möchten, **ohne das Entwicklerkonto in Xcode zu integrieren**, und den **Application Loader verwenden** . dann können Sie .ipa mit **iTunes generieren** .

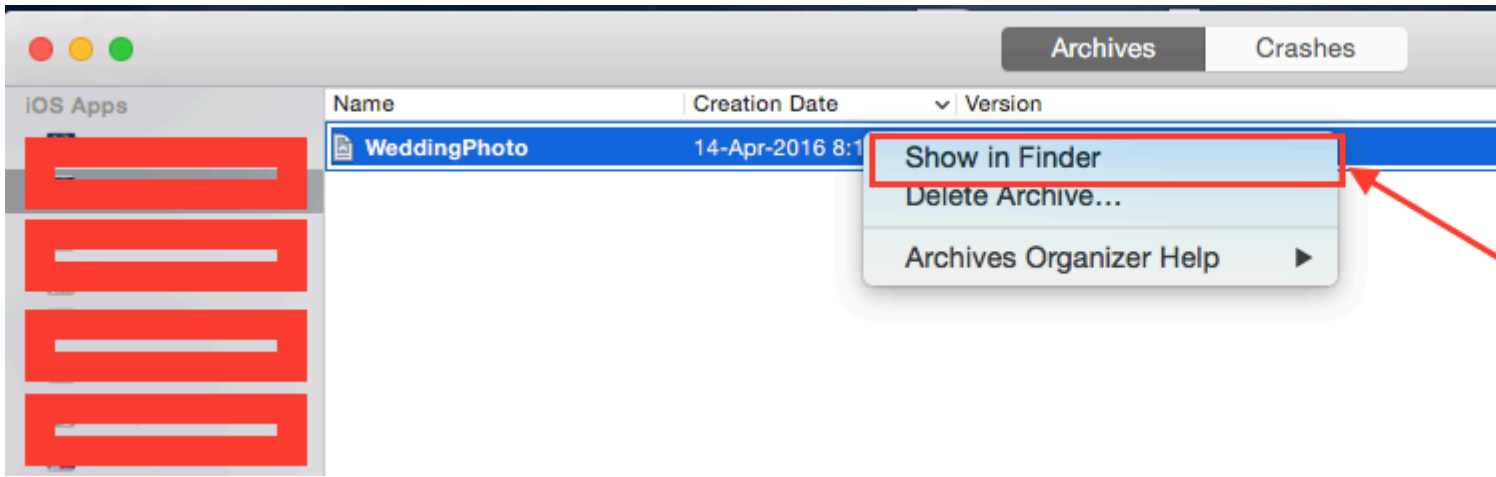
Schritt 1: - Wählen Sie das Gerät anstelle des Simulators.



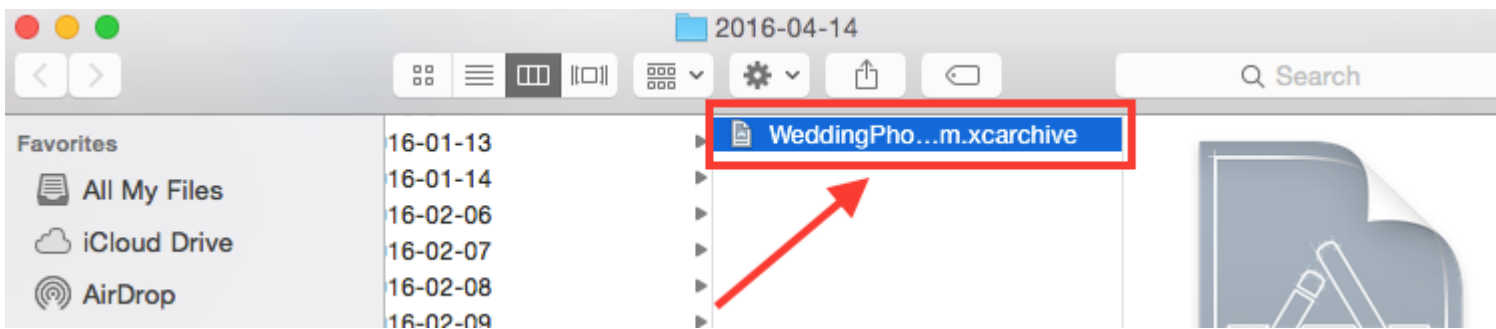
Schritt 2: - Gehen Sie zu Produkt -> Wählen Sie Archiv



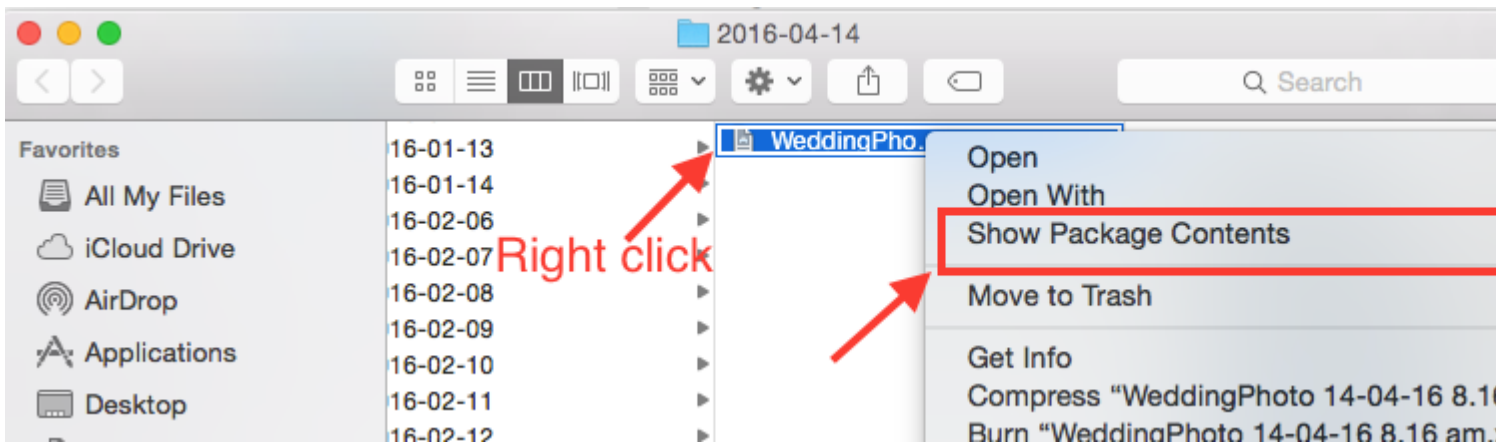
Schritt 3: - Klicken Sie nach Abschluss des Vorgangs mit der rechten Maustaste auf Ihr Archiv -> und wählen Sie In Finder anzeigen



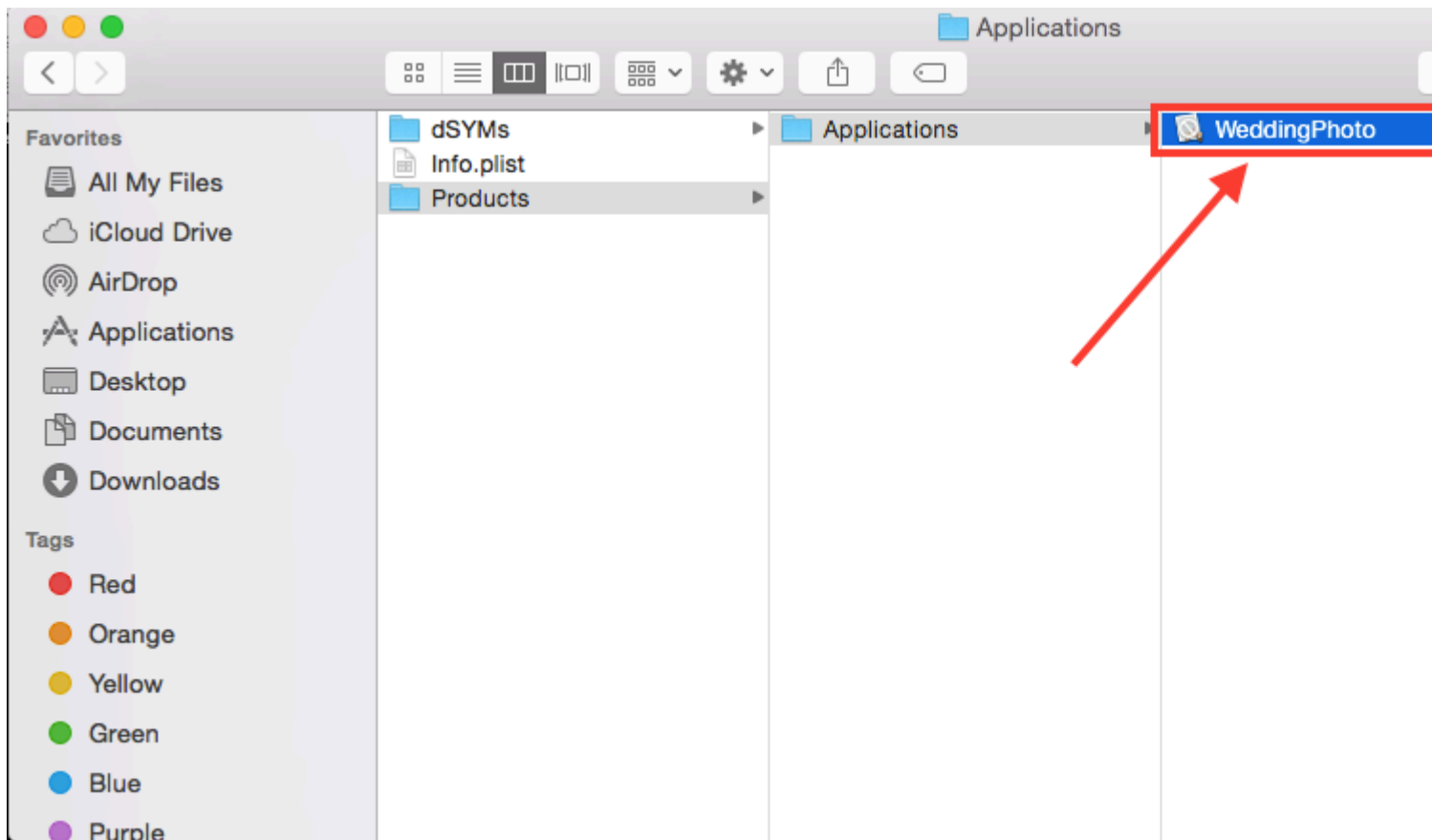
Schritt 4: - Wenn Sie auf im Finder anzeigen klicken, werden Sie in den Archivordner umgeleitet



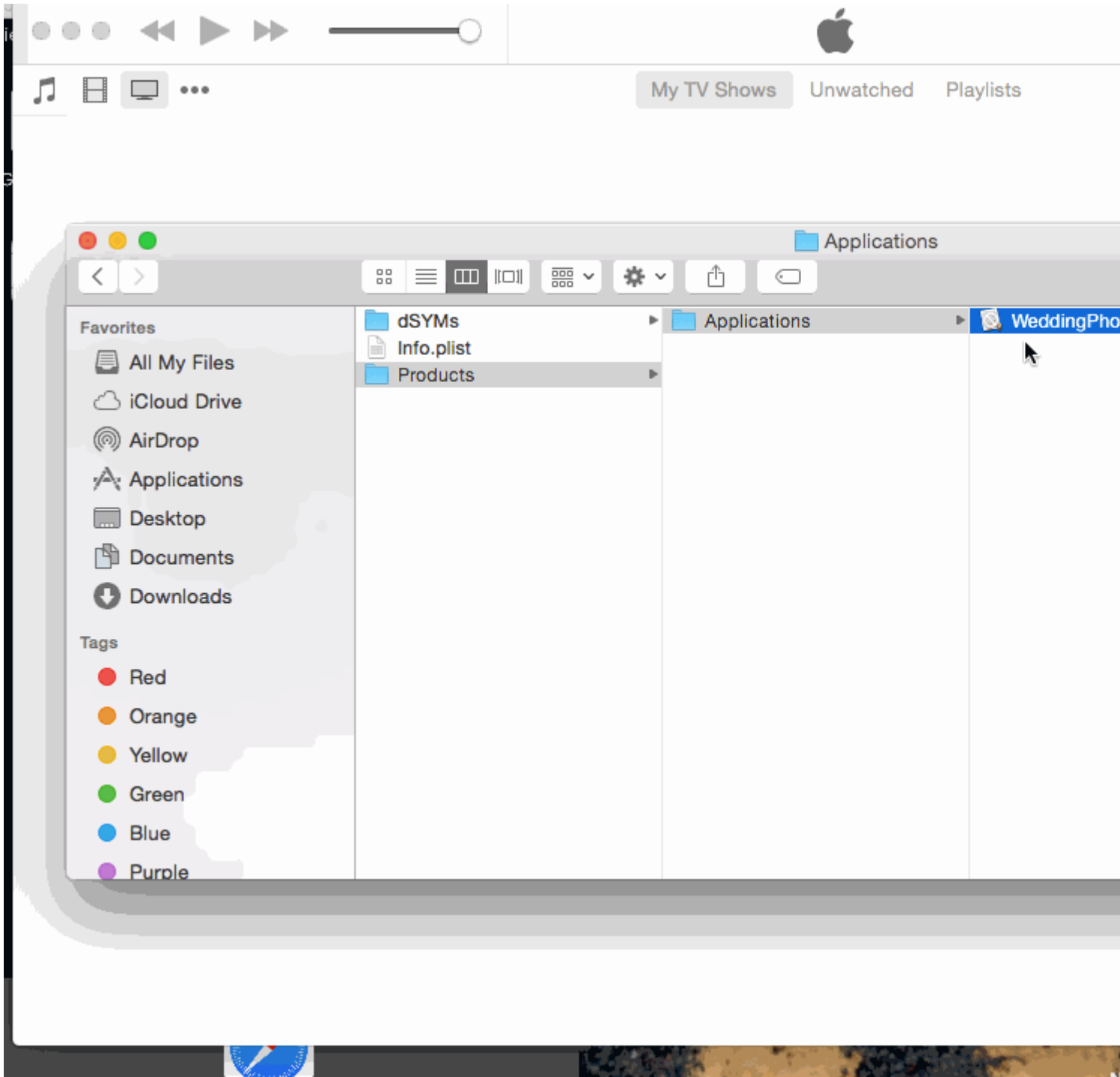
Schritt 5: - Klicken Sie mit der rechten Maustaste auf die .xarchive-Datei -> wählen Sie die Option In Finder anzeigen.



Schritt 6: - Gehen Sie zu Produktordner -> Anwendungsordner -> Sie finden Ihren Projektnamen.app



Schritt 7: - Um .app in .ipa zu konvertieren, ziehen Sie sie einfach in iTunes. überprüfen sie unten bild,



Schritt 8: - Legen Sie diese .ipa-Datei an einem sicheren Ort ab und verwenden Sie sie beim Hochladen mit dem Anwendungsladeprogramm.

Hinweis: - Wenn Sie wissen möchten, wie eine App mit dem Anwendungsladeprogramm hochgeladen wird, überprüfen Sie dies.

[App mit dem Programm Loader hochladen](#)

EDIT: -

WARNUNG: - Machen Sie keine .ipa-Datei, wenn Sie die Erweiterung von .aap in .zip und .zip in .ipa ändern.

Ich habe in vielen Antworten gesehen, dass sie eine .app-Datei vorschlagen und dann die Erweiterung von .zip in .ipa ändern. Es funktioniert jetzt nicht. Durch diese Methode erhalten Sie Fehler wie

IPA ist ungültig, es enthält kein Payload-Verzeichnis.

Erstellen Sie eine .ipa-Datei, die mit Apploadloader im Appstore hochgeladen wird online lesen:
<https://riptutorial.com/de/ios/topic/6119/erstellen-sie-eine-ipa-datei--die-mit-apploadloader-im-appstore-hochgeladen-wird>

Kapitel 56: Erweiterung für umfassende Push-Benachrichtigung - iOS 10.

Einführung

iOS 10 gab uns `UserNotifications.framework`, die neue API für lokale / Remote-Benachrichtigungen. Es bietet das Anzeigen von Medienanhängen oder das Beantworten von Nachrichten direkt aus der Benachrichtigung.

Der Benachrichtigungsinhalt besteht aus Titel, Untertitel, Text und Anhang. Anhänge können Bilder / Gifs / Videos bis zu 50 MB enthalten.

Examples

Benachrichtigungsinhalt-Erweiterung

Warum brauchen wir es?

Die Inhaltserweiterung hilft uns, eine benutzerdefinierte Benutzeroberfläche nach Benachrichtigungseinstellung zu erstellen.

Sie verwenden dieses Framework, um eine Erweiterung zu definieren, die die Benachrichtigungsdaten empfängt und die entsprechende visuelle Darstellung bereitstellt. Ihre Erweiterung kann auch auf benutzerdefinierte Aktionen reagieren, die mit diesen Benachrichtigungen verknüpft sind.

Implementierung

1. Gehen Sie im xCode `Navigator` Fenster zum Abschnitt `Targets`. Klicken `Add New Target`.
2. Vorlage für `Notification Content Extension` auswählen:

Choose a template for your new target:

ios watchOS tvOS macOS Cross-Platform

Call Directory Extension

Content Blocker Extension

iMessage

Intents Extension

Notification Service Extension

Photo Editing Extension

Spotlight Index

Sticker Pack

Cancel

Schlüssel fest:

▼ NSExtension

▼ NSExtensionAttributes

UNNotificationExtensionDefaultContentHidden

UNNotificationExtensionCategory

UNNotificationExtensionInitialContentSizeRatio

NSExtensionMainStoryboard

NSExtensionPointIdentifier

UNNotificationExtensionAttributes :

`UNNotificationExtensionCategory` (Erforderlich)

Der Wert dieses Schlüssels ist eine Zeichenfolge oder ein String-Array. Jede Zeichenfolge enthält den Bezeichner einer Kategorie, die von der App mithilfe der Klasse `UNNotificationCategory` deklariert wurde.

`UNNotificationExtensionInitialContentSizeRatio` (Erforderlich)

Zahl, die die Anfangsgröße der Ansicht des View Controller darstellt, ausgedrückt als Verhältnis von Höhe zu Breite.

`UNNotificationExtensionDefaultContentHidden` (optional)

Bei der Einstellung YES zeigt das System nur Ihren benutzerdefinierten View-Controller in der Benachrichtigungsoberfläche an. Bei NEIN zeigt das System den Standardbenachrichtigungsinhalt zusätzlich zum Inhalt Ihres View-Controllers an.

`UNNotificationExtensionOverridesDefaultTitle` (optional)

Der Wert dieses Schlüssels ist ein Boolean. Bei der Einstellung true verwendet das System die `title`-Eigenschaft Ihres View-Controllers als Titel der Benachrichtigung. Bei Festlegung auf "false" setzt das System den Titel der Benachrichtigung auf den Namen Ihrer App. Wenn Sie diesen Schlüssel nicht angeben, wird der Standardwert auf "false" gesetzt.

4. Erstellen Sie eine benutzerdefinierte Ansicht in der Datei `NotificationViewController.swift`
5. Fügen Sie einen neuen `category` key und setzen Sie seinen Wert auf das, was wir in der Info.plist (Schritt 3) eingegeben haben:

Drücken:

```
{  
  aps: {
```

```
alert: { ... },
category: 'io.swifting.notification-category'
}
}
```

Lokal:

```
let mutableNotificationContent = UNMutableNotificationContent()
mutableNotificationContent.category = "io.swifting.notification-category"
mutableNotificationContent.title = "Swifting.io Notifications"
mutableNotificationContent.subtitle = "Swifting.io presents"
mutableNotificationContent.body = "Custom notifications"
```

Überprüfen Sie auch die offizielle API-Referenz:

https://developer.apple.com/reference/usernotificationsui/unnotificationcontentextension?utm_source=swift

Erweiterung für umfassende Push-Benachrichtigung - iOS 10. online lesen:

<https://riptutorial.com/de/ios/topic/9501/erweiterung-fur-umfassende-push-benachrichtigung---ios-10->

Kapitel 57: EventKit

Examples

Erlaubnis beantragen

Ihre App kann ohne Erlaubnis nicht auf Ihre Erinnerungen und Ihren Kalender zugreifen. Stattdessen muss der Benutzer eine Warnung erhalten, die ihn auffordert, Zugriff auf Ereignisse für die App zu gewähren.

Importieren Sie zunächst das `EventKit` Framework:

Schnell

```
import EventKit
```

Ziel c

```
#import <EventKit/EventKit.h>
```

Einen `EKEventStore`

Dann `EKEventStore` wir ein `EKEventStore` Objekt. Dies ist das Objekt, von dem aus wir auf Kalender- und Erinnerungsdaten zugreifen können:

Schnell

```
let eventStore = EKEventStore()
```

Ziel c

```
EKEventStore *eventStore = [[EKEventStore alloc] init];
```

Hinweis

Das `EKEventStore` eines `EKEventStore` Objekts bei jedem Zugriff auf den Kalender ist nicht effizient. Versuchen Sie es einmal zu erstellen und verwenden Sie es überall in Ihrem Code.

Verfügbarkeit überprüfen

Verfügbarkeit hat drei verschiedene Status: Autorisiert, Abgelehnt und Nicht bestimmt. Nicht bestimmt bedeutet, dass die App Zugriff gewähren muss.

Um die Verfügbarkeit zu prüfen, verwenden wir die Methode `EKEventStore.authorizationStatusForEntityType()` des `EKEventStore` Objekts:

Schnell

```
switch EKEventStore.authorizationStatusForEntityType(EKEntityTypeEvent) {
    case .Authorized: //...
    case .Denied: //...
    case .NotDetermined: //...
    default: break
}
```

Ziel c

```
switch ([EKEventStore authorizationStatusForEntityType:EKEntityTypeEvent]){
    case EKAAuthorizationStatus.Authorized:
        //...
        break;
    case EKAAuthorizationStatus.Denied:
        //...
        break;
    case EKAAuthorizationStatus.NotDetermined:
        //...
        break;
    default:
        break;
}
```

Erlaubnis beantragen

`NotDetermined` Sie den folgenden Code in den `NotDetermined` Fall ein:

Schnell

```
eventStore.requestAccessToEntityType(EKEntityTypeEvent, completion: { [weak self]
(userGrantedAccess, _) -> Void in
    if userGrantedAccess{
        //access calendar
    }
})
```

Zugriff auf verschiedene Kalendertypen

Zugriff auf die Kalenderreihe

Für den Zugriff auf das Array von `EKCalendar` wir die Methode `calendarsForEntityType` :

Schnell

```
let calendarsArray = eventStore.calendarsForEntityType(EKEntityType.Event) as! [EKCalendar]
```

Iteration durch Kalender

Verwenden Sie einfach eine einfache `for` Schleife:

Schnell

```
for calendar in calendarsArray{  
    //...  
}
```

Zugriff auf den Kalendertitel und die Farbe

Schnell

```
let calendarColor = UIColor(CGColor: calendar.CGColor)  
let calendarTitle = calendar.title
```

Ziel c

```
UIColor *calendarColor = [UIColor initWithCGColor: calendar.CGColor];  
NSString *calendarTitle = calendar.title;
```

Ereignis hinzufügen

Ereignisobjekt erstellen

Schnell

```
var event = EKEvent(eventStore: eventStore)
```

Ziel c

```
EKEvent *event = [EKEvent initWithEventStore:eventStore];
```

Einstellen des zugehörigen Kalenders, Titels und Datums

Schnell

```
event.calendar = calendar  
event.title = "Event Title"  
event.startDate = startDate //assuming startDate is a valid NSDate object  
event.endDate = endDate //assuming endDate is a valid NSDate object
```

Ereignis zum Kalender hinzufügen

Schnell

```
try {  
    do eventStore.saveEvent(event, span: EKSpan.ThisEvent)  
} catch let error as NSError {  
    //error  
}
```

Ziel c

```
NSError *error;  
BOOL *result = [eventStore saveEvent:event span:EKSpanThisEvent error:&error];  
if (result == NO){  
    //error  
}
```

EventKit online lesen: <https://riptutorial.com/de/ios/topic/5854/eventkit>

Kapitel 58: FacebookSDK

Examples

FacebookSDK-Integration

Schritt 1: Installieren Sie das SDK

Sie können das SDK [manuell](#) oder über `CocoaPods` installieren. Die letztere Option wird dringend empfohlen.

Podfile diese Zeilen in Podfile :

```
target 'MyApp' do
  use_frameworks!

  pod 'FBSDKCoreKit'
  pod 'FBSDKLoginKit'
  pod 'FBSDKShareKit'
end
```

Führen Sie die `pod install` im Terminal aus und öffnen Sie anschließend `.xcworkspace` anstelle von `.xcodeproj`.

`FBSDKLoginKit` und `FBSDKShareKit` sind optional. Möglicherweise brauchen Sie sie nicht.

Schritt 2: Erstellen Sie eine App auf Facebook

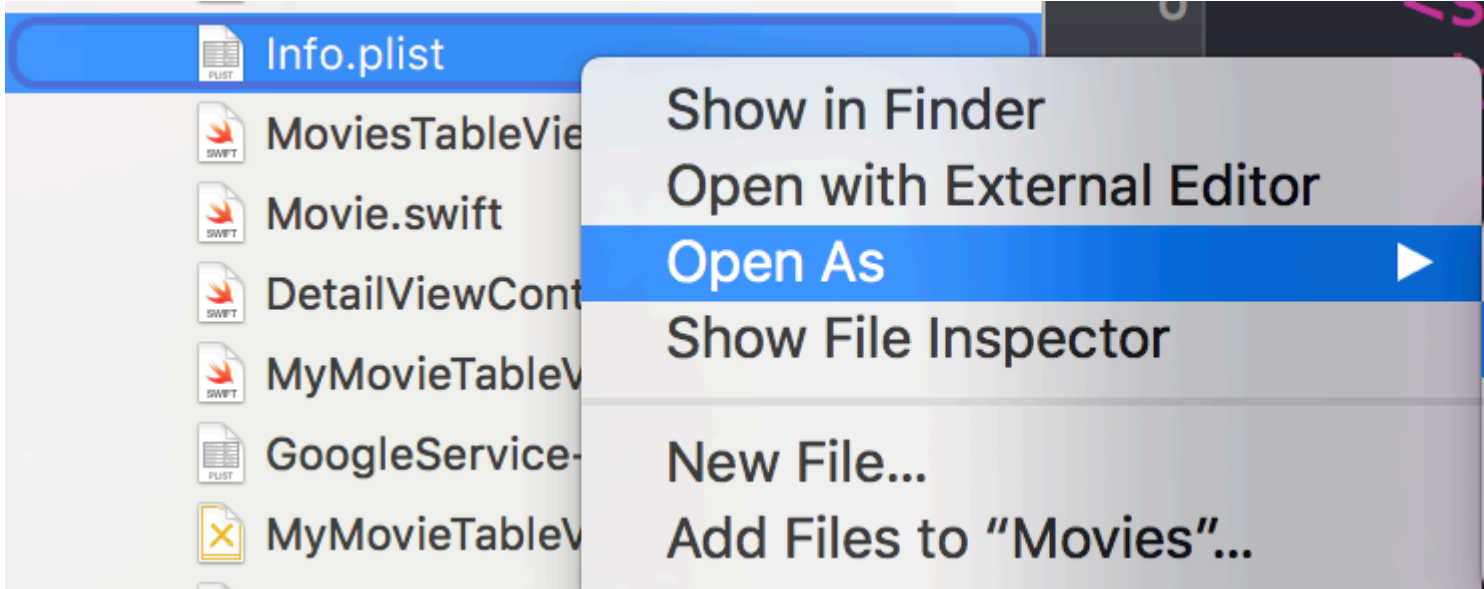
Gehen Sie zu [Schnellstart - Facebook für Entwickler](#), um eine App zu erstellen.

Facebook fordert Sie auf, das SDK nach der Erstellung der App herunterzuladen. Sie können diesen Teil überspringen, wenn Sie das SDK bereits über `CocoaPods` installiert haben.

Schritt 3: Bearbeiten Sie die `.plist`

ein. Damit Ihre App mit Facebook "kommunizieren" kann, müssen Sie einige Einstellungen in Ihrer `.plist` Datei `.plist`. Facebook zeigt Ihnen das angepasste Snippet auf der Schnellstartseite.

b. Bearbeiten Sie Ihre `.plist` Datei als Quellcode.



c. Fügen Sie Ihr benutzerdefiniertes Snippet in den Quellcode ein. **Achtung!** Das Snippet muss genau das Kind des `<dict>` -Tags sein. Ihr Quellcode sollte ungefähr so aussehen:

```
<plist version="1.0">
<dict>
  // ...
  //some default settings
  // ...
  <key>CFBundleURLTypes</key>
  <array>
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>fb{FBAppId}</string>
      </array>
    </dict>
  </array>
  <key>FacebookAppID</key>
  <string>{FBAppId}</string>
  <key>FacebookDisplayName</key>
  <string>{FBAppName}</string>
  <key>LSApplicationQueriesSchemes</key>
  <array>
    <string>fbapi</string>
    <string>fb-messenger-api</string>
    <string>fbauth2</string>
    <string>fbshareextension</string>
  </array>
  <key>NSAppTransportSecurity</key>
  <dict>
    <key>NSExceptionDomains</key>
    <dict>
      <key>facebook.com</key>
      <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSEnvironmentRequiresForwardSecrecy</key>
        <false/>
      </dict>
      <key>fbcdn.net</key>
    </dict>
  </dict>
</plist>
```

```

        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSEnvironmentRequiresForwardSecrecy</key>
        <false/>
    </dict>
    <key>akamaihd.net</key>
    <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSEnvironmentRequiresForwardSecrecy</key>
        <false/>
    </dict>
</dict>
</plist>

```

Wenn Sie das Snippet an einer falschen Stelle einfügen, stoßen Sie auf Probleme.

Schritt 4: Teilen Sie Facebook Ihre Paketkennung auf der Schnellstartseite mit.

=> [Wie bekomme ich eine Paketkennung?](#)

Schritt 5: Bearbeiten Sie Ihre `AppDelegate.swift`

ein.

```
import FBSDKCoreKit
```

b.

```

func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    FBSDKApplicationDelegate.sharedInstance().application(application,
didFinishLaunchingWithOptions: launchOptions)
    return true
}

func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?,
annotation: AnyObject) -> Bool {
    return FBSDKApplicationDelegate.sharedInstance().application(application, openURL: url,
sourceApplication: sourceApplication, annotation: annotation)
}

```

Erstellen Sie Ihren eigenen benutzerdefinierten Button "Mit Facebook anmelden"

Manchmal möchten wir eine eigene Benutzeroberfläche für die Schaltfläche "Mit Facebook anmelden" anstelle der mit FacebookSDK gelieferten Schaltfläche erstellen.

1. Ziehen Sie in Ihrem Storyboard Ihren UIButton und stellen Sie ihn so ein, wie Sie möchten.
2. Strg + Ziehen Sie Ihre Schaltfläche als IBAction auf Ihren View-Controller.
3. In der IBAction-Methode müssen Sie den tatsächlichen Facebook-Button wie folgt antippen:

Schnell:

```

let loginButton = FBSDKLoginButton()
loginButton.delegate = self
// Your Custom Permissions Array
loginButton.readPermissions =
[
    "public_profile",
    "email",
    "user_about_me",
    "user_photos"
]
// Hiding the button
loginButton.hidden = true
self.view.addSubview(loginButton)
// Simulating a tap for the actual Facebook SDK button
loginButton.sendActionsForControlEvents(UIControlEvents.TouchUpInside)

```

Ziel c:

```

FBSDKLoginButton *FBButton = [FBSDKLoginButton new];

// Your Custom Permissions Array
FBButton.readPermissions = @[@"public_profile",
    @"email",
    @"user_about_me",
    @"user_photos"
];

FBButton.loginBehavior = FBSDKLoginBehaviorNative;
[FBButton setDelegate:self];
[FBButton setHidden:true];
[loginButton addSubview:FBButton];

[FBButton sendActionsForControlEvents:UIControlEventTouchUpInside];

```

Sie sind fertig.

Facebook-Benutzerdaten abrufen

Nachdem sich der Benutzer bei Ihrer App bei Facebook angemeldet hat, können Sie jetzt die von den `FBButton.readPermissions` angeforderten Daten `FBButton.readPermissions` .

Schnell:

```

enum FacebookParametesField : String
{
    case FIELDS_KEY = "fields"
    case FIELDS_VALUE = "id, email, picture, first_name, last_name"
}

if FBSDKAccessToken.currentAccessToken() != nil
{
    // Getting user facebook data
    FBSDKGraphRequest(graphPath: "me",
        parameters: [FacebookParametesField.FIELDS_KEY.rawValue :
        FacebookParametesField.FIELDS_VALUE.rawValue])
    .startWithCompletionHandler({ (graphConnection : FBSDKGraphRequestConnection!, result :
    AnyObject!, error : NSError!) -> Void in

```

```

if error == nil
{
    print("Facebook Graph phaze")

    let email = result["email"]
    let facebookToken = FBSDKAccessToken.currentAccessToken().tokenString
    let userFacebookId = result["id"]
    let firstName = result["first_name"]
    let lastName = result["last_name"]

    if let result = result as? Dictionary<String, AnyObject>
    {
        if let picture = result["picture"] as? Dictionary<String,AnyObject>
        {
            if let data = picture["data"] as? Dictionary <String,AnyObject>
            {
                if let url = data["url"] as? String
                {
                    // Profile picture URL
                    let profilePictureURL = url
                }
            }
        }
    }
}
})
}

```

FacebookSDK online lesen: <https://riptutorial.com/de/ios/topic/2972/facebooksdk>

Kapitel 59: Farbe der Statusleiste ändern

Examples

Für Nicht-UINavigationController-Statusleisten

1. Setzen Sie unter `info.plist` die Darstellung der `view controller-based status bar appearance` auf `YES`
2. In View-Controllern, die nicht in `UINavigationController` enthalten `UINavigationController` implementieren Sie diese Methode.

In Ziel-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

In Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return UIStatusBarStyle.LightContent
}
```

Für UINavigationController-Statusleisten

Unterklasse `UINavigationController` und überschreiben Sie dann diese Methoden:

In Ziel-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

In Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return .lightContent
}
```

Alternativ können Sie `barStyle` in der `UINavigationController` Instanz `UINavigationController` :

Ziel c:

```
// e.g. in your view controller's viewDidLoad method:
self.navigationController.navigationBar.barStyle = UIBarStyleBlack; // this will give you a
white status bar
```


Schnell

```
// e.g. in your view controller's viewDidLoad method:  
navigationController?.navigationBar.barStyle = .black // this will give you a white status bar
```

`UIBarStyle` Optionen sind `default`, `black`, `blackOpaque`, `blackTranslucent`. Die letzten drei sollten Ihnen eine Statusleiste mit weißem Text geben, nur die letzten beiden geben die Deckkraft der Leiste an.

Hinweis: Sie können das Erscheinungsbild Ihrer Navigationsleiste weiterhin beliebig ändern.

Wenn Sie den Code von ViewController nicht ändern können

Wenn Sie eine Bibliothek verwenden, die `AwesomeViewController` mit einer falschen Statusleistenfarbe enthält, können Sie Folgendes versuchen:

```
let awesomeViewController = AwesomeViewController()  
awesomeViewController.navigationBar.barStyle = .blackTranslucent // or other style
```

Zur ViewController-Eindämmung

Wenn Sie `UIViewControllerContainment` gibt es ein paar andere Methoden, die es wert sind, betrachtet zu werden.

Wenn Sie möchten, dass ein untergeordneter `viewController` die Darstellung der Statusleiste steuert (dh, ob das untergeordnete Element oben auf dem Bildschirm angezeigt wird)

in Swift

```
class RootViewController: UIViewController {  
  
    private let messageBarViewController = MessageBarViewController()  
  
    override func childViewControllerForStatusBarStyle() -> UIViewController? {  
        return messageBarViewController  
    }  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        //add child vc code here...  
  
        setNeedsStatusBarAppearanceUpdate()  
    }  
}  
  
class MessageBarViewController: UIViewController {  
  
    override func preferredStatusBarStyle() -> UIStatusBarStyle {  
        return .Default  
    }  
}
```

Statusleistenstil für die gesamte Anwendung ändern

SCHNELL:

Schritt 1:

Fügen Sie in Ihrer **Info.plist** das folgende Attribut hinzu:

```
View controller-based status bar appearance
```

und seinen Wert auf

```
NO
```

wie im Bild unten beschrieben:

Key	Type	Value
▼ Information Property List	Dictionary	(15 items)
View controller-based status...	Boolean	NO

Schritt 2:

Fügen Sie in Ihrer **AppDelegate.swift**- Datei in der Methode `didFinishLaunchingWithOptions` diesen Code hinzu:

```
UIApplication.shared.statusBarStyle = .lightContent
```

oder

```
UIApplication.shared.statusBarStyle = .default
```

- Die **.lightContent**- Option setzt die Farbe der **Statusleiste** für die gesamte App auf Weiß.
- Die Option **.default** setzt die Farbe der **Statusleiste** für die gesamte App auf die ursprüngliche schwarze Farbe.

ZIEL C:

Folgen Sie dem ersten Schritt aus dem **SWIFT-** Bereich. **Fügen** Sie diesen Code dann der **AppDelegate.m-** Datei hinzu:

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];
```

oder

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleDefault];
```

Farbe der Statusleiste ändern online lesen: <https://riptutorial.com/de/ios/topic/378/farbe-der-statusleiste-andern>

Kapitel 60: FCM Messaging in Swift

Bemerkungen

FCM: <https://firebase.google.com/docs/cloud-messaging/ios/client>

Examples

FCM in Swift initialisieren

Folgen Sie dem nachstehenden Schritt, um FCM in Ihr schnelles Projekt einzufügen

1- Wenn Sie noch kein Xcode-Projekt haben, erstellen Sie jetzt eines. Erstellen Sie eine Pod-Datei, wenn Sie keine haben:

```
$ cd Ihr Projektverzeichnis  
$ pod init
```

2- Fügen Sie die Pods hinzu, die Sie installieren möchten. Sie können einen Pod so in Ihre Pod-Datei einfügen:

```
pod 'Firebase / Core'  
pod 'Firebase / Messaging'
```

3- Installieren Sie die Pods und öffnen Sie die .xcworkspace-Datei, um das Projekt in Xcode anzuzeigen.

```
$ pod installieren  
$ öffnen Sie Ihren Projekt.xcworkspace
```

4- Laden Sie eine GoogleService-Info.plist-Datei von [plist](#) herunter und fügen Sie sie in Ihre App ein.

5- Laden Sie das APNs-Zertifikat in Firebase hoch. [APN Cert](#)

6- Fügen Sie "import Firebase" in Ihre appDelegate-Datei des Projekts ein

7- fügen Sie "FIRApp.configure ()" in Ihre "Anwendung: didFinishLaunchingWithOptions" hinzu

8- Registrieren Sie sich für die Fernbenachrichtigung

```
if #available(iOS 10.0, *) {  
    let authOptions : UNAuthorizationOptions = [.Alert, .Badge, .Sound]  
    UNUserNotificationCenter.currentNotificationCenter().requestAuthorizationWithOptions(  
        authOptions,  
        completionHandler: {_,_ in })  
  
    // For iOS 10 display notification (sent via APNS)
```

```

UNUserNotificationCenter.currentNotificationCenter().delegate = self
// For iOS 10 data message (sent via FCM)
FIRMessaging.messaging().remoteMessageDelegate = self

} else {
    let settings: UIUserNotificationSettings =
    UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
    application.registerUserNotificationSettings(settings)
}

application.registerForRemoteNotifications()

```

9- um die Registrierung des Token zu erhalten

```
let token = FIRInstanceID.instanceID().token()!
```

10 - und wenn Sie einen Token-Änderungsmonitor verwenden möchten, verwenden Sie den folgenden Code in der appDelegate-Datei

```

func tokenRefreshNotification(notification: NSNotification) {
    if let refreshedToken = FIRInstanceID.instanceID().token() {
        print("InstanceID token: \(refreshedToken)")
    }

    // Connect to FCM since connection may have failed when attempted before having a token.
    connectToFcm()
}

```

11- um eine Nachricht von fcm zu erhalten, fügen Sie den folgenden Code in appDelegate hinzu

```

func connectToFcm() {
    FIRMessaging.messaging().connectWithCompletion { (error) in
        if (error != nil) {
            print("Unable to connect with FCM. \(error)")
        } else {
            print("Connected to FCM.")
        }
    }
}

```

12- und zum Trennen verwenden

```

func applicationDidEnterBackground(application: UIApplication) {
    FIRMessaging.messaging().disconnect()
    print("Disconnected from FCM.")
}

```

in deiner appDelegate.

Die Initialisierung ist abgeschlossen und der Client kann Nachrichten vom fcm-Panel empfangen oder per Token von einem Drittanbieter-Server senden

FCM Messaging in Swift online lesen: <https://riptutorial.com/de/ios/topic/7326/fcm-messaging-in-swift>

Kapitel 61: FileHandle

Einführung

Datei in Stücken aus dem Dokumentverzeichnis lesen

Examples

Lesen Sie die Datei aus dem Dokumentverzeichnis in Stücken

Ich erhalte den Dateipfad aus dem Dokumentverzeichnis und lese diese Datei in 1024-`NSData` und speichere das `NSData` Objekt (anfügen) oder Sie können direkt in den Socket schreiben.

```
// MARK: - Get file data as chunks Methode.
func getFileDataInChunks() {

    let documentDirectoryPath = NSSearchPathForDirectoriesInDomains(.documentDirectory,
        .userDomainMask, true)[0] as NSString
    let filePath = documentDirectoryPath.appendingPathComponent("video.mp4")

    //Check file exists at path or not.
    if FileManager.default.fileExists(atPath: filePath) {

        let chunkSize = 1024 // divide data into 1 kb

        //Create NSData object to save read data.
        let ReadData = NSMutableData()

        do {

            //open file for reading.
            outputFileHandle = try FileHandle(forReadingFrom: URL(fileURLWithPath: filePath))

            // get the first chunk
            var datas = outputFileHandle?.readData(ofLength: chunkSize)

            //check next chunk is empty or not.
            while !(datas?.isEmpty)! {

                //here I write chunk data to ReadData or you can directly write to socket.
                ReadData.append(datas!)

                // get the next chunk
                datas = outputFileHandle?.readData(ofLength: chunkSize)

                print("Running: \(ReadData.length)")
            }

            //close outputFileHandle after reading data complete.
            outputFileHandle?.closeFile()

            print("File reading complete")
        }
    }
}
```

```
        }catch let error as NSError {
            print("Error : \(error.localizedDescription)")
        }
    }
}
```

Nach dem Lesen der Datei erhalten Sie die Datei Daten in der `ReadData` Variablen. Hier ist `outputFileHandle` ein Objekt von `FileHandle`

```
var outputFileHandle:FileHandle?
```

FileHandle online lesen: <https://riptutorial.com/de/ios/topic/10665/filehandle>

Kapitel 62: GameCenter-Bestenlisten

Examples

GameCenter-Bestenlisten

Voraussetzungen:

1. Apple-Entwicklerkonto
2. Richten Sie GameCenter-Bestenlisten mit iTunesConnect ein

Einrichten von GameCenter-Bestenlisten:

1. *Melden Sie sich bei iTunesConnect an*
2. Gehen Sie zu *Meine Apps* . Erstellen Sie eine App für Ihr Projekt und gehen Sie zu *Features* .
3. Klicken Sie auf *Game Center*
4. Klicken Sie auf das Pluszeichen neben Ranglisten.
5. Wählen Sie *Single Leaderboard* für Leaderboard-Typen.
6. Erstellen Sie einen *Leaderboard-Referenznamen als Referenz*.
7. Erstellen Sie eine *Leaderboard-ID* für Ihre App, auf die bei der Berichterstellung von Scores verwiesen werden soll.
8. Legen Sie das Score-Format auf *Integer fest*
9. Score Submission wird *Best Score*
10. Klicken Sie auf *Sprache hinzufügen* und füllen Sie die Einträge aus.

Kopieren Sie Ihre `LeaderboardID` , die Sie erstellt haben, und lassen Sie sich zu Xcode übergehen.

Mit Xcode arbeiten

Es gibt 4 Funktionen, mit denen wir arbeiten werden.

1. Framework importieren und Protokolle einrichten
2. Überprüfen, ob der Benutzer bei GameCenter angemeldet ist
3. Die Ergebnisse an GameCenter melden
4. Bestenlisten anzeigen
5. Import GameKit `import GameKit` Protokolle `GKGameCenterControllerDelegate`
6. Jetzt möchten wir überprüfen, ob der Benutzer bei GameCenter angemeldet ist

```
func authenticateLocalPlayer() {  
  
    let localPlayer = GKLocalPlayer.localPlayer()  
    localPlayer.authenticateHandler = { (viewController, error) -> Void in
```



```

    if viewController != nil {
        //If the user is not signed in to GameCenter, we make them sign in
        let vc:UIViewController = self.view!.window!.rootViewController!
        vc.presentViewController(viewController!, animated: true, completion: nil)

    } else {

        //Do something here if you want
    }
}
}

```

3. Jetzt verwendet der Benutzer die App und plötzlich hat der Benutzer einen neuen Highscore. Wir melden den Highscore, indem wir die Funktion unten aufrufen.

Die Funktion unterhält 2 Parameter.

`Identifizier` der als Zeichenfolge definiert ist und zur Eingabe Ihrer in iTunesConnect erstellten `LeaderboardID` verwendet wird.

`score` der als `Int` definiert ist, wobei der Score des Benutzers an iTunesConnect übermittelt wird

```

func saveHighScore(identifizier:String, score:Int) {

    if GKLocalPlayer.localPlayer().authenticated {

        let scoreReporter = GKScore(leaderboardIdentifizier: identifizier)

        scoreReporter.value = Int64(score)

        let scoreArray:[GKScore] = [scoreReporter]

        GKScore.reportScores(scoreArray, withCompletionHandler: {
            error -> Void in

            if error != nil {
                print("Error")
            } else {

            }

        })
    }
}
}

```

4. Wenn der Benutzer die Bestenlisten anzeigen möchte, rufen Sie die folgende Funktion auf

```

//This function will show GameCenter leaderboards and Achievements if you call this function.
func showGameCenter() {

    let gameCenterViewController = GKGameCenterViewController()
    gameCenterViewController.gameCenterDelegate = self

    let vc:UIViewController = self.view!.window!.rootViewController!
    vc.presentViewController(gameCenterViewController, animated: true, completion:nil)
}

```

```
}  
  
//This function closes gameCenter after showing.  
func gameCenterViewControllerDidFinish(gameCenterViewController:  
GKGameCenterViewController) {  
  
    gameCenterViewController.dismissViewControllerAnimated(true, completion: nil)  
    self.gameCenterAchievements.removeAll()  
  
}
```

GameCenter-Bestenlisten online lesen: <https://riptutorial.com/de/ios/topic/6720/gamecenter-bestenlisten>

Kapitel 63: GameplayKit

Examples

Zufallszahlen generieren

Obwohl es sich bei `GameplayKit` (das mit iOS 9 SDK eingeführt wird) um die Implementierung von Spiellogik handelt, kann es auch zur Erzeugung von Zufallszahlen verwendet werden, was in Apps und Spielen sehr nützlich ist.

Neben `GKRandomSource.sharedRandom` das in den folgenden Kapiteln verwendet wird, gibt es drei zusätzliche Typen von `GKRandomSource` 's out-of-the-box.

- **GKARC4RandomSource** Verwendet den ARC4-Algorithmus
- **GKLinearCongruentialRandomSource** Dies ist eine schnelle, aber nicht zufällige `GKRandomSource`
- **GKMersenneTwisterRandomSource** Die Implementierung eines MersenneTwister-Algorithmus. Es ist langsamer aber zufällig.

Im folgenden Kapitel verwenden wir nur die `nextInt()` Methode einer `GKRandomSource`. Dazu kommen `nextBool() -> Bool` und `nextUniform() -> Float`

Generation

Importiere zuerst `GameplayKit`:

Schnell

```
import GameplayKit
```

Ziel c

```
#import <GameplayKit/GameplayKit.h>
```

Verwenden Sie dann diesen Code, um eine Zufallszahl zu generieren:

Schnell

```
let randomNumber = GKRandomSource.sharedRandom().nextInt()
```

Ziel c

```
int randomNumber = [[GKRandomSource sharedRandom] nextInt];
```

Hinweis

Wenn die `nextInt()` - Funktion ohne Parameter verwendet wird, wird eine Zufallszahl zwischen -2.147.483.648 und 2.147.483.647 einschließlich ihrer zurückgegeben. Daher sind wir nicht sicher, dass es sich immer um eine positive oder eine ungleich Null handelt.

Eine Zahl von 0 bis n erzeugen

Um dies zu erreichen, sollten Sie der `nextIntWithUpperBound()` Methode `n` geben:

Schnell

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 10)
```

Ziel c

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 10];
```

Dieser Code gibt uns eine Zahl zwischen 0 und 10, einschließlich sich selbst.

Eine Zahl von m bis n generieren

Dazu erstellen Sie ein `GKRandomDistribution` Objekt mit einer `GKRandomSource` und übergeben die Grenzen. Eine `GKRandomDistribution` kann verwendet werden, um das Verteilungsverhalten wie `GKGaussianDistribution` oder `GKShuffledDistribution` zu ändern.

Danach kann das Objekt wie jede reguläre `GKRandomSource` da es auch das `GKRandom` Protokoll implementiert.

Schnell

```
let randomizer = GKRandomDistribution(randomSource: GKRandomSource(), lowestValue: 0, highestValue: 6)
let randomNumberInBounds = randomizer.nextInt()
```

Objective-C veraltet

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: n - m] + m;
```

Um beispielsweise eine Zufallszahl zwischen 3 und 10 zu generieren, verwenden Sie diesen Code:

Schnell

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 7) + 3
```

Objective-C *veraltet*

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 7] + 3;
```

GKEntity und GKComponent

Eine Entität repräsentiert ein Objekt eines Spiels wie eine Spielerfigur oder eine gegnerische Figur. Da dieses Objekt ohne Arme und Beine nicht viel ausmacht, können wir die Komponenten dazu hinzufügen. Um dieses System zu erstellen, verfügt Apple über die Klassen `GKEntity` und `GKComponent`.

Nehmen wir an, wir haben die folgende Klasse für die folgenden Kapitel:

```
class Player: GKEntity{}
class PlayerSpriteComponent: GKComponent {}
```

GKEntity

Eine Entität ist eine Sammlung von Komponenten und bietet verschiedene Funktionen zum Hinzufügen, Entfernen und Interagieren mit Komponenten davon.

Während wir die `GKEntity` einfach verwenden könnten, ist es üblich, sie für einen bestimmten Typ von Spieleinheiten zu subclassieren.

Es ist wichtig, dass eine Komponente nur einmal hinzugefügt werden kann. Wenn Sie eine zweite Komponente derselben Klasse hinzufügen, überschreibt sie die erste vorhandene Komponente in `GKEntity`

```
let otherComponent = PlayerSpriteComponent()
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.addComponent(otherComponent)
print(player.components.count) //will print 1
print(player.components[0] === otherComponent) // will print true
```

Sie fragen vielleicht warum. Der Grund dafür sind die als `component(for: T.Type)` Methoden, die die Komponente eines bestimmten Entitätstyps zurückgeben.

```
let component = player.component(ofType: PlayerSpriteComponent.self)
```

Neben den Komponenten-Methoden gibt es eine `update`, mit der die Deltazeit oder die aktuelle Zeit der Spiellogik an ihre Komponenten delegiert wird.

```
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.update(deltaTime: 1.0) // will call the update method of the PlayerSpriteComponent
added to it
```

GKComponent

Eine Komponente stellt etwas von einer Entität dar, beispielsweise die visuelle Komponente oder die Logikkomponente.

Wenn eine Aktualisierungsmethode einer Entität aufgerufen wird, wird sie an alle ihre Komponenten delegiert. Das Überschreiben dieser Methode wird zum Manipulieren einer Entität verwendet.

```
class PlayerSpriteComponent: GKComponent {
    override func update(deltaTime seconds: TimeInterval) {
        //move the sprite depending on the update time
    }
}
```

Darüber hinaus ist es möglich, die Methode `didAddToEntity` und `willRemoveFromEntity` zu überschreiben, um andere Komponenten über das Entfernen oder Hinzufügen zu informieren.

Um eine andere Komponente innerhalb einer Komponente zu bearbeiten, ist es möglich, die `GKEntity` abzurufen, zu der die Komponente hinzugefügt wird.

```
override func update(deltaTime seconds: TimeInterval) {
    let controller = self.entity?.component(ofType: PlayerControlComponent.self)
    //call methods on the controller
}
```

Während dies möglich ist, ist es **kein** allgemeines Muster, da es die beiden Komponenten miteinander verbindet.

GKComponentSystem

Während wir gerade über die Verwendung des Aktualisierungsdelegationsmechanismus von `GKEntity` zum Aktualisieren der `GKComponents` gibt es eine andere Möglichkeit, `GKComponents` zu aktualisieren, die als `GKComponentSystem`.

Es wird verwendet, wenn alle Komponenten eines bestimmten Typs auf einmal aktualisiert werden

müssen.

Ein `GKComponentSystem` wird für einen bestimmten Komponententyp erstellt.

```
let system = GKComponentSystem(componentClass: PlayerSpriteComponent.self)
```

Um eine Komponente hinzuzufügen, können Sie die `add`-Methode verwenden:

```
system.addComponent(PlayerSpriteComponent())
```

Eine üblichere Methode besteht jedoch darin, die erstellte Entität mit ihren Komponenten an das `GKComponentSystem` und es wird eine passende Komponente innerhalb der Entität gefunden.

```
system.addComponent(foundIn: player)
```

Um alle Komponenten eines bestimmten Typs zu aktualisieren, rufen Sie das `Update` auf:

```
system.update(deltaTime: delta)
```

Sie das `GKComponentSystem` anstelle eines `GKComponentSystem` Aktualisierungsmechanismus verwenden `GKComponentSystem`, müssen `GKComponentSystem` für jede Komponente ein `GKComponentSystem` und das `Update` auf allen Systemen aufrufen.

GameplayKit online lesen: <https://riptutorial.com/de/ios/topic/4966/gameplaykit>

Kapitel 64: GCD (Grand Central Dispatch)

Einführung

Grand Central Dispatch (GCD) ist Apples Antwort auf Multithreading. Es handelt sich um ein einfaches Framework, um Aufgaben synchron oder asynchron in Warteschlangen auszuführen und CPU-Threads für Sie hinter den Kulissen zu verarbeiten.

Verwandte Themen: [Parallelität](#)

Examples

Erstellen Sie eine Versandwarteschlange

Sie können Ihre eigene Warteschlange mit `dispatch_queue_create` erstellen

Ziel c

```
dispatch_queue_t queue = dispatch_queue_create("com.example.myqueue", DISPATCH_QUEUE_SERIAL);
```

Schnell

```
// Before Swift 3
let queue = dispatch_queue_create("com.example.myqueue", DISPATCH_QUEUE_SERIAL)
// Swift 3
let queue = DispatchQueue(label: "com.example.myqueue") //default is serial queue, unless
.concurrent is specified as an attribute otherwise
```

Die Hauptwarteschlange abrufen

Die Hauptwarteschlange ist die Versandwarteschlange, in der alle Aktualisierungen der Benutzeroberfläche stattfinden und der Code, der Änderungen an der Benutzeroberfläche beinhaltet, platziert wird.

Sie müssen zur Hauptwarteschlange gelangen, um die Benutzeroberfläche nach Abschluss eines asynchronen Prozesses wie `NSURLSession`

Es gibt zwei Arten von Hauptwarteschlangenaufrufen, `synchronous` und `asynchronous`. Wenn Sie etwas `synchronously` aufrufen, bedeutet dies, dass der Thread, der diesen Vorgang initiiert hat, auf die Beendigung der Aufgabe wartet, bevor Sie fortfahren. `Asynchronous` bedeutet, dass es nicht wartet.

Code Objective-C

`Synchronous` Aufruf der Hauptwarteschlange


```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
```

Asynchroner Aufruf der Hauptwarteschlange

```
dispatch_async(dispatch_get_main_queue(), ^{  
    // do work here to Usually to update the User Interface  
});
```

SWIFT 3

Asynchroner Aufruf der Hauptwarteschlange

```
DispatchQueue.main.async {  
  
}
```

Synchroner Aufruf der Hauptwarteschlange

```
DispatchQueue.main.sync {  
  
}
```

Versandgruppe

DispatchGroup ermöglicht die Aggregation der Arbeitssynchronisation. Sie können sie verwenden, um mehrere verschiedene Arbeitselemente zu senden und zu verfolgen, wann alle abgeschlossen sind, auch wenn sie in verschiedenen Warteschlangen ausgeführt werden. Dieses Verhalten kann hilfreich sein, wenn keine Fortschritte erzielt werden können, bis alle angegebenen Aufgaben abgeschlossen sind.

Ein Szenario, in dem dies nützlich sein kann, ist, wenn Sie mehrere Webservice-Aufrufe haben, die alle beendet werden müssen, bevor Sie fortfahren. Beispielsweise müssen Sie mehrere Datensätze herunterladen, die von einer Funktion verarbeitet werden müssen. Sie müssen warten, bis alle Webservices abgeschlossen sind, bevor Sie die Funktion aufrufen, um alle empfangenen Daten zu verarbeiten.

Swift 3

```
func doLongTasksAndWait () {  
    print("starting long running tasks")  
    let group = DispatchGroup() //create a group for a bunch of tasks we are about to  
    do  
    for i in 0...3 { //launch a bunch of tasks (eg a bunch of webservice  
calls that all need to be finished before proceeding to the next ViewController)  
        group.enter() //let the group know that something is being added  
        DispatchQueue.global().async { //run tasks on a background thread  
            sleep(arc4random() % 4) //do some long task eg webservice or database lookup  
(here we are just sleeping for a random amount of time for demonstration purposes)  
            print("long task \(i) done!")  
            group.leave() //let group know that the task is finished  
        }  
    }  
}
```

```

}
group.wait() //will block whatever thread we are on here until all
the above tasks have finished (so maybe dont use this function on your main thread)
print("all tasks done!")
}

```

Wenn Sie nicht warten möchten, bis die Gruppen abgeschlossen sind, sondern stattdessen eine Funktion ausführen möchten, nachdem alle Aufgaben abgeschlossen sind, verwenden Sie die `notify` anstelle der `group.wait()`

```

group.notify(queue: DispatchQueue.main) { //the queue: parameter is which queue this block
will run on, if you need to do UI updates, use the main queue
    print("all tasks done!") //this will execute when all tasks have left the
group
}

```

Beispielausgabe:

```

starting long running tasks
long task 0 done!
long task 3 done!
long task 1 done!
long task 2 done!
all tasks done!

```

Weitere Informationen finden Sie in den [Apple Docs](#) oder im verwandten [Thema](#)

Dispatch Semaphore

`DispatchSemaphore` bietet eine effiziente Implementierung eines traditionellen Zählsemaphors, mit dem der Zugriff auf eine Ressource über mehrere Ausführungskontexte gesteuert werden kann.

Ein Szenario für die Verwendung eines Semaphors kann beispielsweise beim Lesen / Schreiben von Dateien auftreten. Wenn mehrere Tasks gleichzeitig aus einer Datei lesen und schreiben möchten, kann dies die Leistung erhöhen, wenn jede Task warten muss um den E / A-Controller nicht zu überlasten.

Swift 3

```

func do2TasksAtATime () {
    print("starting long running tasks (2 at a time)")
    let sem = DispatchSemaphore(value: 2) //this semaphore only allows 2 tasks to
run at the same time (the resource count)
    for i in 0...7 { //launch a bunch of tasks
        DispatchQueue.global().async { //run tasks on a background thread
            sem.wait() //wait here if no resources available
            sleep(2) //do some long task eg file access (here
we are just sleeping for a 2 seconds for demonstration purposes)
            print("long task \(i) done! \(Date())")
            sem.signal() //let the semaphore know this resource is
now available
        }
    }
}

```

```
}  
}
```

Beispielausgabe: (Beachten Sie die Zeitstempel)

```
starting long running tasks (2 at a time)  
long task 0 done! 2017-02-16 07:11:53 +0000  
long task 1 done! 2017-02-16 07:11:53 +0000  
long task 2 done! 2017-02-16 07:11:55 +0000  
long task 3 done! 2017-02-16 07:11:55 +0000  
long task 5 done! 2017-02-16 07:11:57 +0000  
long task 4 done! 2017-02-16 07:11:57 +0000  
long task 6 done! 2017-02-16 07:11:59 +0000  
long task 7 done! 2017-02-16 07:11:59 +0000
```

Weitere Informationen finden Sie in den [Apple Docs](#)

Serielle versus gleichzeitige Dispatch-Warteschlangen

Swift 3

Serienwarteschlange

```
func serialQueues () {  
    let serialQueue = DispatchQueue(label: "com.example.serial") //default queue type is a  
    serial queue  
    let start = Date ()  
    for i in 0...3 {  
        serialQueue.async { //launch a bunch of tasks  
            //run tasks on a background  
            thread, using our serial queue  
                sleep(2) //do some long task eg  
                webservice or database lookup  
                let timeTaken = Date().timeIntervalSince(start)  
                print("serial long task \(i) done! total time taken: \(timeTaken)")  
        }  
    }  
}
```

Beispielausgabe:

```
serial long task 0 done! total time taken: 2.07241100072861  
serial long task 1 done! total time taken: 4.16347700357437  
serial long task 2 done! total time taken: 6.23209798336029  
serial long task 3 done! total time taken: 8.30682599544525
```

Gleichzeitige Warteschlange

```
func concurrentQueues () {  
    let concurrentQueue = DispatchQueue(label: "com.example.concurrent", attributes:  
    .concurrent) //explicitly specify the queue to be a concurrent queue  
    let start = Date ()  
    for i in 0...3 { //launch a bunch of tasks  
        concurrentQueue.async { //run tasks on a background thread, using our concurrent queue  
            sleep(2) //do some long task eg webservice or database lookup
```

```
        let timeTaken = Date().timeIntervalSince(start)
        print("concurrent long task \(i) done! total time taken: \(timeTaken)")
    }
}
}
```

Beispielausgabe:

```
concurrent long task 3 done! total time taken: 2.07092100381851
concurrent long task 0 done! total time taken: 2.07087397575378
concurrent long task 2 done! total time taken: 2.07086700201035
concurrent long task 1 done! total time taken: 2.07089096307755
```

Diskussion

Wie aus den obigen Beispielen ersichtlich, wird eine serielle Warteschlange jede Aufgabe in der Reihenfolge abschließen, in der sie an die Warteschlange übergeben werden. Jede Aufgabe wartet, bis die vorherige Aufgabe abgeschlossen ist, bevor sie ausgeführt wird. Bei der gleichzeitigen Warteschlange wartet jede Aufgabe nicht auf die anderen in der Warteschlange und wird so bald wie möglich ausgeführt. Der Vorteil ist, dass alle Tasks in der Warteschlange gleichzeitig in separaten Threads ausgeführt werden, sodass eine gleichzeitige Warteschlange weniger Zeit als eine serielle Warteschlange benötigt.

Wenn die Reihenfolge der Ausführung von Aufgaben nicht wichtig ist, verwenden Sie immer eine gleichzeitige Warteschlange, um die beste Effizienz zu erzielen.

GCD (Grand Central Dispatch) online lesen: <https://riptutorial.com/de/ios/topic/4626/gcd--grand-central-dispatch->

Kapitel 65: Gesichtserkennung mit CoreImage / OpenCV

Examples

Gesichts- und Funktionserkennung

Ziel c

Importieren Sie Folgendes in Ihren ViewController

```
#import <CoreImage/CoreImage.h>
#import <CoreImage/CoreImage.h>
#import <QuartzCore/QuartzCore.h>
```

Rufen Sie die Funktion auf

```
[self faceDetector];
```

Funktionsdefinition:

```
-(void)faceDetector
{
    // Load the picture for face detection
    UIImageView* image = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"download.jpeg"]];

    // Draw the face detection image
    [self.view addSubview:image];

    // Execute the method used to markFaces in background
    [self performSelectorInBackground:@selector(markFaces:) withObject:image];

    // flip image on y-axis to match coordinate system used by core image
    [image setTransform:CGAffineTransformMakeScale(1, -1)];

    // flip the entire window to make everything right side up
    [self.view setTransform:CGAffineTransformMakeScale(1, -1)];
}
```

Gesichtsfunktion markieren

```
//Adds face squares and color masks to eyes and mouth
-(void)markFaces:(UIImageView *)facePicture
{
    // draw a CI image with the previously loaded face detection picture
    CIImage* image = [CIImage imageWithCGImage:facePicture.image.CGImage];
```

```

// create a face detector - since speed is not an issue we'll use a high accuracy
// detector
CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeFace
                        context:nil options:[NSDictionary
dictionaryWithObject:CIDetectorAccuracyHigh forKey:CIDetectorAccuracy]];

// create an array containing all the detected faces from the detector
NSArray* features = [detector featuresInImage:image];
NSLog(@"Number of faces %d",[features count]);

// we'll iterate through every detected face. CIFaceFeature provides us
// with the width for the entire face, and the coordinates of each eye
// and the mouth if detected. Also provided are BOOL's for the eye's and
// mouth so we can check if they already exist.
// for (features in image)
// {
for(CIFaceFeature* faceFeature in features)
{
    // get the width of the face
    CGFloat faceWidth = faceFeature.bounds.size.width;

    // create a UIView using the bounds of the face
    UIView* faceView = [[UIView alloc] initWithFrame:faceFeature.bounds];

    // add a border around the newly created UIView
    faceView.layer.borderWidth = 1;
    faceView.layer.borderColor = [[UIColor redColor] CGColor];

    // add the new view to create a box around the face
    [self.view addSubview:faceView];

    if(faceFeature.hasLeftEyePosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* leftEyeView = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.leftEyePosition.x-faceWidth*0.15,
faceFeature.leftEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
        // change the background color of the eye view
        [leftEyeView setBackgroundColor:[UIColor blueColor]
colorWithAlphaComponent:0.3]];
        // set the position of the leftEyeView based on the face
        [leftEyeView setCenter:faceFeature.leftEyePosition];
        // round the corners
        leftEyeView.layer.cornerRadius = faceWidth*0.15;
        // add the view to the window
        [self.view addSubview:leftEyeView];
    }

    if(faceFeature.hasRightEyePosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* leftEye = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.rightEyePosition.x-faceWidth*0.15,
faceFeature.rightEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
        // change the background color of the eye view
        [leftEye setBackgroundColor:[UIColor blueColor] colorWithAlphaComponent:0.3]];
        // set the position of the rightEyeView based on the face
        [leftEye setCenter:faceFeature.rightEyePosition];
        // round the corners
        leftEye.layer.cornerRadius = faceWidth*0.15;
        // add the new view to the window
    }
}

```

```

        [self.view addSubview:leftEye];
    }

    if(faceFeature.hasMouthPosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* mouth = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.mouthPosition.x-faceWidth*0.2,
faceFeature.mouthPosition.y-faceWidth*0.2, faceWidth*0.4, faceWidth*0.4)];
        // change the background color for the mouth to green
        [mouth setBackgroundColor:[UIColor greenColor] colorWithAlphaComponent:0.3];
        // set the position of the mouthView based on the face
        [mouth setCenter:faceFeature.mouthPosition];
        // round the corners
        mouth.layer.cornerRadius = faceWidth*0.2;
        // add the new view to the window
        [self.view addSubview:mouth];
    }
}

// }
}

```

Der Simulator ScreenShot für die Funktion



Gesichtserkennung mit CoreImage / OpenCV online lesen:

<https://riptutorial.com/de/ios/topic/7298/gesichtserkennung-mit-coreimage---opencv>

Kapitel 66: Graph (Coreplot)

Examples

Erstellen von Diagrammen mit CorePlot

Core Plot bietet eine Podspec-Funktion, mit der Sie Cocoapods als Bibliotheksmanager verwenden können, was die Installation und Aktualisierung erheblich vereinfacht

Installieren Sie Cocoapods auf Ihrem System

Fügen Sie im Projektverzeichnis eine Textdatei mit dem Namen Podfile hinzu, indem Sie `pod init` in Ihr Projektverzeichnis eingeben

Fügen Sie im Podfile den Zeilen-Pod 'CorePlot', '~> 1.6' hinzu.

Wechseln Sie im Terminal in Ihr Projektverzeichnis und führen Sie `pod install` aus

Cocoapods generiert eine `xcworkspace`-Datei, die Sie zum Starten Ihres Projekts verwenden sollten (die `.xcodeproj`-Datei enthält keine Pod-Bibliotheken).

Öffnen Sie den von CocoaPods generierten `.xcworkspace`

In der `ViewController.h`-Datei

```
#import <CorePlot/ios/CorePlot.h>
//#import "CorePlot-CocoaTouch.h" or the above import statement
@interface ViewController : UIViewController<CPTPlotDataSource>
```

In der `ViewController.m`-Datei

```
-(void)loadView
{
    [super loadView];
    // We need a hostview, you can create one in IB (and create an outlet) or just do this:
    CPTGraphHostingView* hostView = [[CPTGraphHostingView alloc] initWithFrame:CGRectMake(10,
    40, 300, 400)];
    hostView.backgroundColor=[UIColor whiteColor];
    self.view.backgroundColor=[UIColor blackColor];
    [self.view addSubview: hostView];
    // Create a CPTGraph object and add to hostView
    CPTGraph* graph = [[CPTXYGraph alloc] initWithFrame:CGRectMake(10, 40, 300, 400)];
    hostView.hostedGraph = graph;
    // Get the (default) plotspace from the graph so we can set its x/y ranges
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    // Note that these CPTPlotRange are defined by START and LENGTH (not START and END) !!
    [plotSpace setYRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( 0 )
    length:CPTDecimalFromFloat( 20 )]];
    [plotSpace setXRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( -4 )
    length:CPTDecimalFromFloat( 8 )]];
    // Create the plot (we do not define actual x/y values yet, these will be supplied by the
    datasource...)
```

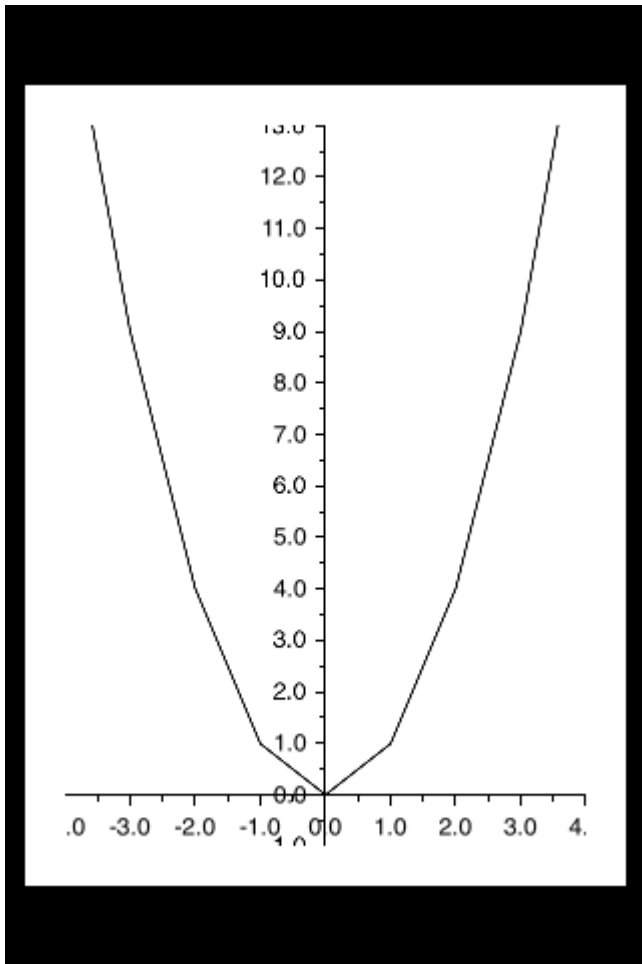


```

    CPTScatterPlot* plot = [[CPTScatterPlot alloc] initWithFrame:CGRectZero];
    // Let's keep it simple and let this class act as datasource (therefore we implemtn
    <CPTPlotDataSource>)
    plot.dataSource = self;
    // Finally, add the created plot to the default plot space of the CPTGraph object we
    created before
    [graph addPlot:plot toPlotSpace:graph.defaultPlotSpace];
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plotnumberOfRecords
{
    return 9; // Our sample graph contains 9 'points'
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSNumber *)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
recordIndex:(NSUInteger)index
{
    // We need to provide an X or Y (this method will be called for each) value for every
    index
    int x = index - 4;
    // This method is actually called twice per point in the plot, one for the X and one for
    the Y value
    if(fieldEnum == CPTScatterPlotFieldX)
    {
        // Return x value, which will, depending on index, be between -4 to 4
        return [NSNumber numberWithInt: x];
    } else
    {
        // Return y value, for this example we'll be plotting y = x * x
        return [NSNumber numberWithInt: x * x];
    }
}
}

```

Die generierte Ausgabe ist wie folgt:



Graph (Coreplot) online lesen: <https://riptutorial.com/de/ios/topic/7302/graph--coreplot->

Kapitel 67: Größenklassen und Adaptivität

Bemerkungen

Beachten Sie beim Erstellen von anpassungsfähigen Apps die Einschränkungen der Größenklassen: Sie sind *Verallgemeinerungen* und keine spezifischen Anleitungen für genaue Pixelgrößen oder Geräte. Versuchen Sie niemals zu ermitteln, auf welchem Gerät Ihre App ausgeführt wird oder ob es sich im Split-Screen-Modus befindet, basierend auf den Größenklassen.

Treffen Sie stattdessen Layoutentscheidungen auf hoher Ebene für die Größenklasse und verwenden Sie das automatische Layout, um genaue Ansichtsrahmen zu ändern. (Siehe auch die UIViewController-Methode `viewWillTransition(to:with:)` für eine genauere Benachrichtigung darüber, wie groß die Ansicht eines Controllers nach einem Übergang ist.)

Examples

Trait-Sammlungen

In einer iOS-App kann Ihre Benutzeroberfläche einige allgemeine Formen und Größen annehmen. Diese werden mithilfe von **Größenklassen** definiert, die über die **Eigenschaftensammlung** eines View- oder View-Controllers verfügbar sind.

Apple definiert zwei Größenklassen: **normal** und **kompakt**. Jede dieser Größenklassen ist auf beiden Achsen des Geräts (**horizontal** und **vertikal**) verfügbar. Ihre App kann sich während ihrer gesamten Lebensdauer in einem dieser vier Zustände befinden. Als Abkürzung beschreiben Entwickler häufig eine Größenklassenkombination, indem sie die beiden Größenklassen mit der horizontalen Achse zuerst sagen oder schreiben: "Compact / Regular" beschreibt eine Schnittstelle, die horizontal kompakt, aber vertikal regelmäßig ist.

Verwenden Sie in Ihrer App Methoden für das UITraitEnvironment-Protokoll, um Ihre aktuelle Größenklasse zu überprüfen und auf Änderungen zu reagieren:

```
class MyViewController: UIViewController {
    override func viewDidLoad(_ animated: Bool) {
        super.viewDidLoad(animated)
        print("Horizontal size class: \(traitCollection.horizontalSizeClass)")
        print("Vertical size class: \(traitCollection.verticalSizeClass)")
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        print("Trait collection changed; size classes may be different.")
    }
}
```

Sowohl UIView als auch UIViewController sind mit UITraitEnvironment kompatibel, sodass Sie Ihre aktuelle Merkmalsauflistung anzeigen und Änderungen in Unterklassen von beiden

bearbeiten können.

Automatisches Layout mit Änderungen der Merkmalerfassung aktualisieren

Um eine App **anpassungsfähig zu machen**, d. H. Auf Änderungen der Größenklasse durch Ändern des Layouts zu reagieren, ist häufig eine große Hilfe durch das Auto Layout-System erforderlich. Eine der wichtigsten Möglichkeiten, mit der Apps anpassungsfähig werden, besteht darin, die aktiven Einschränkungen für das automatische Layout zu aktualisieren, wenn sich die Größenklasse einer Ansicht ändert.

Stellen Sie sich beispielsweise eine App vor, die eine `UIStackView` zum Anordnen von zwei `UILabels` verwendet. Wir möchten, dass diese Etiketten in horizontal kompakten Umgebungen übereinander gestapelt werden. Wenn sich jedoch in horizontal regulären Umgebungen etwas mehr Platz befindet, sitzen wir nebeneinander.

```
class ViewController: UIViewController {
    var stackView: UIStackView!

    override func viewDidLoad() {
        super.viewDidLoad()

        stackView = UIStackView()
        for text in ["foo", "bar"] {
            let label = UILabel()
            label.translatesAutoresizingMaskIntoConstraints = false
            label.text = text
            stackView.addArrangedSubview(label)
        }

        view.addSubview(stackView)
        stackView.translatesAutoresizingMaskIntoConstraints = false
        stackView.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        stackView.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        updateAxis(forTraitCollection: traitCollection)
    }

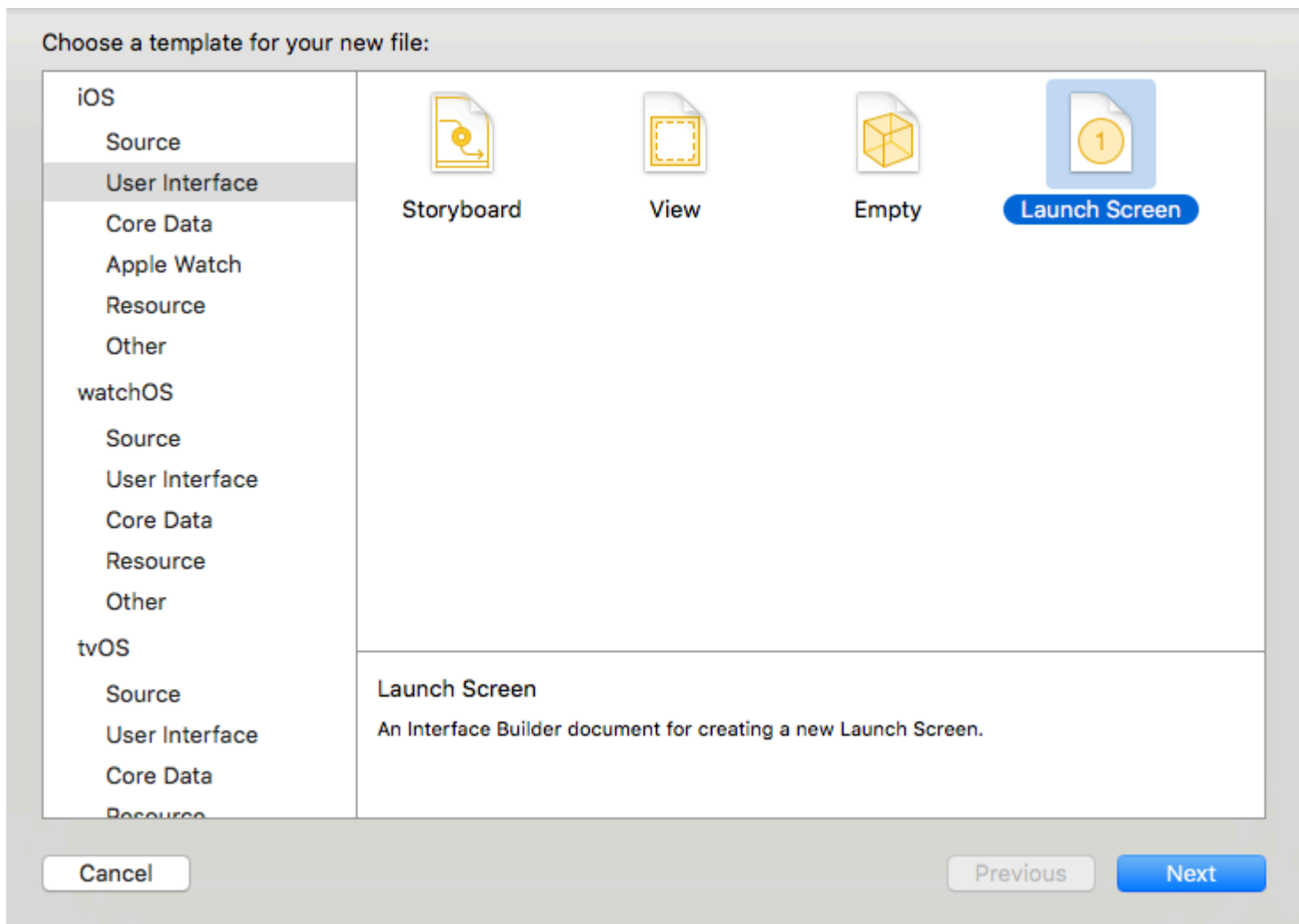
    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        updateAxis(forTraitCollection: traitCollection)
    }

    private func updateAxis(forTraitCollection traitCollection: UITraitCollection) {
        switch traitCollection.horizontalSizeClass {
            case .regular:
                stackView.axis = .horizontal
            case .compact:
                stackView.axis = .vertical
            case .unspecified:
                print("Unspecified size class!")
                stackView.axis = .horizontal
        }
    }
}
```

Unterstützung für iOS Multitasking auf dem iPad


Ein Schlüsselement der Adaptivität in einer modernen iOS-App ist die Unterstützung von Multitasking auf dem iPad. Standardmäßig werden in Xcode 7 und neuer erstellte Apps für die Unterstützung von Multitasking konfiguriert: Sie verfügen über eine LaunchScreen.storyboard-Datei, die Auto Layout verwendet.

Bestehende Apps können sich am einfachsten für Multitasking entscheiden, indem Sie ein solches Storyboard erstellen und es dann als Startbildschirm des Projekts festlegen:



App Icons Source  

Launch Images Source

Launch Screen File 

Wenn Ihre App das iPad-Multitasking unterstützt, prüfen Sie vorhandene Ansichten und Ansichtskontroller, um sicherzustellen, dass sie das automatische Layout verwenden und eine Vielzahl von Kombinationen von Größenklassen unterstützen.

Größenklassen und Adaptivität online lesen: <https://riptutorial.com/de/ios/topic/4628/größenklassen-und-adaptivität>

Kapitel 68: Größenklassen und Adaptivität

Bemerkungen

Für weitere Details (Größenklassen und Anpassungsfähigkeit durch Storyboard) zur Verwendung des automatischen Layouts für Anpassungsfähigkeit in iOS können Sie dem [Link](#) zur [Apple-Entwicklersite](#) folgen.

Wir können Einschränkungen auch **programmatisch** mit **Visual Format Language** hinzufügen, wie [hier auf der Apple-Entwickler-Site beschrieben](#) .

Examples

Größenklassen und Adaptivität durch Storyboard

Wir können jeder Unterklasse von `UIView` Anpassungsfähigkeit `UIView` die wir dem View-Controller in Nib-Datei hinzufügen.

Nehmen wir ein Beispiel für das Hinzufügen von Adaptivität mithilfe von Größenklassen zu einer Ansicht.

1. Fügen Sie eine Ansicht für den Ansichts-Controller hinzu als:



<https://riptutorial.com/de/ios/topic/6424/gro-enklassen-und-adaptivitat>

Kapitel 69: Grundlegende Textdatei-E / A

Examples

Lesen und schreiben Sie aus dem Ordner Dokumente

Swift 3

```
import UIKit

// Save String to file
let fileName = "TextFile"
let documentDirectory = try FileManager.default.urlForDirectory(.documentDirectory, in:
.userDomainMask, appropriateFor: nil, create: true)

var fileURL = try
documentDirectory.appendingPathComponent(fileName).appendingPathExtension("txt")

print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
do {
    // Write to file
    try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
} catch let error as NSError {
    print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
    fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
    print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

Schnell 2

```
import UIKit

// Save String to file
let fileName = "TextFile"
let DocumentDirectoryURL = try!
NSFileManager.defaultManager().URLForDirectory(.DocumentDirectory, inDomain: .UserDomainMask,
appropriateForURL: nil, create: true)

let fileURL =
DocumentDirectoryURL.URLByAppendingPathComponent(fileName).URLByAppendingPathExtension("txt")
print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
do {
    // Write to file
    try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
```

```
} catch let error as NSError {
    print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
    fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
    print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

Grundlegende Textdatei-E / A online lesen: <https://riptutorial.com/de/ios/topic/8892/grundlegende-textdatei-e---a>

Kapitel 70: Healthkit

Examples

HealthKit

Ziel c

Gehen `Target->Capabilities` zuerst auf `Target->Capabilities` und aktivieren Sie `HealthKit`. Dies würde den `info.plist`-Eintrag einrichten.

CocoaClass Sie eine neue CocoaClass vom Typ `NSObject` Der von `NSObject` Dateiname ist `GSHealthKitManager`

GSHealthKitManager.h

```
#import <Foundation/Foundation.h>
#import <HealthKit/HealthKit.h>
@interface GSHealthKitManager : NSObject

+ (GSHealthKitManager *)sharedManager;

- (void)requestAuthorization;

- (NSDate *)readBirthDate;
- (void)writeWeightSample:(double)weight;
- (NSString *)readGender;

@end
```

GSHealthKitManager.m

```
#import "GSHealthKitManager.h"
#import <HealthKit/HealthKit.h>

@interface GSHealthKitManager ()

@property (nonatomic, retain) HKHealthStore *healthStore;

@end

@implementation GSHealthKitManager

+ (GSHealthKitManager *)sharedManager {
    static dispatch_once_t pred = 0;
    static GSHealthKitManager *instance = nil;
    dispatch_once(&pred, ^{
        instance = [[GSHealthKitManager alloc] init];
        instance.healthStore = [[HKHealthStore alloc] init];
    });
    return instance;
}
```

```

- (void)requestAuthorization {

    if ([HKHealthStore isHealthDataAvailable] == NO) {
        // If our device doesn't support HealthKit -> return.
        return;
    }

    NSArray *readTypes = @[[HKObjectType
characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierDateOfBirth],[HKObjectType
characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierBiologicalSex]];

    [self.healthStore requestAuthorizationToShareTypes:nil readTypes:[NSSet
initWithArray:readTypes] completion:nil];
}

- (NSDate *)readBirthDate {
    NSError *error;
    NSDate *dateOfBirth = [self.healthStore dateOfBirthWithError:&error]; // Convenience
method of HKHealthStore to get date of birth directly.

    if (!dateOfBirth) {
        NSLog(@"Either an error occured fetching the user's age information or none has been
stored yet. In your app, try to handle this gracefully.");
    }

    return dateOfBirth;
}

- (NSString *)readGender
{
    NSError *error;
    HKBiologicalSexObject *gen=[self.healthStore biologicalSexWithError:&error];
    if (gen.biologicalSex==HKBiologicalSexMale)
    {
        return(@"Male");
    }
    else if (gen.biologicalSex==HKBiologicalSexFemale)
    {
        return(@"Female");
    }
    else if (gen.biologicalSex==HKBiologicalSexOther)
    {
        return(@"Other");
    }
    else{
        return(@"Not Set");
    }
}

@end

```

Aufruf von ViewController

```

- (IBAction)pressed:(id)sender {

    [[GSHealthKitManager sharedManager] requestAuthorization];
}

```

```
NSDate *birthDate = [[GSHealthKitManager sharedManager] readBirthDate];
    NSLog(@"birthdate %@", birthDate);
    NSLog(@"gender 2131321 %@", [[GSHealthKitManager sharedManager] readGender]);

}
```

Protokollausgabe

```
2016-10-13 14:41:39.568 random[778:26371] birthdate 1992-11-29 18:30:00 +0000
2016-10-13 14:41:39.570 random[778:26371] gender 2131321 Male
```

Healthkit online lesen: <https://riptutorial.com/de/ios/topic/7412/healthkit>

Kapitel 71: Hintergrund festlegen

Examples

Hintergrund festlegen

Ziel c:

```
view.backgroundColor = [UIColor redColor];
```

Schnell:

```
view.backgroundColor! = UIColor.redColor()
```

Swift 3

```
view.backgroundColor = UIColor.redColor
```

Füllen Sie das Hintergrundbild einer UIView

Ziel c

```
UIGraphicsBeginImageContext(self.view.frame.size);
[[UIImage imageNamed:@"image.png"] drawInRect:self.view.bounds];
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
self.view.backgroundColor = [UIColor colorWithPatternImage:image];
```

Setze Hintergrund mit Bild

```
self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"Background.png"]];
```

Erstellen einer Verlaufshintergrundansicht

Um einen Hintergrund mit Farbverlauf zu erstellen, können Sie die [CAGradientLayer](#)- Klasse verwenden:

Schnell 3,1:

```
func createGradient() {
    let caLayer = CAGradientLayer()
    caLayer.colors = [UIColor.white, UIColor.green, UIColor.blue]
    caLayer.locations = [0, 0.5, 1]
    caLayer.bounds = self.bounds
    self.layer.addSublayer(caLayer)
}
```

Dies kann wie folgt auf `viewDidLoad ()` aufgerufen werden:

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    createGradient()  
}
```

Die `CAGradientLayer`-Variablen für Positionen und Grenzen können mehrere Werte annehmen, um eine Verlaufsebene mit beliebig vielen Farben zu erstellen. Aus der Dokumentation:

Standardmäßig werden die Farben gleichmäßig über die Ebene verteilt. Sie können jedoch optional Positionen für die Steuerung der Farbpositionen im Verlauf angeben.

Hintergrund festlegen online lesen: <https://riptutorial.com/de/ios/topic/6854/hintergrund-festlegen>

Kapitel 72: Hintergrundmodi

Einführung

Reaktionsfähig zu sein, ist eine Notwendigkeit für jede App. Benutzer möchten Apps haben, deren Inhalte beim Öffnen bereitstehen. Entwickler sollten daher Hintergrundmodi verwenden, um ihre Apps benutzerfreundlicher zu gestalten.

Examples

Aktivieren der Hintergrundmodus-Funktion

1. Gehen Sie zu Xcode und öffnen Sie Ihr Projekt.
2. Navigieren Sie in Ihrem App-Ziel zur Registerkarte Funktionen.
3. Aktivieren Sie die Hintergrundmodi.



O. ↕

General

Capabilities

Resource Tags



Background Modes

- Modes:
- Audio, AirPlay, and Picture in Picture
 - Location updates
 - Voice over IP
 - Newsstand downloads
 - External accessory communication
 - Uses Bluetooth LE accessories
 - Acts as a Bluetooth LE accessory
 - Background fetch
 - Remote notifications

Steps: ✓ Add the Required Background Modes

Hintergrundabruf

Hintergrundabruf ist ein neuer Modus, mit dem Ihre App mit den neuesten Informationen immer auf dem neuesten Stand erscheint und die Auswirkungen auf den Akku so gering wie möglich gehalten werden. Mit dieser Funktion können Sie Feeds innerhalb festgelegter Zeitintervalle herunterladen.

Um anzufangen:

1- Überprüfen Sie den Hintergrundabruf im Funktionsbildschirm in Xcode.

2- `application(_:didFinishLaunchingWithOptions:)` in der `application(_:didFinishLaunchingWithOptions:)` -Methode in `AppDelegate` hinzu:

Schnell

```
UIApplication.shared.setMinimumBackgroundFetchInterval (UIApplicationBackgroundFetchIntervalMinimum)
```

Ziel c

```
[[UIApplication shared]
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum]
```

Anstelle von `UIApplicationBackgroundFetchIntervalMinimum` können Sie einen beliebigen `CGFloat` Wert verwenden, um die `CGFloat` .

3- Sie müssen die `application(_:performFetchWithCompletionHandler:)` implementieren `application(_:performFetchWithCompletionHandler:)` . Fügen Sie `AppDelegate` Ihrer `AppDelegate` :

Schnell

```
func application(_ application: UIApplication, performFetchWithCompletionHandler
completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
    // your code here
}
```

Testen des Hintergrundabrufs

1- Führen Sie die App auf einem echten Gerät aus und hängen Sie sie an den Xcode-Debugger an.

2- Wählen Sie im Debug-Menü die **Option Hintergrundabruf simulieren aus** :

Pause

⌘ Y

Continue To Current Line

⌘ C

Step Over

F6

Step Into

F7

Step Out

F8

Step Over Instruction

⌘ F6

Step Over Thread

⌘ ↑ F6

Step Into Instruction

⌘ F7

Step Into Thread

⌘ ↑ F7

Capture GPU Frame

GPU Overrides



Simulate Location



Simulate Background Fetch

Simulate UI Snapshot

iCloud



View Debugging



Deactivate Breakpoints

⌘ Y

Breakpoints



Debug Workflow



Kapitel 73: Hintergrundmodi und Ereignisse

Examples

Audio im Hintergrund abspielen

Fügen Sie einen Schlüssel mit dem Namen " **Erforderliche Hintergrundmodi**" in der Eigenschaftslistendatei (.plist) hinzu.

als folgendes Bild ..

Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
Localization native development r...	String	en
Bundle display name	String	
Executable file	String	\${EXECUTABLE_NAME}
▶ Icon files	Array	(14 items)
Bundle identifier	String	
InfoDictionary version	String	6.0
Bundle name	String	\${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.1
Bundle creator OS Type code	String	????
Bundle version	String	1.1
Application requires iPhone envir...	Boolean	YES
▼ Required background modes	Array	(1 item)
Item 0	String	App plays audio or streams audio/video using AirPlay
Icon already includes gloss effects	Boolean	YES
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(1 item)

Fügen Sie den folgenden Code hinzu

AppDelegate.h

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

AppDelegate.m

in der Anwendung didFinishLaunchingWithOptions

```
[[AVAudioSession sharedInstance] setDelegate:self];
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];
```

```
[[UIApplication sharedApplication] beginReceivingRemoteControlEvents];

UInt32 size = sizeof(CFStringRef);
CFStringRef route;
AudioSessionGetProperty(kAudioSessionProperty_AudioRoute, &size, &route);
NSLog(@"route = %@", route);
```

Wenn Sie Änderungen pro Ereignis vornehmen möchten, müssen Sie den folgenden Code in AppDelegate.m hinzufügen

```
- (void)remoteControlReceivedWithEvent:(UIEvent *)theEvent {

    if (theEvent.type == UIEventTypeRemoteControl)    {
        switch(theEvent.subtype)    {
            case UIEventSubtypeRemoteControlPlay:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlStop:
                break;
            case UIEventSubtypeRemoteControlTogglePlayPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            default:
                return;
        }
    }
}
```

Basierend auf Benachrichtigung müssen daran arbeiten ..

Hintergrundmodi und Ereignisse online lesen:

<https://riptutorial.com/de/ios/topic/3515/hintergrundmodi-und-ereignisse>

Kapitel 74: HINZUFÜGEN EINES SWIFT BRIDGING HEADER

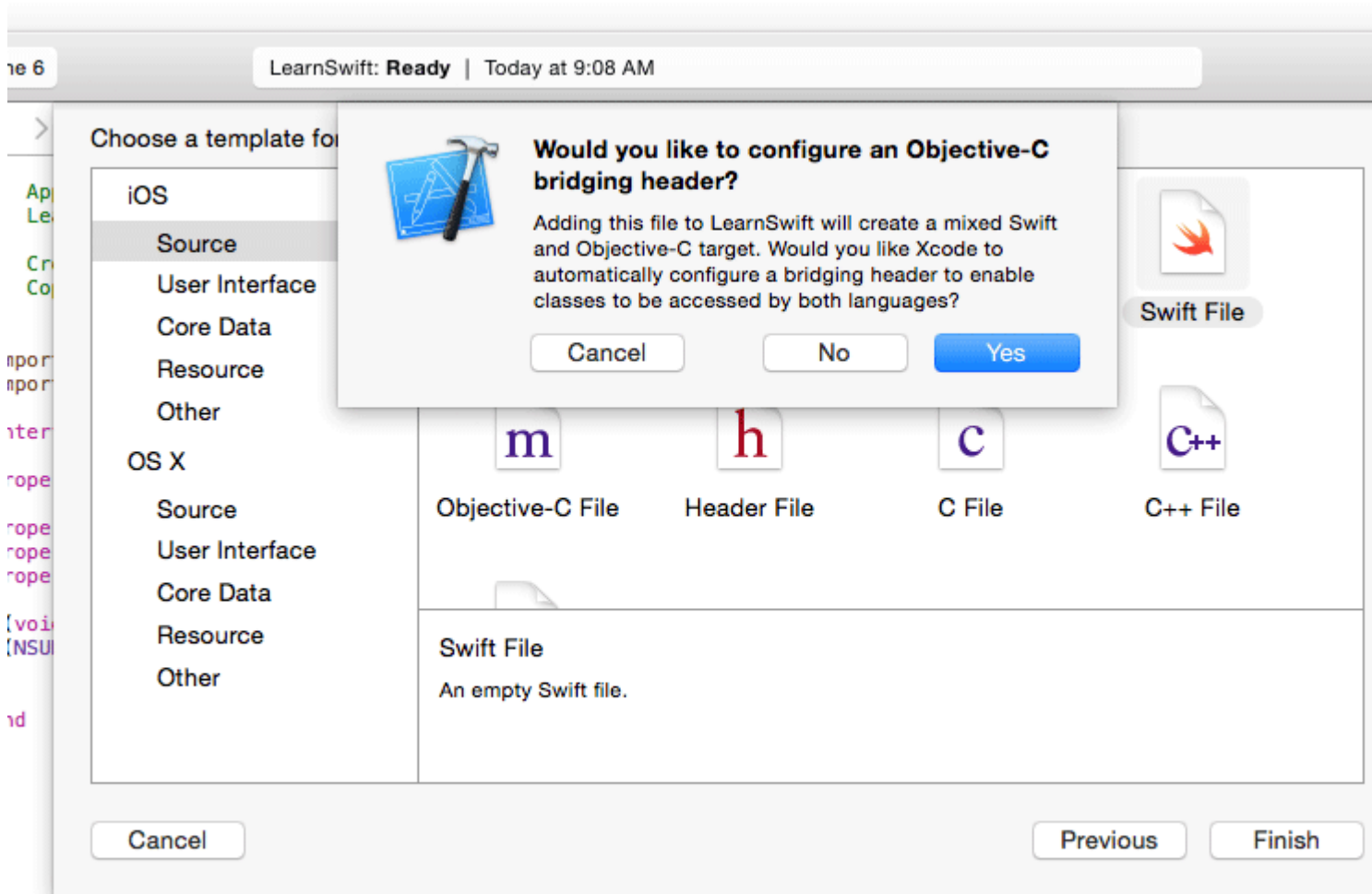
Examples

So erstellen Sie einen Swift Bridging Header manuell

- Fügen Sie eine neue Datei zu Xcode hinzu (Datei > Neu > Datei), wählen Sie dann „Quelle“ und klicken Sie auf „Header-Datei“.
- Benennen Sie Ihre Datei "YourProjectName-Bridging-Header.h". Beispiel: In meiner App Station heißt die Datei "Station-Bridging-Header".
- Erstellen Sie die Datei.
- Navigieren Sie zu den Einstellungen Ihres Projekts, und suchen Sie nach dem Abschnitt "Swift Compiler - Code Generation". Möglicherweise finden Sie es schneller, "Swift Compiler" in das Suchfeld einzugeben, um die Ergebnisse einzugrenzen. Hinweis: Wenn Sie keinen Abschnitt "Swift Compiler - Code Generation" haben, bedeutet dies, dass Sie wahrscheinlich noch keine Swift-Klassen zu Ihrem Projekt hinzugefügt haben. Fügen Sie eine Swift-Datei hinzu und versuchen Sie es erneut.
- Neben "Objective-C Bridging Header" müssen Sie den Namen / Pfad Ihrer Header-Datei angeben. Wenn sich Ihre Datei im Stammordner Ihres Projekts befindet, geben Sie einfach den Namen der Headerdatei an. Beispiele: "ProjectName / ProjectName-Bridging-Header.h" oder einfach "ProjectName-Bridging-Header.h".
- Öffnen Sie den neu erstellten Bridging-Header und importieren Sie Ihre Objective-C-Klassen mithilfe von #import-Anweisungen. Auf jede Klasse, die in dieser Datei aufgelistet ist, kann von Ihren schnellen Klassen aus zugegriffen werden.

Xcode automatisch erstellen

Fügen Sie Ihrem Xcode-Projekt eine neue Swift-Datei hinzu. Nennen Sie es wie gewünscht und Sie sollten eine Warnmeldung erhalten, in der Sie gefragt werden, ob Sie einen Bridging-Header erstellen möchten. Hinweis: Wenn Sie keine Aufforderung zum Hinzufügen eines Bridging-Headers erhalten, haben Sie diese Nachricht wahrscheinlich schon einmal abgelehnt und müssen den Header manuell hinzufügen (siehe unten).



HINZUFÜGEN EINES SWIFT BRIDGING HEADER online lesen:

<https://riptutorial.com/de/ios/topic/10851/hinzufugen-eines-swift-bridging-header>

Kapitel 75: iBeacon

Parameter

Parameter	Einzelheiten
Manager	CLLocationManager-Referenz
Region	CLRegion kann kreisförmige Region sein (Geofence- oder Beacon-Region)
Baken	Das Array von CLBeacon enthält alle Ranger Beacons

Bemerkungen

Baken sind IOT-Objekte. Wir konzentrieren uns auf diejenigen, die dem iBeacon-Protokoll und einem Apple-Standard entsprechen. Jede Bake ist ein Einweggerät, das 3 Dinge überträgt

1. UUID
2. Haupt
3. Geringer

Wir können iBeacons scannen, indem Sie unser CLLocation Manager-Objekt so einrichten, dass es nach Beacons für bestimmte UUID sucht. Alle Beacons mit der angegebenen UUID werden gescannt.

Der CLLocation Manager gibt auch einen Anruf beim Ein- und Austritt der Beacon-Region an.

Examples

iBeacon-Grundbedienung

1. Überwachungsbaken einrichten

```
func initiateRegion(ref:BeaconHandler) {
    let uuid: NSUUID = NSUUID(UUIDString: "<UUID>")
    let beacon = CLBeaconRegion(proximityUUID: uuid, identifier: "")
    locationManager?.requestAlwaysAuthorization() //CLLocationManager obj.
    beacon?.notifyOnEntry = true
    beacon?.notifyOnExit = true
    beacon?.notifyEntryStateOnDisplay = true
    locationManager?.startMonitoringForRegion(beacon!)
    locationManager?.delegate = self;
    // Check if beacon monitoring is available for this device
    if (!CLLocationManager.isMonitoringAvailableForClass(CLBeaconRegion)) {
        print("error")
    }
    locationManager!.startRangingBeaconsInRegion(self.beacon!)
}
```


2. Standortmanager betreten und verlassen Region

```
func locationManager(manager: CLLocationManager, didEnterRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.startRangingBeaconsInRegion(self.beacon!)
    }
}

func locationManager(manager: CLLocationManager, didExitRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.stopRangingBeaconsInRegion(self.beacon!)
    }
}
```

3. Bereichsmanager für Positionsmanager

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    print(beacons.first.major)
}
```

Scannen bestimmter Beacons

```
beacon = CLBeaconRegion(proximityUUID: <#NSUUID#>, major: <#CLBeaconMajorValue#>, identifier:
<#String#>) // listening to all beacons with given UUID and major value
beacon = CLBeaconRegion(proximityUUID: <#NSUUID#>, major: <##CLBeaconMajorValue#>, minor:
<##CLBeaconMinorValue#>, identifier: <##String#>) // listening to all beacons with given UUID
and major and minor value
```

IBeacons in Reichweite

Zunächst müssen Sie die Autorisierung von Standortdiensten anfordern

```
let locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()
// OR locationManager.requestAlwaysAuthorization()
```

Dann können Sie alle Informationen von `didRangeBeacons` in `didRangeBeacons`

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    for beacon in beacons {
        print(beacon.major)
        print(beacon.minor)
    }
}
```

iBeacon online lesen: <https://riptutorial.com/de/ios/topic/1958/ibeacon>

Kapitel 76: IBOutlets

Bemerkungen

IBOutlet ist weder ein reserviertes Wort noch eine Variable oder Klasse. Es ist syntaktischer Zucker für den Interface Builder. Nachdem der Objective-C-Quellcode vorverarbeitet wurde, wird er zu nichts aufgelöst.

In Swift wird es als Null aufgelöst.

Es wird in `<UIKit/UINibDeclarations.h>` als `<UIKit/UINibDeclarations.h>`

```
#ifndef IBOutlet
#define IBOutlet
#endif
```

Examples

Verwenden eines IBOutlets in einem UI-Element

Im Allgemeinen werden IBOutlets verwendet, um ein Benutzeroberflächenobjekt mit einem anderen Objekt, in diesem Fall einem UIViewController, zu verbinden. Die Verbindung dient dazu, dass das Objekt programmgesteuert auf meinen Code oder meine Ereignisse wirkt. Dies kann einfach durch Verwenden des Assistenten aus einem Storyboard und durch Klicken mit der rechten Maustaste vom Element in den .h-Eigenschaftsbereich des View-Controllers erfolgen. Dies kann jedoch auch programmgesteuert und manuell durch Verbinden des IBOutlet-Codes mit der Registerkarte "Verbindungen" des Objekts erfolgen die Werkzeugleiste rechts. Hier ist ein objektives Beispiel für einen UIViewController mit Etikettenauslass:

```
//ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

//This is the declaration of the outlet
@property (nonatomic, weak) IBOutlet UILabel *myLabel;

@end

//ViewController.m
#import "ViewController.h"

@implementation ViewController

@synthesize myLabel;

-(void) viewDidLoad {

    [super viewDidLoad];
    //Editing the properties of the outlet
```

```
myLabel.text = @"TextHere";  
  
}  
  
@end
```

Und schnell:

```
import UIKit  
class ViewController: UIViewController {  
    //This is the declaration of the outlet  
    @IBOutlet weak var myLabel: UILabel!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        //Editing the properties of the outlet  
        myLabel.text = "TextHere"  
    }  
}
```

Die Verbindung zwischen dem Storyboard-Objekt und dem programmierten Objekt kann als verbunden überprüft werden, wenn der Punkt links von der Deklaration des Auslasses in der .h gefüllt ist. Ein leerer Kreis implizierte eine unvollständige Verbindung.

IBOutlets online lesen: <https://riptutorial.com/de/ios/topic/4713/iboutlets>

Kapitel 77: In-App-Kauf

Examples

Einzelne IAP in Swift 2

Nach dem Erstellen eines IAP in iTunesConnect:

In dem View-Controller, den Sie kaufen möchten

```
import StoreKit
```

und fügen Sie die entsprechenden Delegierten hinzu

```
class ViewController: UIViewController, SKProductsRequestDelegate, SKPaymentTransactionObserver {
```

Deklarieren Sie eine Variable mit der Produkt-ID von iTunesConnect

```
var product_id: NSString?

override func viewDidLoad() {

    product_id = "YOUR_PRODUCT_ID"
    super.viewDidLoad()
    SKPaymentQueue.defaultQueue().addTransactionObserver(self)

    //Check if product is purchased
    if (NSUserDefaults.standardUserDefaults().boolForKey("purchased")){

        // Hide ads
        adView.hidden = true

    } else {
        print("Should show ads...")
    }

}
```

Verbinden Sie eine Taste mit einer Funktion, um den IAP zu erwerben

```
@IBAction func unlockAction(sender: AnyObject) {

    print("About to fetch the product...")

    // Can make payments
    if (SKPaymentQueue.canMakePayments())
    {
        let productID:NSSet = NSSet(object: self.product_id!);
        let productsRequest:SKProductsRequest = SKProductsRequest(productIdentifiers:
```

```

productID as! Set<NSString>);
    productsRequest.delegate = self;
    productsRequest.start();
    println("Fetching Products");
} else {
    print("Can't make purchases");
}
}

```

Und hier sind einige Hilfsmethoden

```

func buyProduct(product: SKProduct) {
    println("Sending the Payment Request to Apple");
    let payment = SKPayment(product: product)
    SKPaymentQueue.defaultQueue().addPayment(payment);
}

```

die Delegatmethoden, die deklariert werden müssen

```

func productsRequest (request: SKProductsRequest, didReceiveResponse response:
SKProductsResponse) {

    let count : Int = response.products.count
    if (count>0) {
        var validProduct: SKProduct = response.products[0] as SKProduct
        if (validProduct.productIdentifier == self.product_id) {
            print(validProduct.localizedTitle)
            print(validProduct.localizedDescription)
            print(validProduct.price)
            buyProduct(validProduct);
        } else {
            print(validProduct.productIdentifier)
        }
    } else {
        print("nothing")
    }
}

func request(request: SKRequest!, didFailWithError error: NSError!) {
    print("Error Fetching product information");
}

func paymentQueue(_ queue: SKPaymentQueue,
updatedTransactions transactions: [SKPaymentTransaction])
{
    print("Received Payment Transaction Response from Apple");

    for transaction:AnyObject in transactions {
        if let trans:SKPaymentTransaction = transaction as? SKPaymentTransaction{
            switch trans.transactionState {
                case .Purchased:
                    print("Product Purchased");
                    SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
SKPaymentTransaction)
                    // Handle the purchase

```

```
        NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
        adView.hidden = true
        break;
    case .Failed:
        print("Purchased Failed");
        SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
SKPaymentTransaction)
        break;

    case .Restored:
        print("Already Purchased");
        SKPaymentQueue.defaultQueue().restoreCompletedTransactions()

        // Handle the purchase
        NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
        adView.hidden = true
        break;
    default:
        break;
    }
}
}
```

Und dann den Code zum Wiederherstellen eines Nichtverbrauchs beim Kauf einer App

```
if (SKPaymentQueue.canMakePayments()) {
    SKPaymentQueue.defaultQueue().restoreCompletedTransactions()
}
```

In iTunesConnect einrichten

Wählen Sie in [iTunesConnect](#) die App aus, der Sie ein IAP hinzufügen möchten.

Klicken Sie auf Features und Sie werden folgendes sehen:

In-App Purchases (0)

Click

Klicken Sie auf das Plus. Sie müssen dann auswählen, welchen IAP-Typ Sie erstellen möchten.

Dann müssen Sie alle Informationen für Ihr IAP ausfüllen.

In-App Purchase Summary

Enter a reference name and a product ID for this In-App Purchase.

Reference Name

Product ID

Pricing and Availability

Enter the pricing and availability details for this In-App Purchase.

Cleared for Sale

Price

Kapitel 78: Initialisierungs-Idiome

Examples

Auf Tupel setzen, um Code-Wiederholung zu vermeiden

Vermeiden Sie Codewiederholungen in Konstruktoren, indem Sie ein Tupel von Variablen mit einem Einzeiler setzen:

```
class Contact: UIView
{
    private var message: UILabel
    private var phone: UITextView

    required init?(coder aDecoder: NSCoder) {
        (message, phone) = self.dynamicType.setUp()
        super.init(coder: aDecoder)
    }

    override func awakeFromNib() {
        (message, phone) = self.dynamicType.setUp()
        super.awakeFromNib()
    }

    override init(frame: CGRect) {
        (message, phone) = self.dynamicType.setUp()
        super.init(frame: frame)
    }

    private static func setUp(){
        let message = UILabel() // ...
        let phone = UITextView() // ...
        return (message, phone)
    }
}
```

Initialisieren Sie mit Positionskonstanten

```
let mySwitch: UISwitch = {
    view.addSubview($0)
    $0.addTarget(self, action: "action", forControlEvents: .TouchUpInside)
    return $0
}(UISwitch())
```

Attribute in didSet initialisieren

```
@IBOutlet weak var title: UILabel! {
    didSet {
        label.textColor = UIColor.redColor()
        label.font = UIFont.systemFontOfSize(20)
        label.backgroundColor = UIColor.blueColor()
    }
}
```

```
}
```

Es ist auch möglich, einen Wert festzulegen und ihn zu initialisieren:

```
private var loginButton = UIButton() {
    didSet(oldValue) {
        loginButton.addTarget(self, action: #selector(LoginController.didClickLogin),
        forControlEvents: .TouchUpInside)
    }
}
```

Gruppieren Sie Auslässe in einem benutzerdefinierten NSObject

Bewegen Sie jeden Ausgang zu einem NSObject. Ziehen Sie dann ein Objekt aus der Bibliothek in die Controller-Szene des Storyboards und haken Sie die Elemente dort ein.

```
class ContactFormStyle: NSObject
{
    @IBOutlet private weak var message: UILabel! {
        didSet {
            message.font = UIFont.systemFontOfSize(12)
            message.textColor = UIColor.blackColor()
        }
    }
}

class ContactFormVC: UIViewController
{
    @IBOutlet private var style: ContactFormStyle!
}
```

Dann mit initialisieren

Dies entspricht in der Syntax dem Beispiel, das mit Positionskonstanten initialisiert wird, erfordert jedoch die `Then` Erweiterung von <https://github.com/devxoul/Then> (im Anhang unten).

```
let label = UILabel().then {
    $0.textAlignment = .Center
    $0.textColor = UIColor.blackColor()
    $0.text = "Hello, World!"
}
```

Die `Then` Erweiterung:

```
import Foundation

public protocol Then {}

extension Then
{
    public func then(@noescape block: inout Self -> Void) -> Self {
        var copy = self
        block(&copy)
    }
}
```

```
        return copy
    }
}

extension NSObject: Then {}
```

Werksmethode mit Block

```
internal fun<Type> Init<Type>(value : Type, block: @noescape (object: Type) -> Void) -> Type
{
    block(object: value)
    return value
}
```

Verwendungszweck:

```
Init(UILabel(frame: CGRect.zero)) {
    $0.backgroundColor = UIColor.blackColor()
}
```

Initialisierungs-Idiome online lesen: <https://riptutorial.com/de/ios/topic/3513/initialisierungs-idiome>

Kapitel 79: Integration von SqlCipher

Einführung

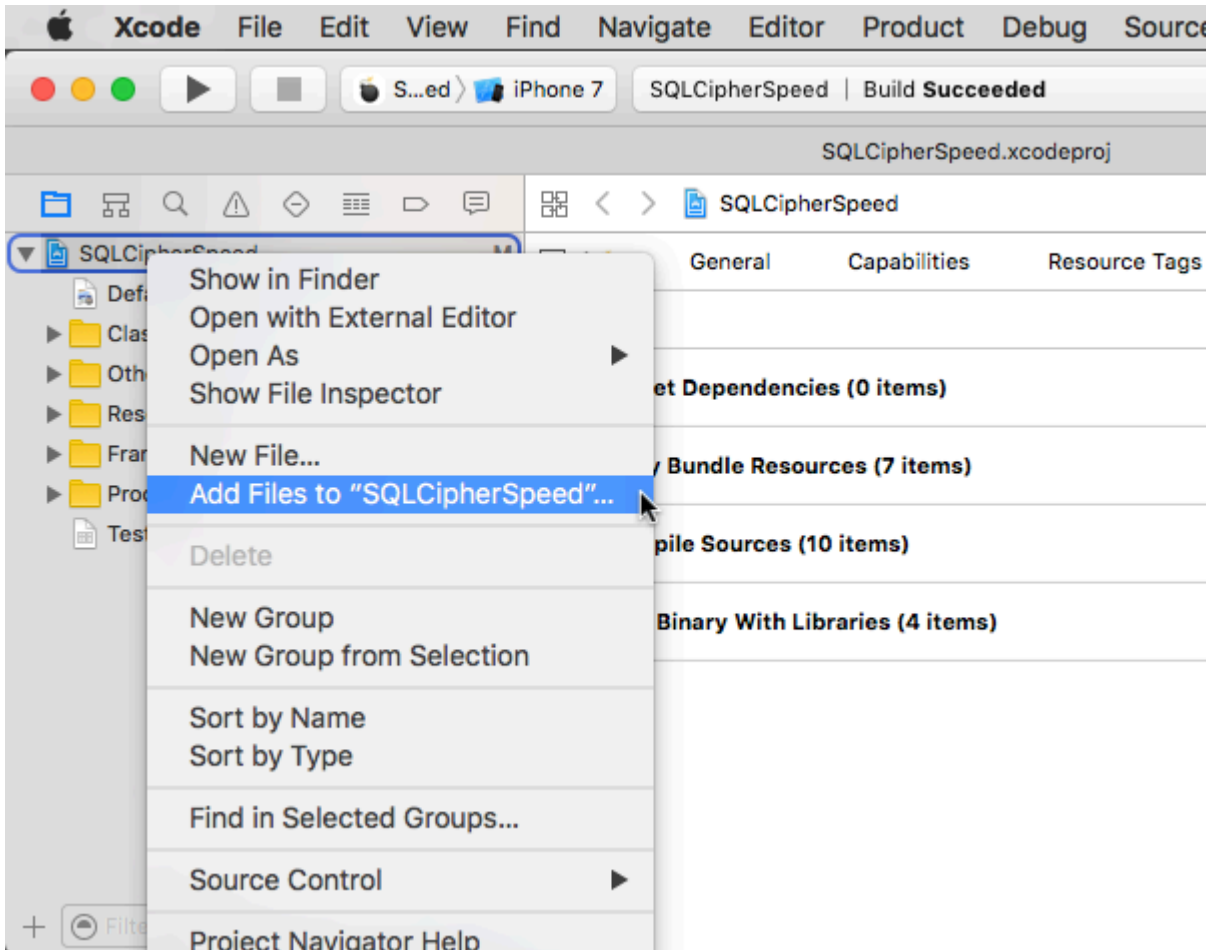
SQLite ist bereits eine beliebte API für die dauerhafte Speicherung von Daten in iOS-Apps. Die Entwicklung liegt also auf der Hand. Als Programmierer arbeiten Sie mit einer stabilen, gut dokumentierten API, für die in Objective-C viele gute Wrapper verfügbar sind, beispielsweise FMDB und Encrypted Core Data. Alle Sicherheitsbedenken sind vom Anwendungscode sauber entkoppelt und werden vom zugrunde liegenden Framework verwaltet.

Bemerkungen

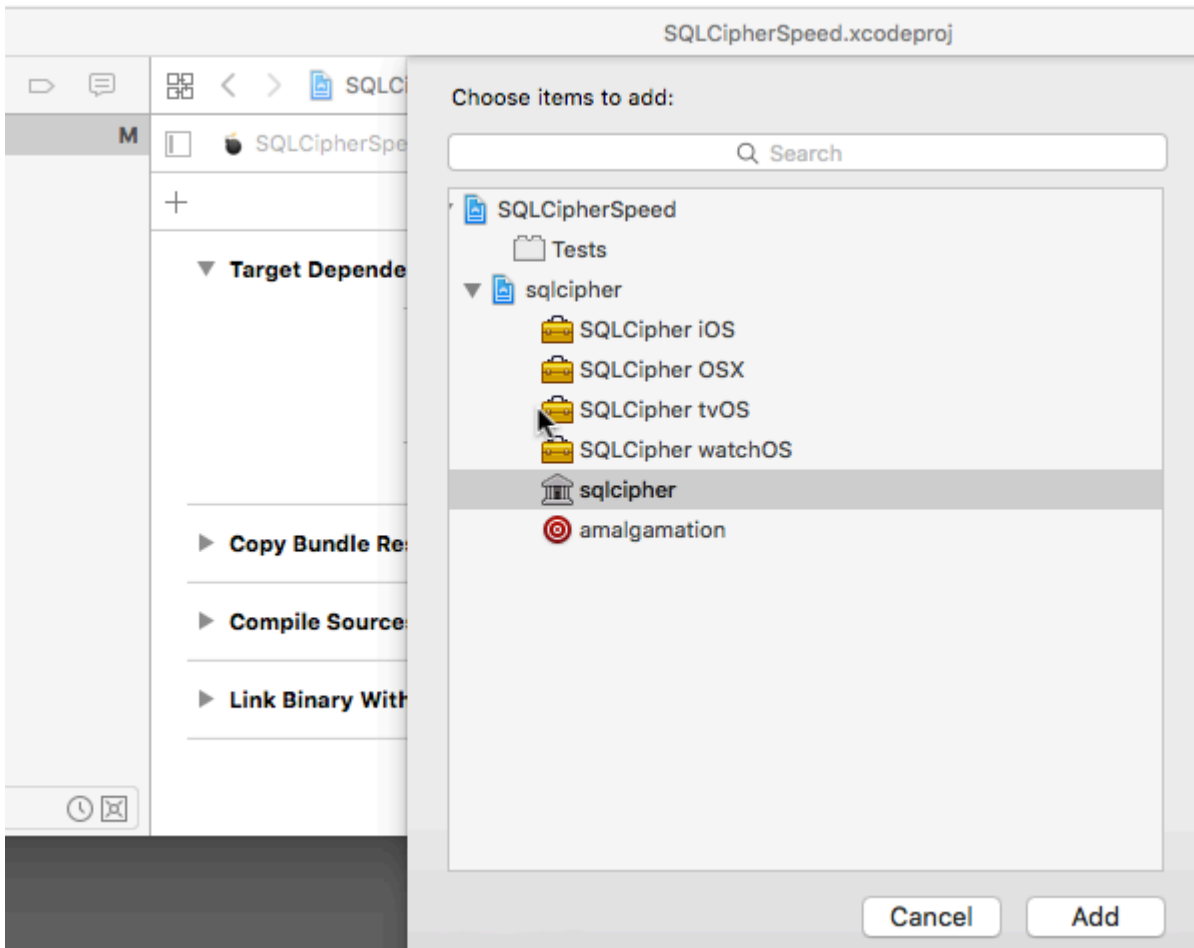
1. Öffnen Sie das Terminal, wechseln Sie in das Stammverzeichnis Ihres Projekts und checken Sie den SQLCipher-Projektcode mithilfe von Git aus:

```
$ git clone https://github.com/sqlcipher/sqlcipher.git
```

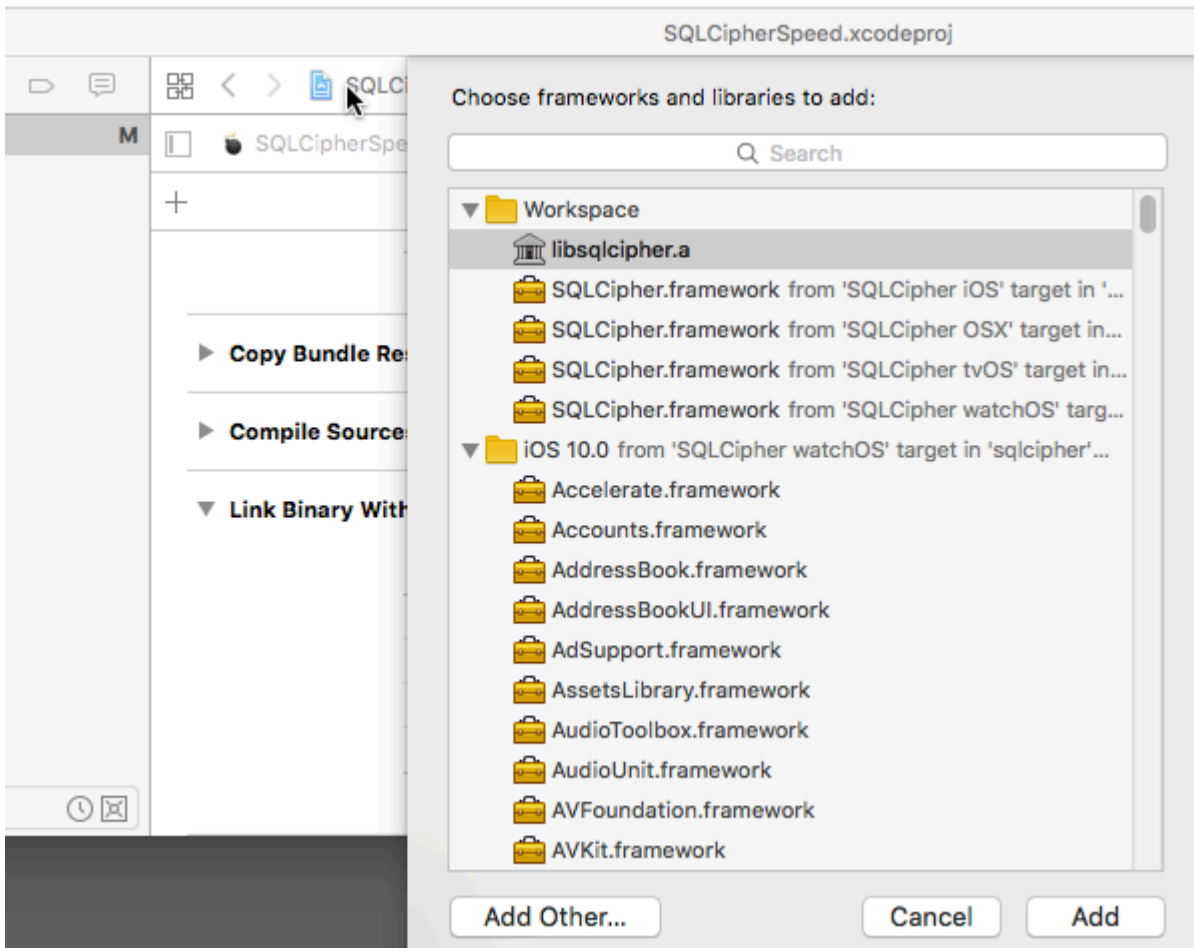
2. Klicken Sie mit der rechten Maustaste auf das Projekt und wählen Sie "Dateien zu" Meine App hinzufügen "" (die Bezeichnung hängt vom Namen Ihrer App ab). Da wir SQLCipher direkt in den gleichen Ordner wie Ihre iOS-App geklont haben, sollte ein sqlcipher-Ordner in Ihrem Stammprojektordner angezeigt werden. Öffnen Sie diesen Ordner und wählen Sie **sqlcipher.xcodeproj** aus



3. Wählen Sie den Bereich Build-Einstellungen aus. Geben Sie im Suchfeld "Header Search Paths" ein. Doppelklicken Sie auf das Feld unter der **Zielspalte** und fügen Sie den folgenden Pfad hinzu: **\$ (PROJECT_DIR) / sqlcipher / src**
4. Beginnen Sie, "Other Linker Flags" in das Suchfeld **einzugeben**, bis die Einstellung angezeigt wird, **doppelklicken** Sie, um sie zu bearbeiten, und fügen Sie den folgenden Wert hinzu:
5. Beginnen Sie, "Other C Flags" in das Suchfeld einzugeben, bis die Einstellung angezeigt wird, **doppelklicken** Sie, um sie zu bearbeiten, und fügen Sie im Popup den folgenden Wert hinzu:
6. Erweitern Sie Zielabhängigkeiten und klicken Sie auf die Schaltfläche + am Ende der Liste. **Wählen** Sie im sich **öffnenden** Browser das statische Bibliotheksziel **sqlcipher aus** :



7. Erweitern Sie Link Binary With Libraries, klicken Sie auf die Schaltfläche + am Ende der Liste und wählen Sie die Bibliothek **libsqlcipher.a** aus .



8. Fügen Sie schließlich auch unter Verknüpfung mit Bibliotheken **Security.framework** hinzu .

Examples

Integration von Code:

Integration zum Öffnen der Datenbank mit Passwort.

```

- (void) checkAndOpenDB{
    sqlite3 *db;
    NSString *strPassword = @"password";

    if (sqlite3_open_v2([[databaseURL path] UTF8String], &db, SQLITE_OPEN_READWRITE |
    SQLITE_OPEN_CREATE, NULL) == SQLITE_OK) {
        const char* key = [strPassword UTF8String];
        sqlite3_key(db, key, (int)strlen(key));
        if (sqlite3_exec(db1, (const char*) "SELECT count(*) FROM sqlcipher_master;", NULL,
        NULL, NULL) == SQLITE_OK) {
            NSLog(@"Password is correct, or a new database has been initialized");
        } else {
            NSLog(@"Incorrect password!");
        }
        sqlite3_close(db);
    }
}

- (NSURL *)databaseURL
{

```

```
NSArray *URLs = [[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
inDomains:NSUserDomainMask];
NSURL *directoryURL = [URLs firstObject];
NSURL *databaseURL = [directoryURL URLByAppendingPathComponent:@"database.sqlite"];
return databaseURL;
}
```

Integration von SqlCipher online lesen: <https://riptutorial.com/de/ios/topic/9969/integration-von-sqlcipher>

Kapitel 80: iOS - Implementierung von XMPP mit dem Robbie Hanson-Framework

Examples

Robbie Hanson-Beispiel für iOS XMPP mit Openfire

SRXMPPDemo

Laden Sie das Beispiel und alle Klassen hier herunter - <https://github.com/SahebRoy92/SRXMPPDemo>

Eine Demo zu XMPP in Objective C mit verschiedenen einfachen und komplexen Funktionen. Alle Funktionen von XMPP werden von Xmpp-Funktionen **"in Band" ausgeführt** . Dieses Projekt enthält nur wenige Funktionen:

SRXMPP - Eine Wrapper-Singleton-Klasse, die fast alle Funktionen aufweist, die für One-to-One-Chat-Anwendungen erforderlich sind.

- Einzelgespräch
- Kerndatenimplementierung des Chats (Kurzmitteilung), dadurch Speichern vorheriger Meldungen, Offline-Meldungen.
- Implementierung von vCard (Profilinformationen von Benutzern, eigenen und anderen) aus XML und Core Data, bereitgestellt durch Robbie Hansons eigenes Framework.
- Verfügbarkeit des Freundesstatus (online / offline / Eingabe)

Schritte zum folgen

Wenn Sie dieses Projekt als Referenz verwenden möchten, können Sie Folgendes tun:

1. Installierte Openfire auf einem Live-Server - Mieten Sie einen Server und installieren Sie Openfire.

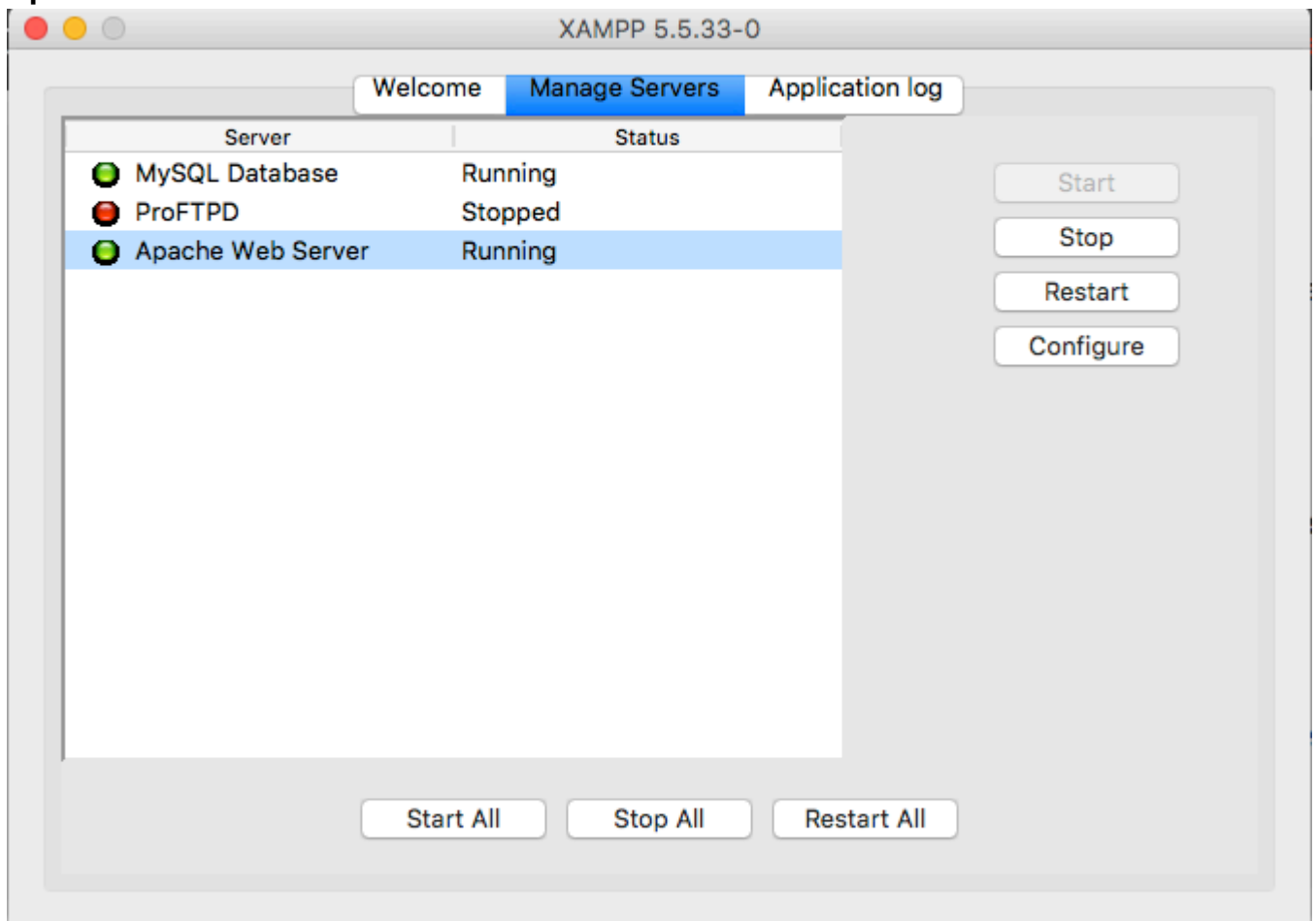
2. Sie möchten es ohne Probleme auf Ihrem eigenen Computer ausprobieren - Sie müssen 3 Dinge herunterladen, installieren und einrichten, um zu starten

ein. Java -

- Laden Sie Java für Mac herunter und installieren Sie es.

b. XAMPP -

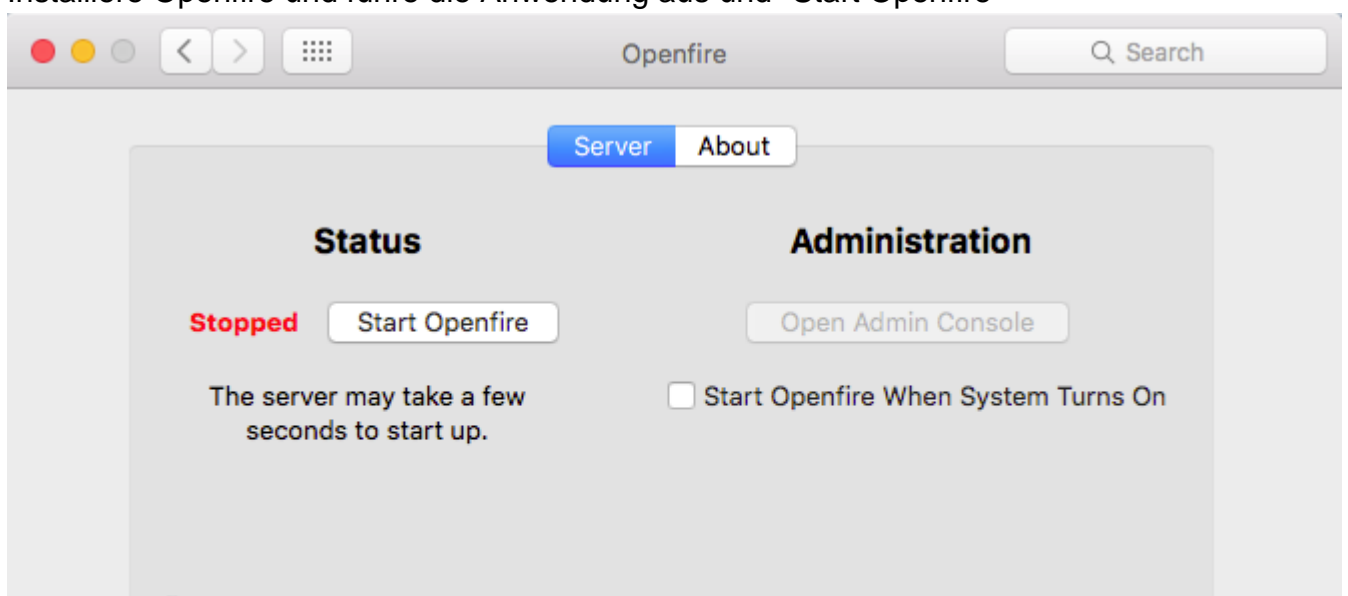
- Die Installation von XAMPP ist relativ einfach.
- Nach der Installation starten Sie einfach XAMPP und starten Sie **Database (SQL)** und **Apache Server** .



- Öffnen Sie dann den Browser und fügen Sie diese URL ein [\[http://localhost/phpmyadmin/\]](http://localhost/phpmyadmin/).
- . Erstellen Sie eine neue Datenbank im linken Bereich.
- **Benennen Sie die Datenbank nicht, sondern erinnern Sie sich an diesen Namen. Angenommen, wir nennen sie ChatDB**

c. Openfire -

- Installiere Openfire und führe die Anwendung aus und "Start Openfire"



- Browser öffnen und diese URL einfügen - [<http://localhost:9090/setup/index.jsp>] (<http://localhost:9090/setup/index.jsp>)
- Führen Sie das normale Setup durch
 - Wählen Sie Sprache>
 - Servereinstellungen, so wie sie sind, einfach fortfahren>
 - Datenbankeinstellungen, so wie es ist "Standard-Datenbankverbindung wie ausgewählt>
 - Datenbankeinstellungen - Standardverbindung ". **Denken** Sie jetzt daran, dass der Name der DB, die Sie festgelegt haben, **ChatDB ist** .
 - Wählen Sie Database Driver Presets als * " *MySQL* " aus . Lassen Sie die JDBC-Treiberklasse unverändert. Nun sehen Sie in der Datenbank-URL Klammern, die den Hostnamen und den Datenbanknamen erwähnen. Ändern Sie einfach Hostname in "**localhost**" und den Datenbanknamen in "**ChatDB**" oder einen anderen Namen der DB, den Sie zuvor festgelegt haben, während Sie XAMPP einrichten. Lassen Sie den Benutzernamen und das Passwort leer. Geben Sie Details wie das Bild hier ein

Database Settings - Standard Connection

Specify a JDBC driver and connection properties to connect to your database. If you need more information about this pro

Note: Database scripts for most popular databases are included in the server distribution at [Openfire_HOME]/resc

Database Driver Presets:

JDBC Driver Class:

Database URL:

Username:

Password:

Minimum Connections:

Maximum Connections:

Connection Timeout: Days

- Schließen Sie anschließend das Setup ab, indem Sie einen Benutzernamen und ein Kennwort eingeben und es erneut bestätigen. Das ist es euer getanes Openfire einrichten.

Jetzt kommt der Teil, wenn Sie ein kleines Detail im Code ändern müssen.

Important Wir müssen zur Klasse **SRXMPP.m** gehen , den NSString extern **SRXMPP_Hostname** (im oberen Bereich) **suchen** und den Wert davon in den Wert überschreiben

- IP des Servers, auf dem OpenFire installiert ist, **ODER**
- Wenn Sie es lokal installiert haben, überschreiben Sie den Wert in - "**localhost**" .

Das ist es, Sie sind bereit, dieses Beispielprojekt zu verwenden und mit dem Programmieren zu beginnen und es in ein besseres Projekt zu verwandeln.

Dieses Starterpaket hilft Ihnen, die XMPP-Struktur besser zu verstehen und sich mit den XMPP-Protokollen vertraut zu machen.

Weitere XMPP-Protokolle finden Sie hier auf dieser Site - [
<https://xmpp.org/rfcs/rfc3920.html>(((<https://xmpp.org/rfcs/rfc3920.html>))

Entwicklung ist noch übrig und Teile, wo ich sie später noch einbauen möchte

1. Gruppenchat
2. Unterstützung beim Senden von Bildern

Kurz gesagt, dieses Beispielprojekt hat zusammen mit dem Singleton fast alle Funktionen, die für eine One-to-One-Chat-Anwendung erforderlich sind.

iOS - Implementierung von XMPP mit dem Robbie Hanson-Framework online lesen:
<https://riptutorial.com/de/ios/topic/1475/ios---implementierung-von-xmpp-mit-dem-robbie-hanson-framework>

Kapitel 81: iOS 10-Spracherkennungs-API

Examples

Sprache zu Text: Erkennen Sie Sprache aus einem Bündel mit Audioaufzeichnung

```
//import Speech
//import AVFoundation

// create a text field to show speech output
@IBOutlet weak var transcriptionTextField: UITextView!
// we need this audio player to play audio
var audioPlayer: AVAudioPlayer!

override func viewDidLoad()
{
    super.viewDidLoad()
}

// this function is required to stop audio on audio completion otherwise it will play same
audio again and again
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool)
{
    player.stop()
}

// this function is required to get a speech recognizer and after that make and request to
speech recognizer
func requestSpeechAuth()
{
    SFSpeechRecognizer.requestAuthorization { authStatus in
        if authStatus == SFSpeechRecognizerAuthorizationStatus.authorized {
            if let path = Bundle.main.url(forResource: "mpthreetest", withExtension: "m4a") {
                do {
                    let sound = try AVAudioPlayer(contentsOf: path)
                    self.audioPlayer = sound
                    self.audioPlayer.delegate = self
                    sound.play()
                } catch {
                    print("error")
                }
            }

            let recognizer = SFSpeechRecognizer()
            let request = SFSpeechURLRecognitionRequest(url:path)
            recognizer?.recognitionTask(with: request) { (result, error) in
                if let error = error {
                    print("there is a error\(error)")
                } else {
                    // here you are printing out the audio output basically showing it on uitext field
                    self.transcriptionTextField.text =
                    result?.bestTranscription.formattedString
                }
            }
        }
    }
}
```

```
    }  
}  
  
// here you are calling requestSpeechAuth function on UIButton press  
@IBAction func playButtonPress(_ sender: AnyObject)  
{  
    requestSpeechAuth()  
}
```

iOS 10-Spracherkennungs-API online lesen: <https://riptutorial.com/de/ios/topic/5986/ios-10-spracherkennungs-api>

Kapitel 82: iOS Google Places-API

Examples

Orte in der Nähe vom aktuellen Standort abrufen

Voraussetzungen

1. Installieren Sie Pods in Ihrem Projekt
2. Installieren Sie das GooglePlaces SDK
3. Standortdienste aktivieren

Zuerst müssen wir den Benutzer anhand der aktuellen Längen- und Breitengrade ermitteln.

1. Importieren Sie GooglePlaces und GooglePlacePicker

```
import GooglePlaces
import GooglePlacePicker
```

2. Fügen Sie das CLLocationManagerDelegate Protokoll hinzu

```
class ViewController: UIViewController, CLLocationManagerDelegate {
}
```

3. Erstelle deinen CLLocationManager ()

```
var currentLocation = CLLocationManager()
```

4. Autorisierung anfordern

```
currentLocation = CLLocationManager()
currentLocation.requestAlwaysAuthorization()
```

5. Erstellen Sie eine Schaltfläche, um die GooglePlacePicker-Methode aufzurufen

@IBAction func placePickerAction (Sender: AnyObject) {

```
if CLLocationManager.authorizationStatuses() == .AuthorizedAlways {

    let center =
CLLocationCoordinate2DMake((currentLocation.location?.coordinate.latitude)!,
(currentLocation.location?.coordinate.longitude)!)
    let northEast = CLLocationCoordinate2DMake(center.latitude + 0.001, center.longitude +
0.001)
    let southWest = CLLocationCoordinate2DMake(center.latitude - 0.001, center.longitude -
0.001)
    let viewport = GMSCoordinateBounds(coordinate: northEast, coordinate: southWest)
    let config = GMSPlacePickerConfig(viewport: viewport)
```

```
placePicker = GMSPlacePicker(config: config)

placePicker?.pickPlaceWithCallback({ (place: GMSPlace?, error: NSError?) -> Void in
    if let error = error {
        print("Pick Place error: \(error.localizedDescription)")
        return
    }

    if let place = place {
        print("Place name: \(place.name)")
        print("Address: \(place.formattedAddress)")
    } else {
        print("Place name: nil")
        print("Address: nil")
    }
})
}
```

iOS Google Places-API online lesen: <https://riptutorial.com/de/ios/topic/6908/ios-google-places-api>

Kapitel 83: iOS TTS

Einführung

Finden Sie heraus, wie Sie synthetisierte Sprache aus Text auf einem iOS-Gerät erzeugen können

Examples

Text zu Sprache

Ziel c

```
AVSpeechSynthesizer *synthesizer = [[AVSpeechSynthesizer alloc] init];
AVSpeechUtterance *utterance = [AVSpeechUtterance speechUtteranceWithString:@"Some text"];
[utterance setRate:0.2f];
[synthesizer speakUtterance:utterance];
```

Schnell

```
let synthesizer = AVSpeechSynthesizer()
let utterance = AVSpeechUtterance(string: "Some text")
utterance.rate = 0.2
```

Sie können die Stimme auch so ändern:

```
utterance.voice = AVSpeechSynthesisVoice(language: "fr-FR")
```

Und dann sprechen

- In Swift 2: `synthesizer.speakUtterance(utterance)`
- In Swift 3: `synthesizer.speak(utterance)`

Vergessen Sie nicht, **AVFoundation** zu importieren

Hilfreiche Methoden

Mit diesen beiden Methoden können Sie die gesamte Sprache anhalten oder anhalten:

```
- (BOOL) pauseSpeakingAtBoundary: (AVSpeechBoundary) boundary;
- (BOOL) stopSpeakingAtBoundary: (AVSpeechBoundary) boundary;
```

Die `AVSpeechBoundary` gibt an, ob die Sprache sofort `AVSpeechBoundaryImmediate` oder gestoppt

werden soll (`AVSpeechBoundaryImmediate`) oder nach dem aktuell gesprochenen Wort (`AVSpeechBoundaryWord`) anhalten oder stoppen `AVSpeechBoundaryWord` .

iOS TTS online lesen: <https://riptutorial.com/de/ios/topic/8909/ios-tts>

Kapitel 84: iOS-Version überprüfen

Examples

iOS 8 und höher

Swift 3:

```
let minimumVersion = OperatingSystemVersion(majorVersion: 8, minorVersion: 1, patchVersion: 2)
if ProcessInfo().isOperatingSystemAtLeast(minimumVersion) {
    //current version is >= (8.1.2)
} else {
    //current version is < (8.1.2)
}
```

Versionen vergleichen

```
let minimumVersionString = "3.1.3"
let versionComparison = UIDevice.current.systemVersion.compare(minimumVersionString, options:
.numeric)
switch versionComparison {
    case .orderedSame, .orderedDescending:
        //current version is >= (3.1.3)
        break
    case .orderedAscending:
        //current version is < (3.1.3)
        fallthrough
    default:
        break;
}
```

Ziel c

```
NSString *version = @"3.1.3";
NSString *currentVersion = @"3.1.1";
NSComparisonResult result = [currentVersion compare:version options:NSNumericSearch];
switch(result) {
    case: NSOrderedAscending:
        //less than the current version
        break;
    case: NSOrderedDescending:
    case: NSOrderedSame:
        // equal or greater than the current version
        break;
}
```

Swift 2.0 und höher

```
if #available(iOS 9, *) {
```

```
// iOS 9
} else {
    // iOS 8 or earlier
}
```

Geräte-iOS-Version

Dadurch wird die aktuelle Systemversion angezeigt.

Ziel c

```
NSString *version = [[UIDevice currentDevice] systemVersion]
```

Schnell

```
let version = UIDevice.currentDevice().systemVersion
```

Swift 3

```
let version = UIDevice.current.systemVersion
```

IOS-Version überprüfen online lesen: <https://riptutorial.com/de/ios/topic/2194/ios-version-uberprufen>

Kapitel 85: Kategorien

Bemerkungen

Kategorien können verwendet werden, um die Methoden einer Klasse zu überschreiben. Auch wenn die Methode eigentlich privat ist. Auf die überschriebene Methode kann nicht von der Kategorie oder von einer anderen Stelle aus zugegriffen werden. Daher ist es wichtig, beim Hinzufügen von Methoden zu einer vorhandenen Klasse sicherzustellen, dass diese Methoden nicht bereits vorhanden sind.

Examples

Erstellen Sie eine Kategorie

Kategorien bieten die Möglichkeit, einem Objekt zusätzliche Funktionen hinzuzufügen, ohne das eigentliche Objekt zu subklassifizieren oder zu ändern.

Zum Beispiel möchten wir einige benutzerdefinierte Schriftarten festlegen. Erstellen Sie eine Kategorie, die der `UIFont` Klasse weitere Funktionen `UIFont` . Öffnen Sie Ihr Xcode-Projekt, klicken Sie auf Datei -> Neu -> Datei und wählen Sie Objective-C-Datei aus. Klicken Sie auf Weiter. Geben Sie Ihren Kategorienamen ein. "

Choose a template for your new file:

The dialog displays the following templates:

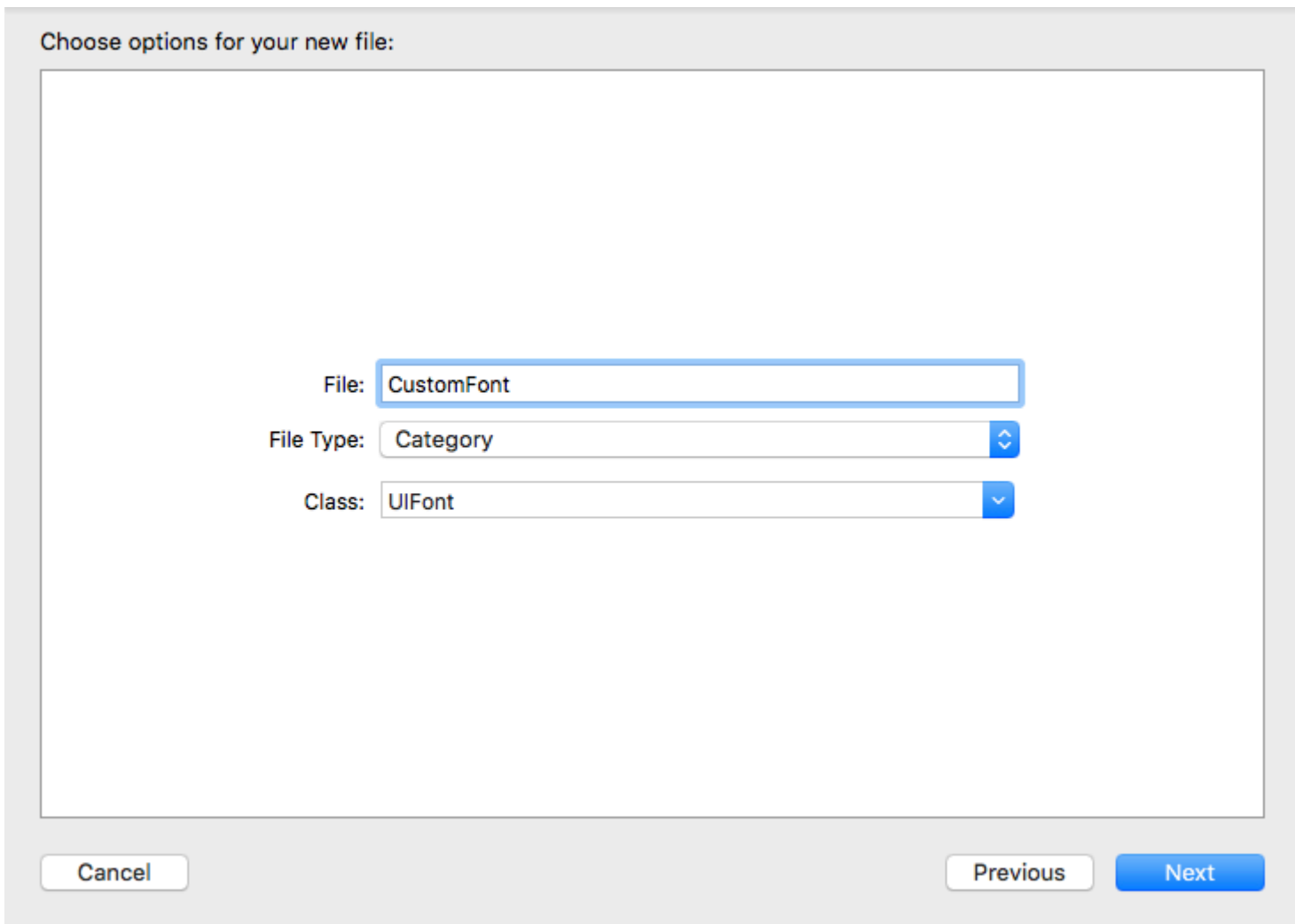
- iOS**
 - Source
 - User Interface
 - Core Data
 - Apple Watch
 - Resource
 - Other
- watchOS**
 - Source
 - User Interface
 - Core Data
 - Resource
 - Other
- tvOS**
 - Source
 - User Interface
 - Core Data
 - Resource

Available file templates:

- Cocoa Touch Class
- UI Test Case Class
- Unit Test Case Class
- Playground
- Swift File
- Objective-C File** (Selected)
- Header File
- C File
- C++ File
- Metal File

Objective-C File
An empty Objective-C file, category, protocol or extension.

Buttons: Cancel, Previous, Next



Deklarieren Sie die Kategoriemethode: -

Klicken Sie auf "UIFont + CustomFonts.h", um die Header-Datei der neuen Kategorie anzuzeigen. Fügen Sie der Schnittstelle den folgenden Code hinzu, um die Methode zu deklarieren.

```
@interface UIFont (CustomFonts)

+ (UIFont *)productSansRegularFontWithSize:(CGFloat) size;

@end
```

Implementieren Sie jetzt die Kategoriemethode: -

Klicken Sie auf "UIFont + CustomFonts.m", um die Implementierungsdatei der Kategorie anzuzeigen. Fügen Sie den folgenden Code hinzu, um eine Methode zu erstellen, mit der ProductSansRegular Font festgelegt wird.

```
+ (UIFont *)productSansRegularFontWithSize:(CGFloat) size{

    return [UIFont fontWithName:@"ProductSans-Regular" size:size];

}
```

Importieren Sie Ihre Kategorie

```
#import "UIFont+CustomFonts.h"
```

Legen Sie nun die Beschriftungsschriftart fest

```
[self.label setFont:[UIFont productSansRegularFontWithSize:16.0]];
```

Kategorien online lesen: <https://riptutorial.com/de/ios/topic/3633/kategorien>

Kapitel 86: Kernbewegung

Examples

Auf Barometer zugreifen, um relative Höhe zu erhalten

Schnell

Importieren Sie die Core Motion-Bibliothek:

```
import CoreMotion
```

Als Nächstes müssen wir ein `CMAltimeter` Objekt erstellen. Eine `CMAltimeter` besteht jedoch darin, es in `viewDidLoad()` zu erstellen. Auf diese Weise ist der Höhenmesser nicht zugänglich, wenn eine Methode darauf aufgerufen werden muss. `CMAltimeter` Sie `CMAltimeter` weiter und erstellen Sie Ihr `CMAltimeter` Objekt kurz vor dem `viewDidLoad()` :

```
let altimeter = CMAltimeter()
```

Jetzt:

1. Wir müssen mit der folgenden Methode prüfen, ob `relativeAltitude` überhaupt verfügbar ist:
`CMAltimeter.isRelativeAltitudeAvailable` .
2. Wenn dies `true` zurückgibt, können Sie die Überwachung der Höhenänderung mit `startRelativeAltitudeUpdatesToQueue`
3. Wenn keine Fehler aufgetreten sind, sollten Sie in der Lage sein, Daten aus den `relativeAltitude` und Druckeigenschaften abzurufen.

Nachfolgend ist die Definition einer Schaltflächenaktion angegeben, um die Überwachung mit unserem Barometer zu beginnen.

```
@IBAction func start(sender: AnyObject){
    if CMAltimeter.isRelativeAltitudeAvailable() {
        // 2
        altimeter.startRelativeAltitudeUpdatesToQueue(NSOperationQueue.mainQueue(), withHandler: {
            data, error in
                // 3
                if (error == nil) {
                    println("Relative Altitude: \(data.relativeAltitude)")
                    println("Pressure: \(data.pressure)")
                }
            })
    }
}
```

Kernbewegung online lesen: <https://riptutorial.com/de/ios/topic/7636/kernbewegung>

Kapitel 87: Kerndatei

Einführung

Core Data ist die Modellschicht Ihrer Anwendung im weitesten Sinne. Es ist das Modell im Model-View-Controller-Muster, das das iOS-SDK durchdringt.

Core Data ist weder die Datenbank Ihrer Anwendung noch eine API, um Daten in einer Datenbank zu speichern. Kerndaten sind ein Framework, das ein Objektdiagramm verwaltet. So einfach ist das. Core Data kann das Objektdiagramm beibehalten, indem es auf die Festplatte geschrieben wird. Dies ist jedoch nicht das Hauptziel des Frameworks.

Examples

Operationen mit Kerndaten

Um Kontext zu erhalten:

```
NSManagedObjectContext *context = ((AppDelegate*)[[UIApplication sharedApplication] delegate]).persistentContainer.viewContext;
```

Daten abrufen:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];  
NSError *error ;  
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Daten mit Sortierung abrufen:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];  
NSSortDescriptor *sortDescriptor = [NSSortDescriptor sortDescriptorWithKey:@"someKey"  
ascending:YES];  
fetchRequest.sortDescriptors = @[sortDescriptor];  
NSError *error ;  
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Daten hinzufügen:

```
NSManagedObject *entityNameObj = [NSEntityDescription  
insertNewObjectForEntityForName:@"EntityName" inManagedObjectContext:context];  
[entityNameObj setValue:@"someValue" forKey:@"someKey"];
```

Kontext speichern:

```
(((AppDelegate*)[[UIApplication sharedApplication] delegate]) saveContext);
```

Kerndatei online lesen: <https://riptutorial.com/de/ios/topic/9489/kerndatei>

Kapitel 88: Kernstandort

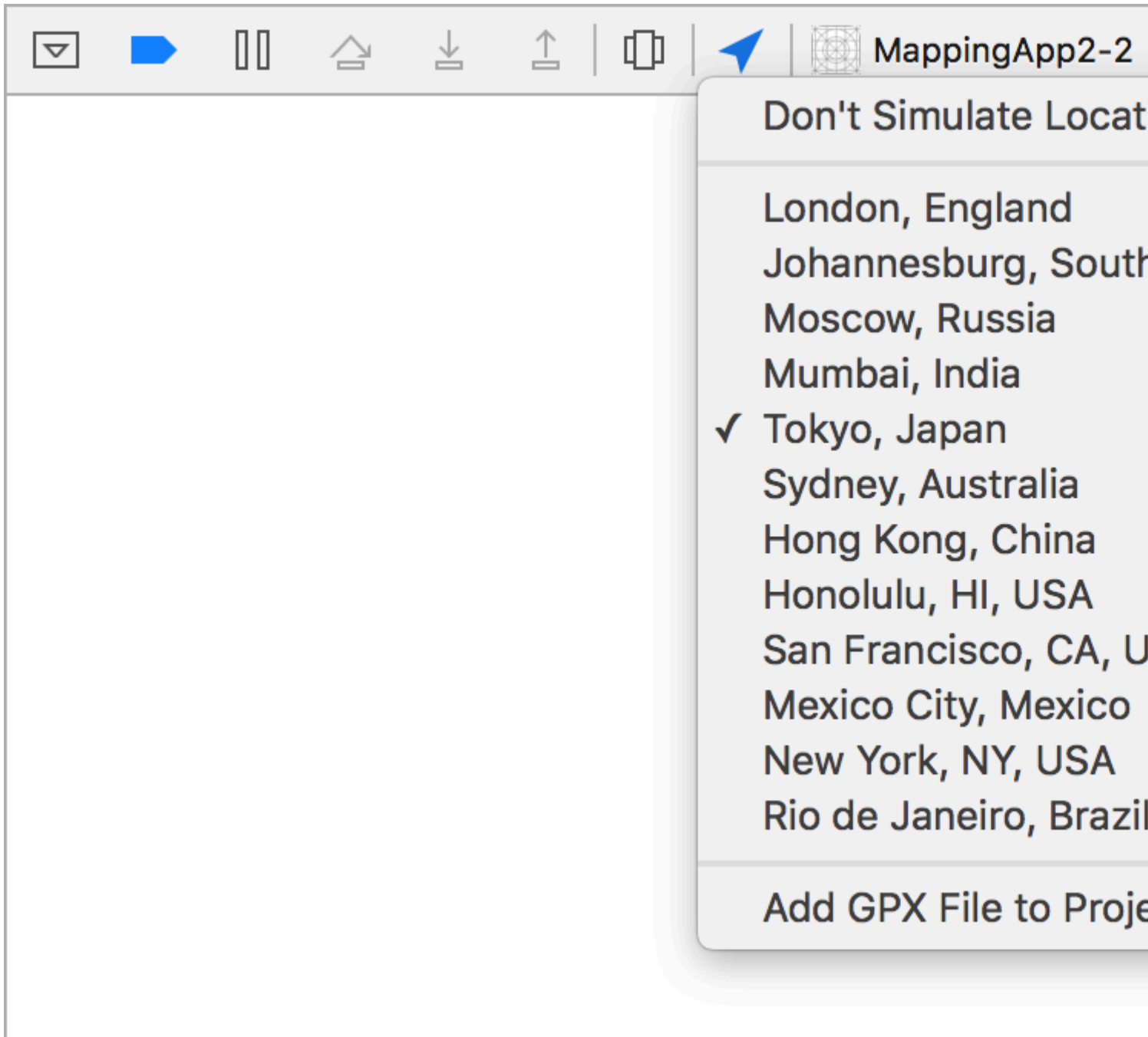
Syntax

1. gewünschte Genauigkeit
2. distanceFilter
3. requestLocation ()
4. startUpdatingLocation ()
5. allowDeferredLocationUpdates (untilTraveled: Zeitüberschreitung :)
6. startMonitoringSignificantLocationChanges ()
7. allowDeferredLocationUpdates (untilTraveled: Zeitüberschreitung :)
8. immer autorisiert
9. AuthorizedWhenInUse
10. locationManager (_: didChangeAuthorization :)

Bemerkungen

Simulieren Sie einen Ort zur Laufzeit

1. Führen Sie die App über Xcode aus.
2. Klicken Sie in der Debugleiste auf die Schaltfläche "Standort simulieren".
3. Wählen Sie einen Ort aus dem Menü.



Examples

[Link CoreLocation Framework](#)



MappingApp



General

Cap

PROJECT



MappingApp2-2

TARGETS



MappingApp2-2

Choose frameworks and libraries

CoreL

▼ iOS 9.2



CoreLocation.framework

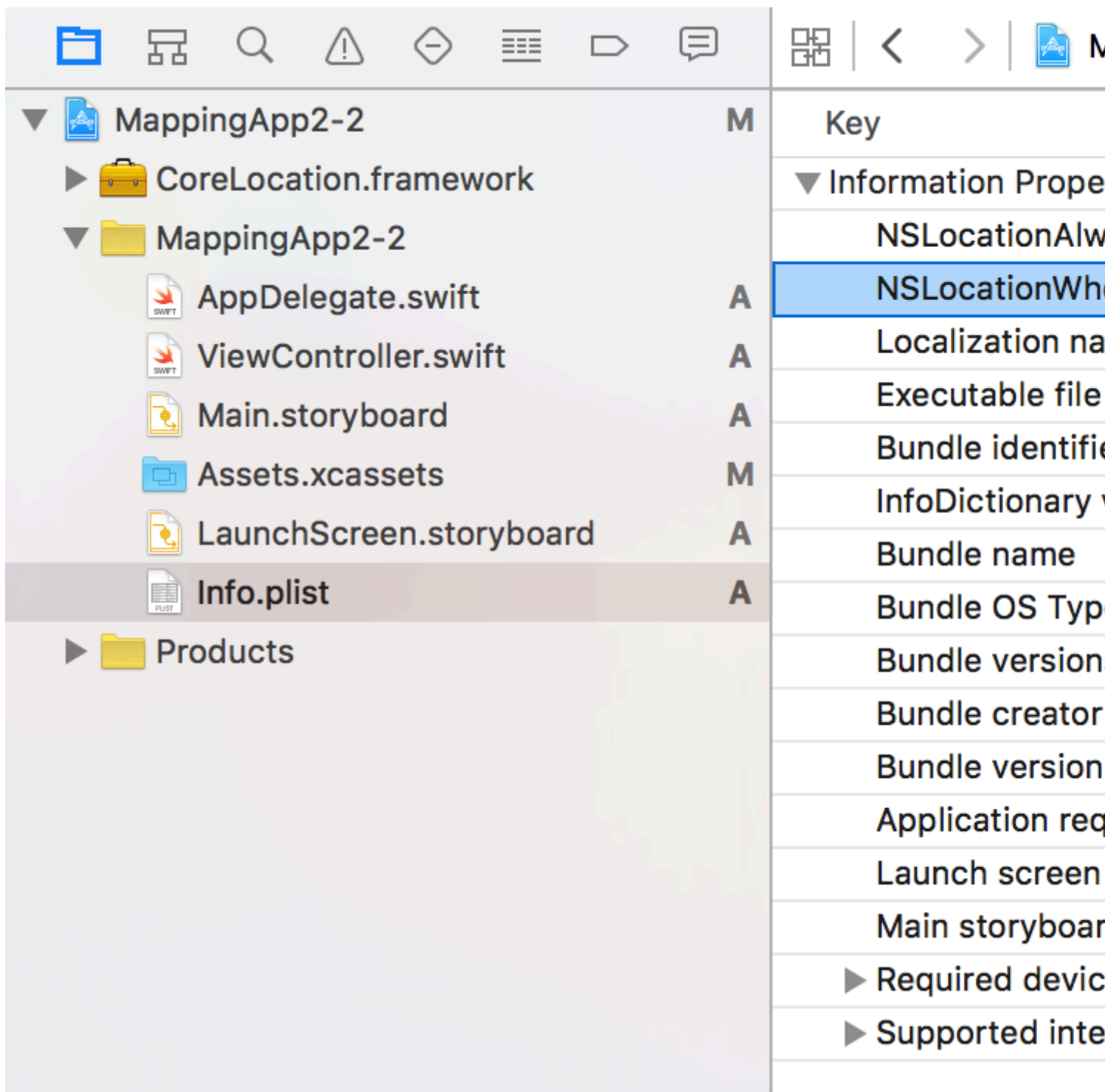


Developer Frameworks

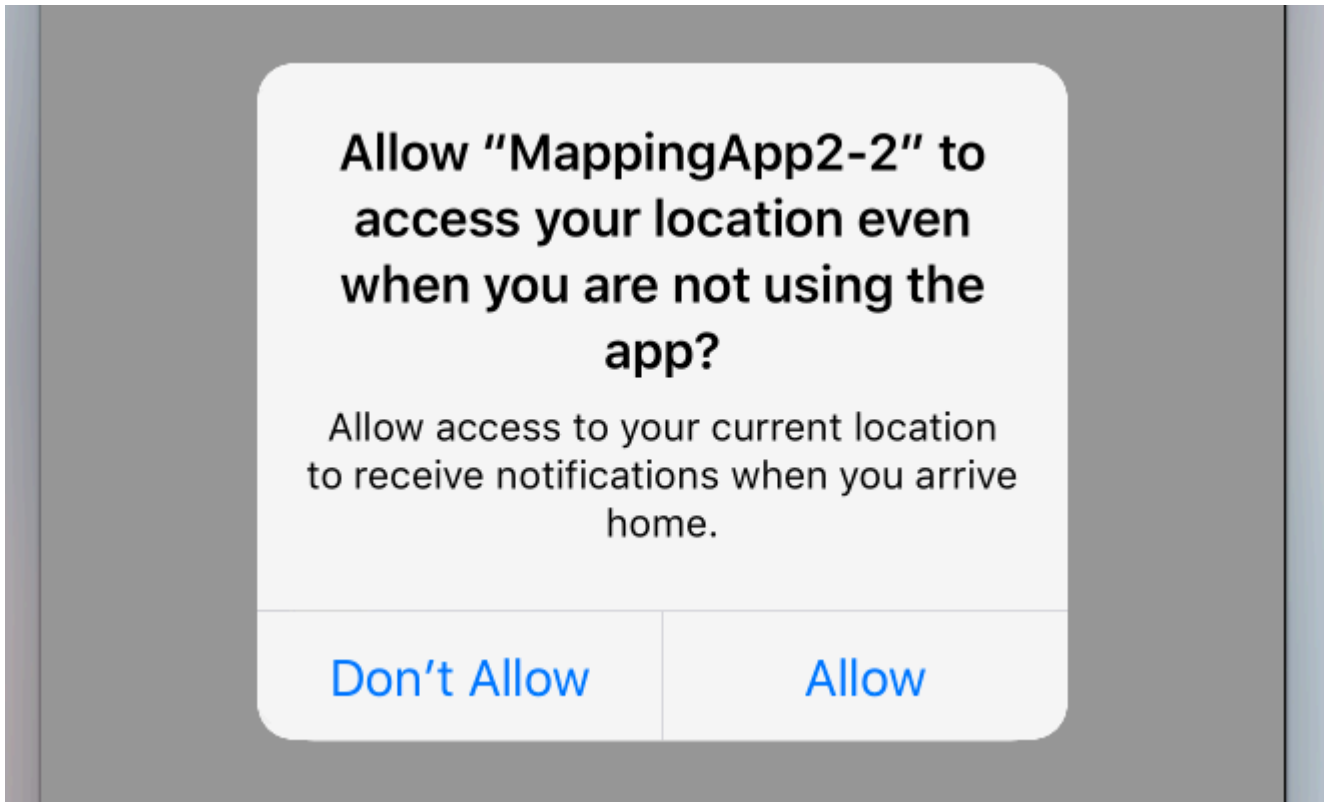
Add Other...



. Der Wert wird im `message` des Alert Controller verwendet.



Immer Standortdienst-Erlaubnis abrufen

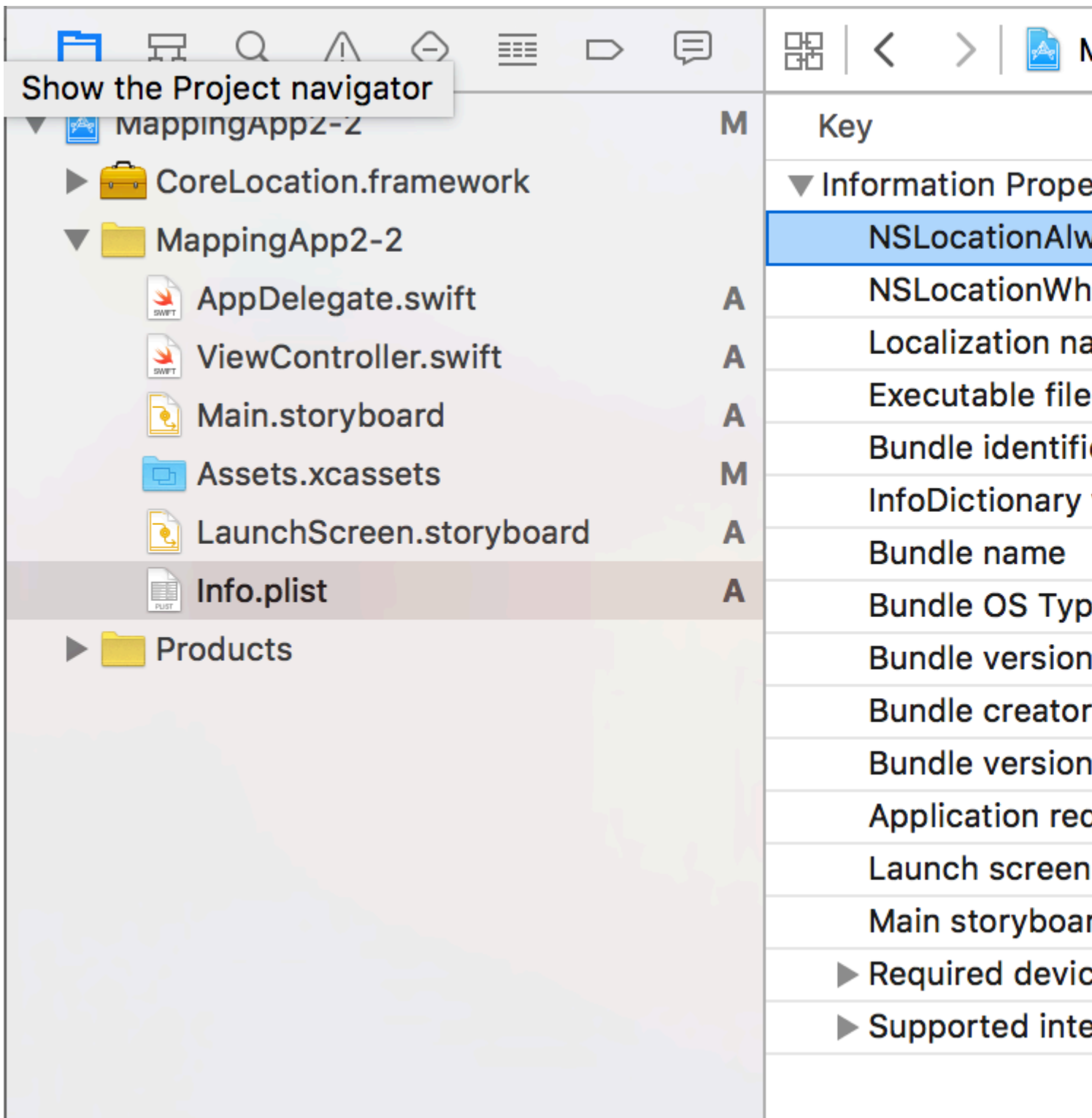


Um die Erlaubnis für die Verwendung von Ortungsdiensten zu erfragen, auch wenn die App nicht aktiv ist, verwenden Sie stattdessen den folgenden Anruf:

```
//Swift
locationManager.requestAlwaysAuthorization()

//Objective-C
[locationManager requestAlwaysAuthorization];
```

Fügen Sie dann den Schlüssel `NSLocationAlwaysUsageDescription` zu Ihrer `Info.plist` hinzu . Der Wert wird wieder in der `message` des Alert Controller verwendet.



Fügen Sie mithilfe der GPX-Datei einen eigenen benutzerdefinierten Speicherort hinzu

Um nach Standortdiensten zu suchen, benötigen wir ein echtes Gerät. Zu Testzwecken können wir auch den Simulator verwenden und unseren eigenen Standort hinzufügen, indem Sie die folgenden Schritte ausführen:

- Fügen Sie Ihrem Projekt eine neue GPX-Datei hinzu.
- in GPX-Datei fügen Sie Wegpunkte wie hinzu


```

<?xml version="1.0"?>
<gpx version="1.1" creator="Xcode">
<!--
    Provide one or more waypoints containing a latitude/longitude pair. If you provide one
    waypoint, Xcode will simulate that specific location. If you provide multiple
waypoints,
    Xcode will simulate a route visitng each waypoint.
-->
<wpt lat="52.599878" lon="4.702029">
    <name>location name (eg. Florida)</name>
</wpt>

```

- Gehen Sie dann zu Produkt -> Schema -> Schema bearbeiten und legen Sie im RUN-Standardverzeichnis den Namen Ihrer GPX-Datei fest.

Standortdienste im Hintergrund

Um Standard-Ortungsdienste zu verwenden, während sich die Anwendung im Hintergrund befindet, müssen Sie zunächst auf der Registerkarte Funktionen der `Background Modes` die `Background Modes` und `Location updates` auswählen.

Oder fügen Sie es direkt zur `Info.plist` hinzu.

```

<key>NSLocationAlwaysUsageDescription</key>
<string>I want to get your location Information in background</string>

<key>UIBackgroundModes</key>
<array>
    <string>location</string>
</array>

```

Dann müssen Sie den `CLLocationManager` einrichten

Ziel c

```

//The Location Manager must have a strong reference to it.
_locationManager = [[CLLocationManager alloc] init];
_locationManager.delegate = self;

//Request Always authorization (iOS8+)
if ([_locationManager respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [_locationManager requestAlwaysAuthorization];
}

//Allow location updates in the background (iOS9+)
if ([_locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)]) {
    _locationManager.allowsBackgroundLocationUpdates = YES;
}

[_locationManager startUpdatingLocation];

```

Schnell

```
self.locationManager.delegate = self
```

```
if #available (iOS 8.0,*) {
    self.locationManager.requestAlwaysAuthorization()
}

if #available (iOS 9.0,*) {
    self.locationManager.allowsBackgroundLocationUpdates = true
}

self.locationManager.startUpdatingLocation()
```

Kernstandort online lesen: <https://riptutorial.com/de/ios/topic/2937/kernstandort>

Kapitel 89: Kettenblöcke in einer Warteschlange (mit MKBlockQueue)

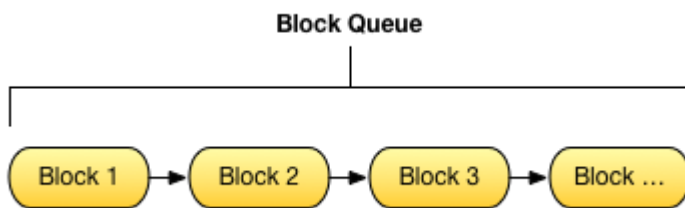
Einführung

Mit MKBlockQueue können Sie eine Kette von Blöcken erstellen und nacheinander in einer Warteschlange ausführen. Verglichen mit NSOperation entscheiden Sie mit MKBlockQueue selbst, wann ein Block abgeschlossen ist und wann die Warteschlange fortgesetzt werden soll. Sie können auch Daten von einem Block zum nächsten übergeben.

<https://github.com/MKGitHub/MKBlockQueue>

Examples

Beispielcode



```
// create the dictionary that will be sent to the blocks
var myDictionary:Dictionary<String, Any> = Dictionary<String, Any>()
myDictionary["InitialKey"] = "InitialValue"

// create block queue
let myBlockQueue:MKBlockQueue = MKBlockQueue()

// block 1
let b1:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    print("Block 1 started with dictionary: \(dictionary)")
    dictionary["Block1Key"] = "Block1Value"

    // tell this block is now completed
    blockQueueObserver.blockCompleted(with:&dictionary)
}

// block 2
let b2:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary
```

```

// test calling on main thread, async, with delay
DispatchQueue.main.asyncAfter(deadline:(.now() + .seconds(1)), execute:
{
    print("Block 2 started with dictionary: \(copyOfDictionary)")

    copyOfDictionary["Block2Key"] = "Block2Value"

    // tell this block is now completed
    blockQueueObserver.blockCompleted(with:&copyOfDictionary)
})
}

// block 3
let b3:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary

    // test calling on global background queue, async, with delay
    DispatchQueue.global(qos:.background).asyncAfter(deadline:(.now() + .seconds(1)), execute:
    {
        print("Block 3 started with dictionary: \(copyOfDictionary)")

        copyOfDictionary["Block3Key"] = "Block3Value"

        // tell this block is now completed
        blockQueueObserver.blockCompleted(with:&copyOfDictionary)
    })
}

// add blocks to the queue
myBlockQueue.addBlock(b1)
myBlockQueue.addBlock(b2)
myBlockQueue.addBlock(b3)

// add queue completion block for the queue
myBlockQueue.queueCompletedBlock(
{
    (dictionary:Dictionary<String, Any>) in
    print("Queue completed with dictionary: \(dictionary)")
})

// run queue
print("Queue starting with dictionary: \(myDictionary)")
myBlockQueue.run(with:&myDictionary)

```

Kettenblöcke in einer Warteschlange (mit MKBlockQueue) online lesen:

<https://riptutorial.com/de/ios/topic/9122/kettenblöcke-in-einer-warteschlange--mit-mkblockqueue->

Kapitel 90: Kontakte-Framework

Bemerkungen

Nützliche Links

- [Apple-Dokumentation](#)
- [Fragen zum Stapelüberlauf](#)
- [WWDC15-Sitzungsvideo](#)

Examples

Kontaktzugriff autorisieren

Framework importieren

Schnell

```
import Contacts
```

Ziel c

```
#import <Contacts/Contacts.h>
```

Zugänglichkeit prüfen

Schnell

```
switch CNContactStore.authorizationStatusForEntityType (CNEntityType.Contacts) {  
    case .Authorized: //access contacts  
    case .Denied, .NotDetermined: //request permission  
    default: break  
}
```

Ziel c

```
switch ([CNContactStore authorizationStatusForEntityType:CNEntityType.Contacts]) {
```

```
case CNAuthorizationStatus.Authorized:
    //access contacts
    break;
case CNAuthorizationStatus.Denied:
    //request permission
    break;
case CNAuthorizationStatus.NotDetermined:
    //request permission
    break;
}
```

Erlaubnis beantragen

Schnell

```
var contactStore = CKContactStore()
contactStore.requestAccessForEntityType(CKEntityType.Contacts, completionHandler: { (ok, _) ->
Void in
    if access{
        //access contacts
    }
}
```

Zugriff auf Kontakte

Filter anwenden

Um auf Kontakte zuzugreifen, sollten wir einen Filter vom Typ `NSPredicate` auf unsere `contactStore`-Variable anwenden, die wir im Beispiel für den Zugriff auf Kontaktzugriff definiert haben. Zum Beispiel möchten wir hier Kontakte mit dem eigenen Namen nach Namen sortieren:

Schnell

```
let predicate = CNContact.predicateForContactsMatchingName("Some Name")
```

Ziel c

```
NSPredicate *predicate = [CNContact predicateForContactsMatchingName:@"Some Name"];
```

Schlüssel zum Abrufen angeben

Hier möchten wir den Vornamen, den Nachnamen und das Profilbild des Kontakts abrufen:

Schnell

```
let keys = [CNContactGivenNameKey, CNContactFamilyNameKey, CNContactImageDataKey]
```

Kontakte abrufen

Schnell

```
do {
    let contacts = try contactStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)
} catch let error as NSError {
    //...
}
```

Zugriff auf Kontaktdaten

Schnell

```
print(contacts[0].givenName)
print(contacts[1].familyName)
let image = contacts[2].imageData
```

Kontakt hinzufügen

Schnell

```
import Contacts

// Creating a mutable object to add to the contact
let contact = CNMutableContact()

contact.imageData = NSData() // The profile picture as a NSData object

contact.givenName = "John"
contact.familyName = "Appleseed"

let homeEmail = CNLabeledValue(label:CNLabelHome, value:"john@example.com")
let workEmail = CNLabeledValue(label:CNLabelWork, value:"j.appleseed@icloud.com")
contact.emailAddresses = [homeEmail, workEmail]

contact.phoneNumbers = [CNLabeledValue(
    label:CNLabelPhoneNumberiPhone,
    value:CNPhoneNumber(stringValue:"(408) 555-0126"))]

let homeAddress = CNMutablePostalAddress()
```

```
homeAddress.street = "1 Infinite Loop"
homeAddress.city = "Cupertino"
homeAddress.state = "CA"
homeAddress.postalCode = "95014"
contact.postalAddresses = [CNLabeledValue(label:CNLabelHome, value:homeAddress)]

let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988 // You can omit the year value for a yearless birthday
contact.birthday = birthday

// Saving the newly created contact
let store = CNContactStore()
let saveRequest = CNSaveRequest()
saveRequest.addContact(contact, toContainerWithIdentifier:nil)
try! store.executeSaveRequest(saveRequest)
```

Kontakte-Framework online lesen: <https://riptutorial.com/de/ios/topic/5872/kontakte-framework>

Kapitel 91: Konvertieren Sie HTML in NSAttributedString-Zeichenfolge und umgekehrt

Examples

Ziel-C-Code zum Konvertieren einer HTML-Zeichenfolge in NSAttributedString und umgekehrt

Konvertierungscode für HTML in NSAttributedString: -

```
//HTML String
NSString *htmlString=[[NSString alloc]initWithFormat:@"<!DOCTYPE html><html><body><h1>My
First Heading</h1><p>My first paragraph.</p></body></html>"];
//Converting HTML string with UTF-8 encoding to NSAttributedString
NSAttributedString *attributedString = [[NSAttributedString alloc]
initWithData: [htmlString
dataUsingEncoding:NSUTF8StringEncoding]
options: @{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType }
documentAttributes: nil
error: nil ];
```

NSAttributedString in HTML-Konvertierung: -

```
//Dictionary to hold all the attributes of NSAttributedString
NSDictionary *documentAttributes = @{NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType};
//Saving the NSAttributedString with all its attributes as a NSData Entity
NSData *htmlData = [attributedString dataFromRange:NSMakeRange(0, attributedString.length)
documentAttributes:documentAttributes error:NULL];
//Convert the NSData into HTML String with UTF-8 Encoding
NSString *htmlString = [[NSString alloc] initWithData:htmlData
encoding:NSUTF8StringEncoding];
```

Konvertieren Sie HTML in NSAttributedString-Zeichenfolge und umgekehrt online lesen:

<https://riptutorial.com/de/ios/topic/7225/konvertieren-sie-html-in-nsattributed-zeichenfolge-und-umgekehrt>

Kapitel 92: Konvertieren Sie NSAttributedString in UIImage

Examples

NSAttributedString in UIImage-Konvertierung

Ziel c

```
NSMutableAttributedString *str = [[NSMutableAttributedString alloc] initWithString:@"Hello.
That is a test attributed string."];
[str addAttribute:NSBackgroundColorAttributeName value:[UIColor yellowColor]
range:NSMakeRange(3, 5)];
[str addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(10, 7)];
[str addAttribute:NSFontAttributeName value:[UIFont fontWithName:@"HelveticaNeue-Bold"
size:20.0] range:NSMakeRange(20, 10)];
UIImage *customImage = [self imageFromAttributedString:str];
```

Die Funktion `imageFromAttributedString` ist wie folgt definiert:

```
- (UIImage *)imageFromAttributedString:(NSAttributedString *)text
{
    UIGraphicsBeginImageContextWithOptions(text.size, NO, 0.0);

    // draw in context
    [text drawAtPoint:CGPointMake(0.0, 0.0)];

    // transfer image
    UIImage *image = [UIGraphicsGetImageFromCurrentImageContext()
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
    UIGraphicsEndImageContext();

    return image;
}
```

Konvertieren Sie NSAttributedString in UIImage online lesen:

<https://riptutorial.com/de/ios/topic/7242/konvertieren-sie-nsattributedString-in-uiimage>

Kapitel 93: Laufzeit in Objective-C

Examples

Zugeordnete Objekte verwenden

Zugeordnete Objekte sind hilfreich, wenn Sie vorhandenen Klassen Funktionalität hinzufügen möchten, für die ein Haltezustand erforderlich ist.

So fügen Sie beispielsweise zu jeder UIView einen Aktivitätsindikator hinzu:

Ziel-C-Implementierung

```
#import <objc/runtime.h>

static char ActivityIndicatorKey;

@implementation UIView (ActivityIndicator)

- (UIActivityIndicatorView *)activityIndicator {
    return (UIActivityIndicatorView *)objc_getAssociatedObject(self, &ActivityIndicatorKey);
}

- (void)setActivityIndicator: (UIActivityIndicatorView *)activityIndicator {
    objc_setAssociatedObject(self, &ActivityIndicatorKey, activityIndicator,
    OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (void)showActivityIndicator {
    UIActivityIndicatorView *activityIndicator = [[UIActivityIndicatorView alloc]
    initWithActivityIndicatorStyle: UIActivityIndicatorViewStyleGray];

    [self setActivityIndicator:activityIndicator];

    activityIndicator.center = self.center;
    activityIndicator.autoresizingMask = UIViewAutoresizingFlexibleTopMargin |
    UIViewAutoresizingFlexibleLeftMargin | UIViewAutoresizingFlexibleRightMargin |
    UIViewAutoresizingFlexibleBottomMargin;

    [activityIndicator startAnimating];

    [self addSubview: activityIndicator];
}

- (void)hideActivityIndicator {
    UIActivityIndicatorView * activityIndicator = [self activityIndicator];

    if (activityIndicator != nil) {
        [[self activityIndicator] removeFromSuperview];
    }
}

@end
```

Sie können auf die Objective-C-Laufzeit auch über Swift zugreifen:

SWIFT-Code

```
extension UIView {
    private struct AssociatedKeys {
        static var activityIndicator = "UIView.ActivityIndicatorView"
    }

    private var activityIndicatorView: UIActivityIndicatorView? {
        get {
            return objc_getAssociatedObject(self, &AssociatedKeys.activityIndicator) as?
                UIActivityIndicatorView
        }
        set (activityIndicatorView) {
            objc_setAssociatedObject(self, &AssociatedKeys.activityIndicator,
                activityIndicatorView, .OBJC_ASSOCIATION_RETAIN_NONATOMIC)
        }
    }

    func showActivityIndicator() {
        activityIndicatorView = UIActivityIndicatorView(activityIndicatorStyle: .gray)
        activityIndicatorView.center = center
        activityIndicatorView.autoresizingMask = [.flexibleLeftMargin, .flexibleRightMargin,
            .flexibleTopMargin, .flexibleBottomMargin]

        activityIndicatorView.startAnimating()

        addSubview(activityIndicatorView)
    }

    func hideActivityIndicator() {
        activityIndicatorView.removeFromSuperview()
    }
}
```

Laufzeit in Objective-C online lesen: <https://riptutorial.com/de/ios/topic/10120/laufzeit-in-objective-c>

Kapitel 94: Leitfaden zur Auswahl der besten iOS-Architekturmuster

Einführung

Entmystifizierung von MVC, MVP, MVVM und VIPER oder anderen Entwurfsmustern, um den besten Ansatz zum Erstellen einer App zu wählen

Examples

MVC-Muster

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}

class GreetingViewController : UIViewController { // View + Controller
    var person: Person!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}

// Assembling of MVC
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
view.person = model
```

MVP-Muster

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}
```

```

protocol GreetingView: class {
    func setGreeting(greeting: String)
}

protocol GreetingViewPresenter {
    init(view: GreetingView, person: Person)
    func showGreeting()
}

class GreetingPresenter : GreetingViewPresenter {
    unowned let view: GreetingView
    let person: Person
    required init(view: GreetingView, person: Person) {
        self.view = view
        self.person = person
    }
    func showGreeting() {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var presenter: GreetingViewPresenter!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.presenter.showGreeting()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}
// Assembling of MVP
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
let presenter = GreetingPresenter(view: view, person: model)
view.presenter = presenter

```

MVVM-Muster

```

import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}

protocol GreetingViewModelProtocol: class {

```

```

    var greeting: String? { get }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())? { get set } // function to
call when greeting did change
    init(person: Person)
    func showGreeting()
}

class GreetingViewModel : GreetingViewModelProtocol {
    let person: Person
    var greeting: String? {
        didSet {
            self.greetingDidChange?(self)
        }
    }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())?
    required init(person: Person) {
        self.person = person
    }
    func showGreeting() {
        self.greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
    }
}

class GreetingViewController : UIViewController {
    var viewModel: GreetingViewModelProtocol! {
        didSet {
            self.viewModel.greetingDidChange = { [unowned self] viewModel in
                self.greetingLabel.text = viewModel.greeting
            }
        }
    }
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self.viewModel, action: "showGreeting",
forControlEvents: .TouchUpInside)
    }
    // layout code goes here
}
// Assembling of MVVM
let model = Person(firstName: "David", lastName: "Blaine")
let viewModel = GreetingViewModel(person: model)
let view = GreetingViewController()
view.viewModel = viewModel

```

VIPER-Muster

```

import UIKit

struct Person { // Entity (usually more complex e.g. NSManagedObject)
    let firstName: String
    let lastName: String
}

struct GreetingData { // Transport data structure (not Entity)
    let greeting: String
    let subject: String
}

```

```

}

protocol GreetingProvider {
    func provideGreetingData()
}

protocol GreetingOutput: class {
    func receiveGreetingData(greetingData: GreetingData)
}

class GreetingInteractor : GreetingProvider {
    weak var output: GreetingOutput!

    func provideGreetingData() {
        let person = Person(firstName: "David", lastName: "Blaine") // usually comes from data
        access layer
        let subject = person.firstName + " " + person.lastName
        let greeting = GreetingData(greeting: "Hello", subject: subject)
        self.output.receiveGreetingData(greeting)
    }
}

protocol GreetingViewEventHandler {
    func didTapShowGreetingButton()
}

protocol GreetingView: class {
    func setGreeting(greeting: String)
}

class GreetingPresenter : GreetingOutput, GreetingViewEventHandler {
    weak var view: GreetingView!
    var greetingProvider: GreetingProvider!

    func didTapShowGreetingButton() {
        self.greetingProvider.provideGreetingData()
    }

    func receiveGreetingData(greetingData: GreetingData) {
        let greeting = greetingData.greeting + " " + greetingData.subject
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var eventHandler: GreetingViewEventHandler!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.eventHandler.didTapShowGreetingButton()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }
}

```



```
    }  
  
    // layout code goes here  
}  
// Assembling of VIPER module, without Router  
let view = GreetingViewController()  
let presenter = GreetingPresenter()  
let interactor = GreetingInteractor()  
view.eventHandler = presenter  
presenter.view = view  
presenter.greetingProvider = interactor  
interactor.output = presenter
```

Leitfaden zur Auswahl der besten iOS-Architekturmuster online lesen:

<https://riptutorial.com/de/ios/topic/10029/leitfaden-zur-auswahl-der-besten-ios-architekturmuster>

Kapitel 95: Lokalisierung

Einführung

Die **Lokalisierung** wird von iOS bereitgestellt, wodurch Ihre App in mehrere Sprachen übersetzt wird. Für die **Lokalisierung** ist **Internationalisierung** erforderlich. **Internationalisierung** ist ein Prozess, bei dem die iOS-App in der Lage ist, unterschiedliche Kulturen, Sprachen und Regionen anzupassen.

Examples

Lokalisierung in iOS

Erstellen Sie für jede Sprache eine eigene `Localizable.strings` Datei. Die rechte Seite wäre für jede Sprache anders. Betrachten Sie es als ein Schlüsselwertpaar:

```
"str" = "str-language";
```

Zugriff auf str in Objective-C:

```
//Try to provide description on the localized string to be able to create a proper  
documentation if needed  
NSString *str = NSLocalizedString(@"string", @"description of the string");
```

Zufahrtsstraße in Swift:

```
let str = NSLocalizedString("string", comment: "language");
```

Lokalisierung online lesen: <https://riptutorial.com/de/ios/topic/1579/lokalisierung>

Kapitel 96: MeinLayout

Einführung

MyLayout ist ein einfaches und einfaches Objective-C-Framework für das Layout von iOS-Ansichten. MyLayout bietet einige einfache Funktionen zum Erstellen einer Vielzahl komplexer Schnittstellen. Es integriert die Funktionen, einschließlich: Autolayout und SizeClass von iOS, fünf Layoutklassen von Android, Float und Flex-Box sowie Bootstrap von HTML / CSS. Sie können besuchen von:

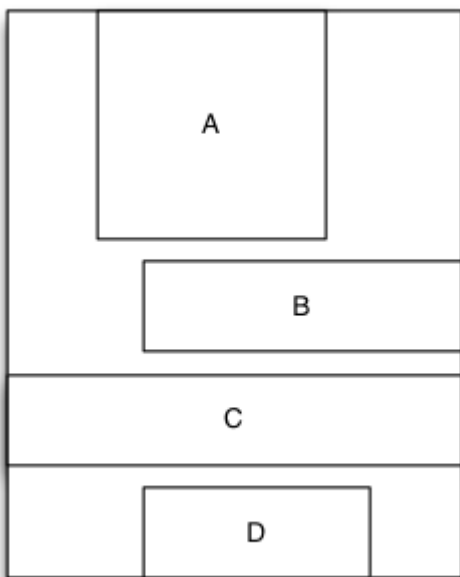
Ziel-C: <https://github.com/youngsoft/MyLinearLayout> Swift: <https://github.com/youngsoft/TangramKit>

Examples

Eine einfache Demo zur Verwendung von MyLayout

1. Es gibt eine Containeransicht S, deren Breite 100 ist und die Höhe auf die Höhe aller Unteransichten umbrochen wird. Es gibt vier Unteransichten A, B, C, D, die von oben nach unten angeordnet sind.
2. Der linke Rand der Unteransicht A ist 20% Breite von S, der rechte Rand ist 30% Breite von S, die Höhe entspricht der Breite von A.
3. Der linke Rand der Unteransicht B ist 40, die Breite ist bis zur Restbreite von S ausgefüllt, die Höhe ist 40. Die Breite der Unteransicht C ist bis S ausgefüllt, die Höhe ist 40.
4. Der rechte Rand der Unteransicht D ist 20, die Breite ist 50% Breite von S, die Höhe ist 40

wie unter der Abbildung:



```
MyLinearLayout *S = [MyLinearLayout
linearLayoutWithOrientation:MyLayoutViewOrientation_Vert];
S.subviewSpace = 10;
S.widthSize.equalTo(@100);

UIView *A = UIView.new;
A.leftPos.equalTo(@0.2);
A.rightPos.equalTo(@0.3);
A.heightSize.equalTo(A.widthSize);
[S addSubview:A];

UIView *B = UIView.new;
B.leftPos.equalTo(@40);
B.widthSize.equalTo(@60);
B.heightSize.equalTo(@40);
[S addSubview:B];

UIView *C = UIView.new;
C.leftPos.equalTo(@0);
C.rightPos.equalTo(@0);
C.heightSize.equalTo(@40);
[S addSubview:C];

UIView *D = UIView.new;
D.rightPos.equalTo(@20);
D.widthSize.equalTo(S.widthSize).multiply(0.5);
D.heightSize.equalTo(@40);
[S addSubview:D];
```

MeinLayout online lesen: <https://riptutorial.com/de/ios/topic/9692/meinlayout>

Kapitel 97: Mitteilungen

Syntax

- `UIUserNotificationSettings.types: UIUserNotificationType` // Eine Bitmaske der Benachrichtigungstypen, die Ihre App verwenden darf
- `UIUserNotificationSettings.categories: Set` // Die registrierten Aktionsgruppen der App

Parameter

Parameter	Beschreibung
Benutzerinformation	Ein Wörterbuch, das Remote-Benachrichtigungsinformationen enthält, einschließlich einer Ausweisnummer für das App-Symbol, einem Warnton, einer Alarmmeldung, einer Benachrichtigungs-ID und benutzerdefinierten Daten.

Examples

Gerät für Push-Benachrichtigungen registrieren

Fügen Sie Ihrer `AppDelegate`-Datei in der `didFinishLaunchingWithOptions` Methode den folgenden Code hinzu, um Ihr Gerät für Push-Benachrichtigungen zu registrieren:

Schnell

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    // Override point for customization after application launch.
    if UIDevice.currentDevice().systemVersion.compare(v, options: .NumericSearch) ==
    NSOrderedAscending {
        // Register for Push Notifications, if running iOS < 8
        if application.respondsToSelector("registerUserNotificationSettings:") {
            let types:UIUserNotificationType = (.Alert | .Badge | .Sound)
            let settings:UIUserNotificationSettings = UIUserNotificationSettings(forTypes:
            types, categories: nil)

            application.registerUserNotificationSettings(settings)
            application.registerForRemoteNotifications()
        } else {
            // Register for Push Notifications before iOS 8
            application.registerForRemoteNotificationTypes(.Alert | .Badge | .Sound)
        }
    } else {
        var center = UNUserNotificationCenter.currentNotificationCenter()
        center.delegate = self
        center.requestAuthorizationWithOptions((UNAuthorizationOptionSound |
        UNAuthorizationOptionAlert | UNAuthorizationOptionBadge)) {(granted: Bool, error: NSError) ->
```

```

Void in
    if !error {
        UIApplication.sharedApplication().registerForRemoteNotifications()
        // required to get the app to do anything at all about push notifications
        print("Push registration success.")
    } else {
        print("Push registration FAILED")
        print("ERROR: \(error.localizedFailureReason!) -
\ (error.localizedDescription)")
        print("SUGGESTIONS: \(error.localizedRecoveryOptions) -
\ (error.localizedRecoverySuggestion!)")
    }
}

return true
}

```

Ziel c

```

#define SYSTEM_VERSION_LESS_THAN(v) ([[UIDevice currentDevice] systemVersion] compare:v
options:NSNumericSearch] == NSOrderedAscending)

if( SYSTEM_VERSION_LESS_THAN( @"10.0" ) )
{
    if ([application respondsToSelector:@selector(isRegisteredForRemoteNotifications)])
    {
        // iOS 8 Notifications
        [application registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeSound | UIUserNotificationTypeAlert |
UIUserNotificationTypeBadge) categories:nil]];

        [application registerForRemoteNotifications];
    }
    else
    {
        // iOS < 8 Notifications
        [application registerForRemoteNotificationTypes:
(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeAlert |
UIRemoteNotificationTypeSound)];
    }
}
else
{
    UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
    center.delegate = self;
    [center requestAuthorizationWithOptions:(UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge) completionHandler:^(BOOL granted,
NSError * _Nullable error)
    {
        if( !error )
        {
            [[UIApplication sharedApplication] registerForRemoteNotifications]; // required
to get the app to do anything at all about push notifications
            NSLog( @"Push registration success." );
        }
        else
        {
            NSLog( @"Push registration FAILED" );
        }
    }
}
}

```

```

        NSLog( @"ERROR: %@ - %@", error.localizedFailureReason,
error.localizedDescription );
        NSLog( @"SUGGESTIONS: %@ - %@", error.localizedRecoveryOptions,
error.localizedRecoverySuggestion );
    }
}];
}

//to check if your App lunch from Push notification
//-----
//Handel Push notification
if (launchOptions != nil)
{
    // Here app will open from pushnotification
    //RemoteNotification
    NSDictionary* dictionary1 = [launchOptions
objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
    //LocalNotification
    NSDictionary* dictionary2 = [launchOptions
objectForKey:UIApplicationLaunchOptionsLocalNotificationKey];
    if (dictionary1 != nil)
    {
        //RemoteNotification Payload
        NSLog(@"Launched from push notification: %@", dictionary1);
        //here handle your push notification
    }
    if (dictionary2 != nil)
    {
        NSLog(@"Launched from dictionary2dictionary2dictionary2 notification: %@",
dictionary2);
        double delayInSeconds = 7;
        dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW,
(int64_t)(delayInSeconds * NSEC_PER_SEC));
        dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
            // [self addMessageFromRemoteNotification:dictionary2 updateUI:NO];
        });
    }
}
else
{}
//-----

```

Der obige Code versucht, mit dem APNs-Server zu kommunizieren, um ein Gerätetoken zu erhalten (Voraussetzung, dass in Ihrem iOS-Bereitstellungsprofil APNs aktiviert sind).

Sobald eine zuverlässige Verbindung zum APNs-Server hergestellt ist, stellt der Server ein Gerätetoken zur Verfügung.

Nachdem Sie den obigen Code hinzugefügt haben, fügen Sie der `AppDelegate` Klasse diese Methoden `AppDelegate` :

Schnell

```

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {

```

```

    print("DEVICE TOKEN = \(deviceToken)")
}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}

```

Ziel c

```

- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    NSString * deviceTokenString = [[[deviceToken description]
        stringByReplacingOccurrencesOfString:@"<" withString:@""]
        stringByReplacingOccurrencesOfString:@">" withString:@""]
        stringByReplacingOccurrencesOfString:@" " withString:@""];

    NSLog(@"The generated device token string is : %@",deviceTokenString);
}

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
    NSLog(@"Failed to get token, error: %@", error.description);
}

```

Die obigen Methoden werden entsprechend dem Registrierungserfolgs- oder -ausfallszenario aufgerufen.

Erfolgsszenario fordert:

Schnell

```

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}

```

In Swift3:

```

@objc(userNotificationCenter:willPresentNotification:withCompletionHandler:) @available(iOS
10.0, *)
func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification:
UNNotification, withCompletionHandler completionHandler: @escaping
(UNNotificationPresentationOptions) -> Void)
{
    //To show notifications in foreground.
    print("Userinfo2 \(notification.request.content.userInfo)")
}

```

Ziel c


```

- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    if(application.applicationState == UIApplicationStateInactive) {
        NSLog(@"Inactive - the user has tapped in the notification when app was closed or in
background");
        //do some tasks
        [self handelPushNotification:userInfo];
    }
    else if (application.applicationState == UIApplicationStateBackground) {
        NSLog(@"application Background - notification has arrived when app was in background");
        [self handelPushNotification:userInfo];
    }
    else {
        NSLog(@"application Active - notication has arrived while app was opened");
        //Show an in-app banner
        //do tasks
    }
}
}

```

Fehlerszenario-Aufrufe:

Schnell

```

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}

```

Ziel c

```

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error

```

Hinweis

Wenn keine der oben genannten Methoden aufgerufen wird, kann Ihr Gerät keine zuverlässige Verbindung mit dem APNs-Server herstellen, was möglicherweise auf Probleme beim Internetzugang zurückzuführen ist.

Überprüfen Sie, ob Ihre App bereits für die Push-Benachrichtigung registriert ist

Schnell

```

let isPushEnabled = UIApplication.sharedApplication().isRegisteredForRemoteNotifications()

```

Registrieren für die (nicht interaktive) Push-Benachrichtigung

Es wird empfohlen, die Logik der Registrierung für Push-Benachrichtigungen in `AppDelegate.swift` da die Callback-Funktionen (Erfolg, Fehler) als ihre bezeichnet werden. Um sich anzumelden, gehen Sie wie folgt vor:

```
let application = UIApplication.sharedApplication()
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
application.registerUserNotificationSettings(settings)
```

Dann wird die Callback-Funktion `didRegisterUserNotificationSettings` aufgerufen. In diesem Fall lösen Sie das Register einfach folgendermaßen aus:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    application.registerForRemoteNotifications()
}
```

In diesem Fall wird ein Systemalarm angezeigt, in dem Sie aufgefordert werden, eine Push-Benachrichtigung zu erhalten. Eine der folgenden Rückruffunktionen wird aufgerufen:

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    let tokenChars = UnsafePointer<CChar>(deviceToken.bytes)
    var tokenString = ""

    for i in 0..
```

In sehr seltenen Fällen werden weder Erfolgs- noch Fehlerrückruffunktionen aufgerufen. Dies ist der Fall, wenn Sie Probleme mit der Internetverbindung haben oder die APNS-Sandbox inaktiv ist. Das System führt einen API-Aufruf an APNS aus, um einige Überprüfungen durchzuführen. Andernfalls wird keine der beiden Callbacks-Funktionen aufgerufen. Besuchen Sie den [Apple-Systemstatus](#), um sicherzustellen, dass alles in Ordnung ist.

Umgang mit Push-Benachrichtigung

Sobald der Benutzer eine Push-Benachrichtigung anklickt, wird die folgende Rückruffunktion aufgerufen. Sie können die JSON-Analyse analysieren, um bestimmte Informationen vom Backend zu erhalten, die Ihnen beim Deep Linking helfen:

Schnell

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
```

```

: AnyObject]) {
    print("Received notification: \(userInfo)")
}

```

Ziel c

```

- (void)application:(UIApplication *)application didReceiveRemoteNotification: (NSDictionary
*)userInfo
{
    NSLog(@"Received notification: %@", userInfo);
}

```

iOS 10

```

#define SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO(v) ([[UIDevice currentDevice] systemVersion]
compare:v options:NSNumericSearch] != NSOrderedAscending)

- (void) application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
*)userInfo fetchCompletionHandler:(void
(^)(UIBackgroundFetchResult))completionHandler
{
    // iOS 10 will handle notifications through other methods
    NSLog(@"Received notification: %@", userInfo);

    if( SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO( @"10.0" ) )
    {
        NSLog( @"iOS version >= 10. Let NotificationCenter handle this one." );
        // set a member variable to tell the new delegate that this is background
        return;
    }
    NSLog( @"HANDLE PUSH, didReceiveRemoteNotification: %@", userInfo );

    // custom code to handle notification content

    if( [UIApplication sharedApplication].applicationState == UIApplicationStateInactive )
    {
        NSLog( @"INACTIVE" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
    else if( [UIApplication sharedApplication].applicationState == UIApplicationStateBackground
)
    {
        NSLog( @"BACKGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
    else
    {
        NSLog( @"FOREGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void (^)(UNNotificationPresentationOptions
options))completionHandler
{
    NSLog( @"Handle push from foreground" );
}

```

```
// custom code to handle push while app is in the foreground
NSLog(@"%@", notification.request.content.userInfo);
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
withCompletionHandler:(void (^)(void))completionHandler
{

    NSLog(@"Handle push from background or closed");
    // if you set a member variable in didReceiveRemoteNotification, you will know if this is
    from closed or background
    NSLog(@"%@", response.notification.request.content.userInfo);
}
```

Registrieren der App-ID für die Verwendung mit Push-Benachrichtigungen

Dinge, die du brauchst

- Eine bezahlte Apple Developer Program-Mitgliedschaft
- Eine gültige App-ID und ein gültiger Bezeichner für Ihre App (z. B. com.example.MyApp), der zuvor nicht verwendet wurde
- Zugriff auf developer.apple.com und Member Center
- Ein iOS-Gerät zum Testen (da Push-Benachrichtigungen auf dem Simulator nicht funktionieren)

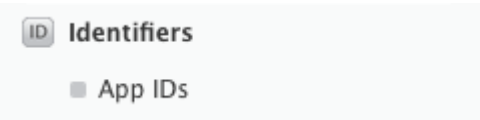
Aktivieren des APNs-Zugriffs für die App-ID im Apple Developer Center

1- Melden Sie sich im Mitgliedercenter von developer.apple.com an (der Konto-Link auf der Startseite).

A light gray rectangular button with the word "Account" in a dark gray sans-serif font.

2- Gehe zu "Zertifikate"

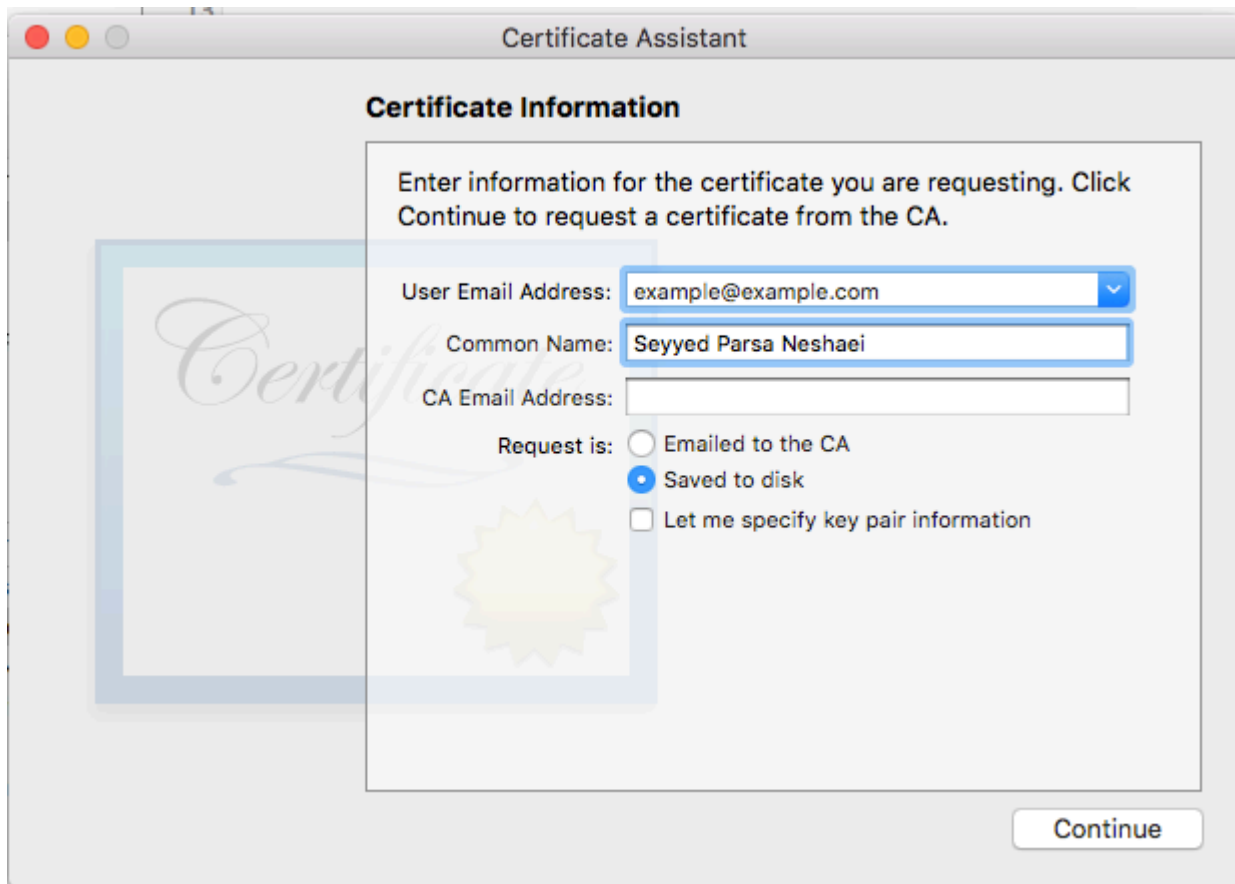
3- Wählen Sie "App ID" im linken Bereich aus

A screenshot of the Apple Developer Center interface. It shows a light gray box with a dark gray header "ID Identifiers" and a sub-item "App IDs" with a small dark gray square to its left.

4- Klicken Sie oben rechts auf "+"

A light gray square button with a dark gray border and a dark gray plus sign (+) in the center.

- 5- App-ID mit Push-Benachrichtigungen hinzufügen aktiviert
- 6- Klicken Sie auf die erstellte App-ID und wählen Sie Bearbeiten
- 7- Klicken Sie im Bereich Push-Benachrichtigungen auf Konfigurieren
- 8- Öffnen Sie die App "Keychain Access" in Ihrem Mac
- 9- Klicken Sie im Menü Schlüsselzugriff auf Zertifikatassistent -> Zertifikat von einer Zertifizierungsstelle anfordern
- 10- Geben Sie Ihre Mail in das erste Textfeld ein
- 11- Geben Sie Ihren Namen in das zweite Textfeld ein



- 12- Lassen Sie die CA-E-Mail-Adresse leer
- 13- Wählen Sie Gespeichert auf Datenträger statt E-Mail an die Zertifizierungsstelle
- 14- Klicken Sie auf Weiter und laden Sie die generierte Datei hoch
- 15- Laden Sie die generierte Datei von Apple herunter und öffnen Sie sie, während Keychain Access geöffnet ist

Aktivieren des APNs-Zugriffs in Xcode

1- Wählen Sie Ihr Projekt aus

2- Öffnen Sie die Registerkarte Funktionen

3- Push-Benachrichtigungen suchen und einschalten

4-Finden Sie die Hintergrundmodi, schalten Sie sie ein und überprüfen Sie die Remote-Benachrichtigungen

Registrierung von Push-Benachrichtigungen aufheben

Um die Registrierung von Remote-Benachrichtigungen programmatisch aufzuheben, können Sie verwenden

Ziel c

```
[[UIApplication sharedApplication] unregisterForRemoteNotifications];
```

Schnell

```
UIApplication.sharedApplication().unregisterForRemoteNotifications()
```

Dies ist vergleichbar mit der Einstellung Ihres Telefons und dem manuellen Deaktivieren von Benachrichtigungen für die Anwendung.

HINWEIS: In seltenen Fällen benötigen Sie dies (z. B. wenn Ihre App Push-Benachrichtigungen nicht mehr unterstützt).

Wenn Sie dem Benutzer nur erlauben möchten, Benachrichtigungen vorübergehend zu deaktivieren. Sie sollten eine Methode implementieren, um das Gerätetoken in der Datenbank Ihres Servers zu entfernen. Wenn Sie die Benachrichtigung nur lokal auf Ihrem Gerät deaktivieren, sendet der Server weiterhin Nachrichten.

Einstellen der Anwendungssymbol-Ausweisnummer

Verwenden Sie den folgenden Code, um die `someNumber` in Ihrer Anwendung `someNumber` (vorausgesetzt, dass zuvor `someNumber` deklariert wurde):

Ziel c

```
[UIApplication sharedApplication].applicationIconBadgeNumber = someNumber;
```

Schnell

```
UIApplication.shared.applicationIconBadgeNumber = someNumber
```

Um das Abzeichen vollständig zu *entfernen*, setzen `someNumber = 0` einfach `someNumber = 0`.

Testen von Push-Benachrichtigungen

Es empfiehlt sich immer, die Funktionsweise von Push-Benachrichtigungen zu testen, noch bevor Sie die Serverseite für sie vorbereitet haben. Nur um sicherzustellen, dass alles auf Ihrer Seite richtig eingerichtet ist. Es ist ziemlich einfach, sich selbst eine Push-Benachrichtigung mit dem folgenden PHP-Skript zu senden.

1. Speichern Sie das Skript als Datei (zum Beispiel send_push.php) im selben Ordner wie Ihr Zertifikat (Entwicklung oder Produktion).
2. Bearbeiten Sie es, um das Gerätetoken und das Kennwort aus dem Zertifikat zu speichern
3. Wählen Sie den korrekten Pfad zum Öffnen einer Verbindung, dev_path oder prod_path (hier erfolgt die Verbindung zum APNS-Server im Skript.)
4. cd in den Ordner im Terminal und den Befehl 'php send_push' ausführen
5. Erhalten Sie die Benachrichtigung auf Ihrem Gerät

```
<?php

// Put your device token here (without spaces):
$deviceToken = '20128697f872d7d39e48c4a61f50cb11d77789b39e6fc6b4cd7ec80582ed5229';
// Put your final pem cert name here. it is supposed to be in the same folder as this script
$cert_name = 'final_cert.pem';
// Put your private key's passphrase here:
$passphrase = '1234';

// sample point
$alert = 'Hello world!';
$event = 'new_incoming_message';

// You can choose either of the paths, depending on what kind of certificate you are using
$dev_path = 'ssl://gateway.sandbox.push.apple.com:2195';
$prod_path = 'ssl://gateway.push.apple.com:2195';

////////////////////////////////////

$ctx = stream_context_create();
stream_context_set_option($ctx, 'ssl', 'local_cert', $cert_name);
stream_context_set_option($ctx, 'ssl', 'passphrase', $passphrase);

// Open a connection to the APNS server
$fp = stream_socket_client(
    $dev_path, $err,
    $errstr, 60, STREAM_CLIENT_CONNECT|STREAM_CLIENT_PERSISTENT, $ctx);

if (!$fp)
    exit("Failed to connect: $err $errstr" . PHP_EOL);

echo 'Connected to APNS' . PHP_EOL;

// Create the payload body
// it should be as short as possible
// if the notification doesnt get delivered that is most likely
// because the generated message is too long
$body['aps'] = array(
    'alert' => $alert,
    'sound' => 'default',
    'event' => $event
```

```

);

// Encode the payload as JSON
$payload = json_encode($body);

// Build the binary notification
$msg = chr(0) . pack('n', 32) . pack('H*', $deviceToken) . pack('n', strlen($payload)) .
$payload;

// Send it to the server
$result = fwrite($fp, $msg, strlen($msg));

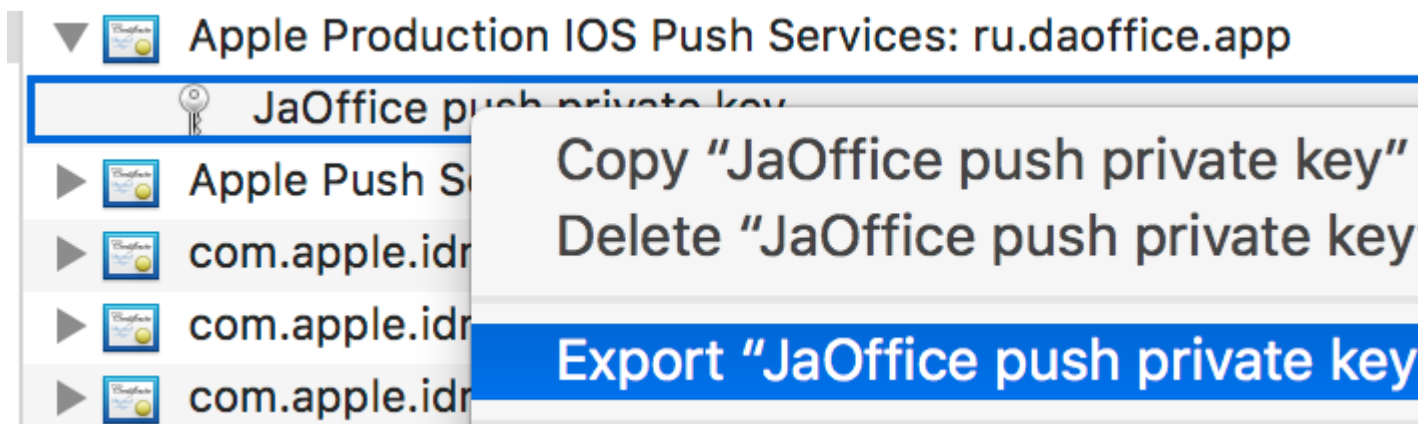
if (!$result)
    echo 'Message not delivered' . PHP_EOL;
else
    echo 'Message successfully delivered' . PHP_EOL;

// Close the connection to the server
fclose($fp);

```

Generierung eines .pem-Zertifikats aus Ihrer .cer-Datei, um sie an den Server-Entwickler weiterzuleiten

1. Aps.cer in einem Ordner speichern
2. Öffnen Sie "Schlüsselbundzugriff" und exportieren Sie den Schlüssel, der sich unter diesem Zertifikat befindet, in eine P12-Datei (nennen Sie sie key.p12). Klicken Sie dazu mit der rechten Maustaste darauf und wählen Sie Exportieren. Speichern Sie es in demselben Ordner wie in Schritt 1. Beim Export werden Sie aufgefordert, ein Kennwort einzugeben. Machen Sie sich etwas aus und merken Sie es sich.



3. cd zu diesem Ordner in Terminal und führen Sie die folgenden Befehle aus:
4. Konvertieren Sie .cer in ein .pem-Zertifikat

```
openssl x509 -in aps.cer -inform der -out aps.pem
```

5. Konvertieren Sie Ihren Schlüssel in das PEM-Format. Um den Schlüssel zu öffnen, geben Sie in Schritt 2 das Kennwort ein, mit dem Sie es aus dem Schlüsselbund exportiert haben. Geben Sie dann ein anderes Kennwort ein, das die exportierte Datei schützt. Sie werden

zweimal zur Bestätigung aufgefordert.

```
openssl pkcs12 -nocerts -out key.pem -in key.p12
```

6. Fügen Sie die Dateien in einer endgültigen Datei zusammen

```
cat key.pem aps.pem > final_cert.pem
```

7. Das final_cert.pem ist das Endergebnis. Geben Sie es mit dem Kennwort aus Schritt 5 an Serverentwickler weiter, damit sie das geschützte Zertifikat verwenden können.

Mitteilungen online lesen: <https://riptutorial.com/de/ios/topic/3492/mitteilungen>

Kapitel 98: MKDistanceFormatter

Examples

String aus der Ferne

Bei einer `CLLocationDistance` (einfach ein `Double` das Meter darstellt), geben Sie eine vom Benutzer lesbare Zeichenfolge aus:

```
let distance = CLLocationDistance(42)
let formatter = MKDistanceFormatter()
let answer = formatter.stringFromDistance(distance)
// answer = "150 feet"
```

Ziel c

```
CLLocationDistance distance=42;
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
NSString *answer=[formatter stringFromDistance:distance];
// answer = "150 feet"
```

Standardmäßig wird das Gebietsschema des Benutzers berücksichtigt.

Entfernungseinheiten

import MapKit Sie die units auf eine der `.Default`, `.Metric`, `.Imperial`, `.ImperialWithYards` :

```
formatter.units = .Metric
var answer = formatter.stringFromDistance(distance)
// "40 m"

formatter.units = .ImperialWithYards
answer = formatter.stringFromDistance(distance)
// "50 yards"
```

Ziel c

```
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
formatter.units=MKDistanceFormatterUnitsMetric;
NSString *answer=[formatter stringFromDistance:distance];
//40 m

formatter.units=MKDistanceFormatterUnitsImperialWithYards;
NSString *answer=[formatter stringFromDistance:distance];
//50 yards
```

Einheitenstil

Setzen Sie `unitStyle` auf `.Default`, `.Abbreviated`, `.Full` :

```
formatter.unitStyle = .Full
var answer = formatter.stringFromDistance(distance)
// "150 feet"

formatter.unitStyle = .Abbreviated
answer = formatter.stringFromDistance(distance)
// "150 ft"
```

Ziel c

```
formatter.unitStyle=MKDistanceFormatterUnitStyleFull;
NSString *answer=[formatter stringFromDistance:distance];
// "150 feet"

formatter.unitStyle=MKDistanceFormatterUnitStyleAbbreviated;
NSString *answer=[formatter stringFromDistance:distance];
// "150 ft"
```

MKDistanceFormatter online lesen: <https://riptutorial.com/de/ios/topic/6677/mkdistanceformatter>

Kapitel 99: MKMapView

Examples

MKMapView hinzufügen

Schnell

```
let mapView = MKMapView(frame: CGRect(x: 0, y: 0, width: 320, height: 500))
```

Es wird empfohlen, MapView als Eigenschaft des enthaltenen `ViewController` zu speichern, da Sie in komplexeren Implementierungen darauf zugreifen möchten.

Ziel c

```
self.map = [[MKMapView alloc] initWithFrame:CGRectMake(0, 0, self.view.frame.size.width, self.view.frame.size.height)];  
[self.view addSubview:self.map];
```

Kartentyp ändern

Es gibt 5 verschiedene Typen ([MKMapType](#)), die `MKMapView` anzeigen kann.

iPhone OS 3

.Standard

Zeigt eine Straßenkarte mit der Position aller Straßen und einigen Straßennamen an.

Schnell 2

```
mapView.mapType = .Standard
```

Swift 3

```
mapView.mapType = .standard
```

Ziel c

```
_mapView.mapType = MKMapTypeStandard;
```



iPhone OS 3

.Satellit

Zeigt Satellitenbilder der Region an.

Schnell 2

```
mapView.mapType = .Satellite
```

Swift 3

```
mapView.mapType = .satellite
```

Ziel c

```
_mapView.mapType = MKMapTypeSatellite;
```



iOS 9

.satelliteFlyover

Zeigt ein Satellitenbild des Bereichs mit Überflugdaten an, sofern verfügbar.

Schnell 2

```
mapView.mapType = .SatelliteFlyover
```

Swift 3

```
mapView.mapType = .satelliteFlyover
```

Ziel c

```
_mapView.mapType = MKMapTypeSatelliteFlyover;
```

iPhone OS 3

.hybrid

Zeigt ein Satellitenbild des Bereichs an, auf dem die Straßen- und Straßennameninformationen angezeigt werden.

Schnell 2

```
mapView.mapType = .Hybrid
```

Swift 3

```
mapView.mapType = .hybrid
```

Ziel c

```
_mapView.mapType = MKMapTypeHybrid;
```



iOS 9

.hybridFlyover

Zeigt ein hybrides Satellitenbild mit Überflugdaten an, sofern verfügbar.

Schnell 2

```
mapView.mapType = .HybridFlyover
```

Swift 3


```
mapView.mapType = .hybridFlyover
```

Ziel c

```
_mapView.mapType = MKMapTypeHybridFlyover;
```

Stellen Sie Zoom / Region für die Karte ein

Wenn Sie eine Zoomstufe einstellen möchten, können Sie den Standort des Benutzers mit dem Standort des Benutzers als Zentrum und 2 km der Fläche als Radius vergrößern. Dann verwenden wir folgenden Code

```
MKUserLocation *userLocation = _mapView.userLocation;  
MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance  
(userLocation.location.coordinate, 2000, 2000);  
[_mapView setRegion:region animated:NO];
```

Lokale Suchimplementierung mit MKLocalSearch

Mit MKLocalSearch können Benutzer nach Orten mit natürlichen Zeichenfolgen wie "Fitnessstudio" suchen. Sobald die Suche abgeschlossen ist, gibt die Klasse eine Liste von Orten innerhalb eines angegebenen Bereichs zurück, die der Suchzeichenfolge entsprechen.

Suchergebnisse sind in Form von MKMapItem innerhalb des MKLocalSearchResponse-Objekts.

Versuchen wir es mit Beispiel

```
MKLocalSearchRequest *request =  
    [[MKLocalSearchRequest alloc] init]; //initialising search request  
request.naturalLanguageQuery = @"Gym"; // adding query  
request.region = _mapView.region; //setting region  
MKLocalSearch *search =  
    [[MKLocalSearch alloc] initWithRequest:request]; //initiate search  
  
[search startWithCompletionHandler:^(MKLocalSearchResponse  
    *response, NSError *error)  
{  
    if (response.mapItems.count == 0)  
        NSLog(@"No Matches");  
    else  
        for (MKMapItem *item in response.mapItems)  
        {  
            NSLog(@"name = %@", item.name);  
            NSLog(@"Phone = %@", item.phoneNumber);  
        }  
}];
```

OpenStreetMap Tile-Overlay

In einigen Fällen möchten Sie möglicherweise nicht die von Apple bereitgestellten Standardkarten verwenden.

Sie können Ihrer `mapView`, die benutzerdefinierte Kacheln enthält, beispielsweise über [OpenStreetMap](#) eine Überlagerung hinzufügen.

Nehmen wir an, `self.mapView` ist Ihre `MKMapView`, die Sie bereits zu Ihrem `ViewController` hinzugefügt `ViewController`.

Zunächst muss Ihr `ViewController` dem Protokoll `MKMapViewDelegate`.

```
class MyViewController: UIViewController, MKMapViewDelegate
```

Dann muss der `ViewController` als Delegierter von `mapView`

```
mapView.delegate = self
```

Als Nächstes konfigurieren Sie die Überlagerung für die Karte. Sie benötigen dazu eine URL-Vorlage. Die URL sollte auf allen Tile-Servern ähnlich sein und auch wenn Sie die Kartendaten offline speichern würden: `http://tile.openstreetmap.org/{z}/{x}/{y}.png`

```
let urlTeplate = "http://tile.openstreetmap.org/{z}/{x}/{y}.png"  
let overlay = MKTileOverlay(urlTemplate: urlTeplate)  
overlay.canReplaceMapContent = true
```

Nachdem Sie das Overlay konfiguriert haben, müssen Sie es zu Ihrer `mapView`.

```
mapView.add(overlay, level: .aboveLabels)
```

Um benutzerdefinierte Karten zu verwenden, wird empfohlen, `.aboveLabels` für `level`. Andernfalls sind die Standardbeschriftungen auf Ihrer benutzerdefinierten Karte sichtbar. Wenn Sie die Standardbeschriftungen `.aboveRoads` möchten, können `.aboveRoads` hier `.aboveRoads` auswählen.

Wenn Sie Ihr Projekt jetzt ausführen würden, würden Sie erkennen, dass Ihre Karte immer noch die Standardzuordnung anzeigen würde:



Das liegt daran, dass wir dem `mapView` noch nicht gesagt `mapView`, wie das Overlay `mapView`. Dies ist der Grund, warum Sie den Delegierten zuvor einstellen mussten. Jetzt können Sie `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer` zu Ihrem **View-Controller** `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer`:

```
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if overlay is MKTileOverlay {
        let renderer = MKTileOverlayRenderer(overlay: overlay)
        return renderer
    } else {
        return MKTileOverlayRenderer()
    }
}
```

Dadurch wird der korrekte `MKOverlayRenderer` an Ihre `mapView`. Wenn Sie Ihr Projekt jetzt ausführen, sollten Sie eine Karte wie diese sehen:



Wenn Sie eine andere Karte anzeigen möchten, müssen Sie lediglich die URL-Vorlage ändern. Es gibt eine [Liste von Tile-Servern](#) im OSM-Wiki.

UserLocation und UserTracking-Beispiel anzeigen

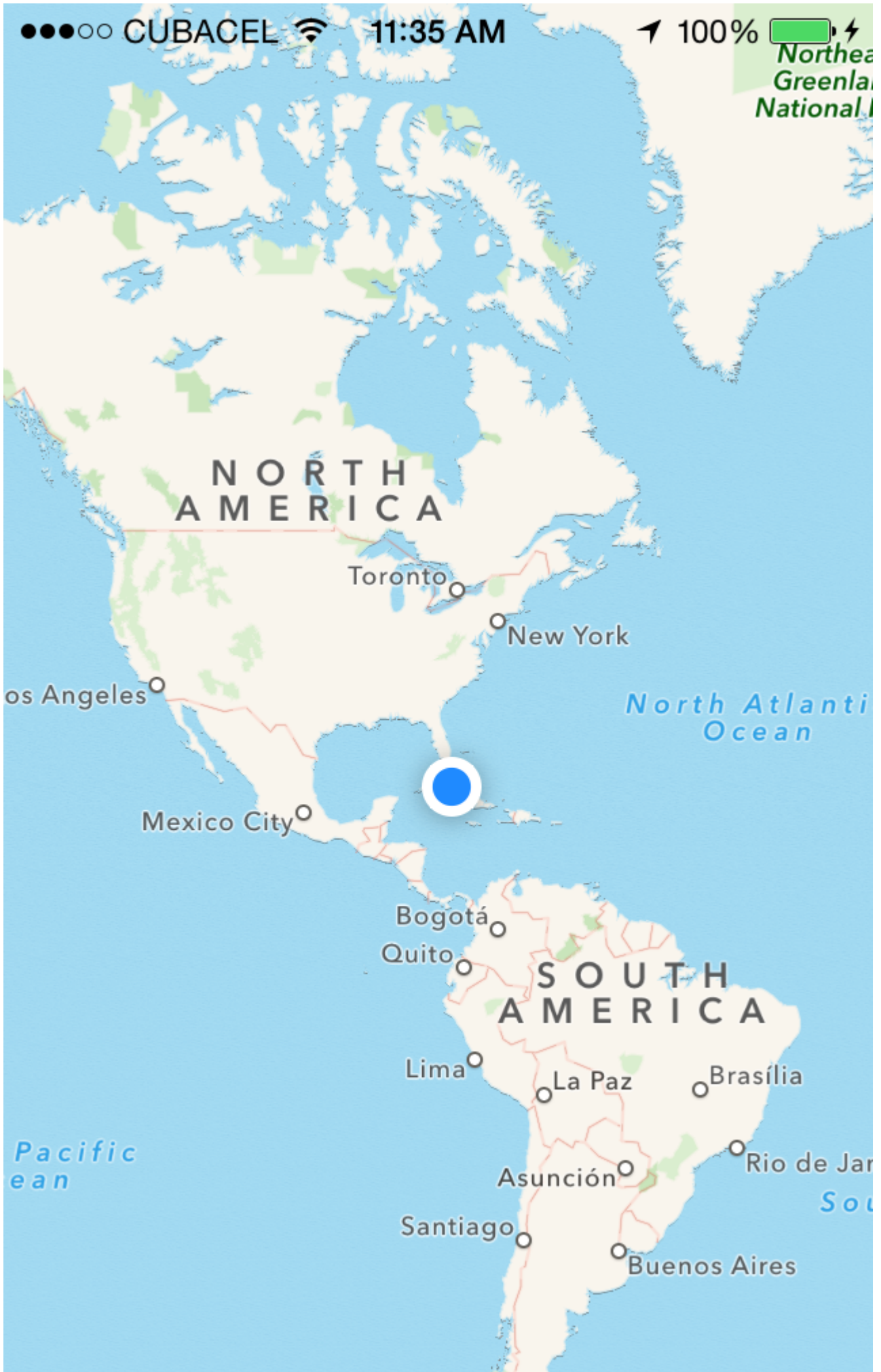
Dadurch wird der Standort des Benutzers auf der Karte angezeigt

Ziel c

```
[self.map setShowsUserLocation:YES];
```

Schnell

```
self.map?.showsUserLocation = true
```



eine Koordinate auf einer `MKMapView` anzeigt, anstatt einen Bereich für die Anzeige `MKMapView`. Diese Funktionalität ist nicht standardmäßig implementiert, so dass Sie erweitern müssen `MKMapView` mit Methoden, die die komplexe Berechnung von einem *Koordinaten* und *Zoom-Ebene* zu einem `tun MKCoordinateRegion`.

```

let MERCATOR_OFFSET = 268435456.0
let MERCATOR_RADIUS = 85445659.44705395
let DEGREES = 180.0

public extension MKMapView {

    //MARK: Map Conversion Methods

    private func longitudeToPixelSpaceX(longitude:Double)->Double{
        return round(MERCATOR_OFFSET + MERCATOR_RADIUS * longitude * M_PI / DEGREES)
    }

    private func latitudeToPixelSpaceY(latitude:Double)->Double{
        return round(MERCATOR_OFFSET - MERCATOR_RADIUS * log((1 + sin(latitude * M_PI /
DEGREES)) / (1 - sin(latitude * M_PI / DEGREES))) / 2.0)
    }

    private func pixelSpaceXToLongitude(pixelX:Double)->Double{
        return ((round(pixelX) - MERCATOR_OFFSET) / MERCATOR_RADIUS) * DEGREES / M_PI
    }

    private func pixelSpaceYToLatitude(pixelY:Double)->Double{
        return (M_PI / 2.0 - 2.0 * atan(exp((round(pixelY) - MERCATOR_OFFSET) /
MERCATOR_RADIUS))) * DEGREES / M_PI
    }

    private func coordinateSpanWithCenterCoordinate(centerCoordinate:CLLocationCoordinate2D,
zoomLevel:Double)->MKCoordinateSpan{
        // convert center coordinate to pixel space
        let centerPixelX = longitudeToPixelSpaceX(longitude: centerCoordinate.longitude)
        let centerPixelY = latitudeToPixelSpaceY(latitude: centerCoordinate.latitude)
        print(centerCoordinate)
        // determine the scale value from the zoom level
        let zoomExponent:Double = 20.0 - zoomLevel
        let zoomScale:Double = pow(2.0, zoomExponent)
        // scale the map's size in pixel space
        let mapSizeInPixels = self.bounds.size
        let scaledMapWidth = Double(mapSizeInPixels.width) * zoomScale
        let scaledMapHeight = Double(mapSizeInPixels.height) * zoomScale
        // figure out the position of the top-left pixel
        let topLeftPixelX = centerPixelX - (scaledMapWidth / 2.0)
        let topLeftPixelY = centerPixelY - (scaledMapHeight / 2.0)
        // find delta between left and right longitudes
        let minLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX)
        let maxLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX + scaledMapWidth)
        let longitudeDelta = maxLng - minLng
        let minLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY)
        let maxLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY + scaledMapHeight)
        let latitudeDelta = -1.0 * (maxLat - minLat)
        return MKCoordinateSpan(latitudeDelta: latitudeDelta, longitudeDelta: longitudeDelta)
    }

    /**
     Sets the center of the `MKMapView` to a `CLLocationCoordinate2D` with a custom zoom-
     level. There is no need to set a region manually. :-)
    */
}

```

```

- author: Mylene Bayan (on GitHub)
*/
public func setCenter(_ coordinate:CLLocationCoordinate2D, zoomLevel:Double,
animated:Bool){
    // clamp large numbers to 28
    var zoomLevel = zoomLevel
    zoomLevel = min(zoomLevel, 28)
    // use the zoom level to compute the region
    print(coordinate)
    let span = self.coordinateSpanWithCenterCoordinate(centerCoordinate: coordinate,
zoomLevel: zoomLevel)
    let region = MKCoordinateRegionMake(coordinate, span)
    if region.center.longitude == -180.00000000{
        print("Invalid Region")
    }
    else{
        self.setRegion(region, animated: animated)
    }
}
}
}

```

(Die ursprüngliche Swift 2-Version von [Mylene Bayan](#) ist auf [GitHub](#) zu finden.)

Nachdem Sie diese `extension` implementiert haben, können Sie die Mittelpunktskoordinate wie folgt festlegen:

```

let centerCoordinate = CLLocationCoordinate2DMake(48.136315, 11.5752901) //latitude, longitude
mapView?.setCenter(centerCoordinate, zoomLevel: 15, animated: true)

```

`zoomLevel` ist ein `Double` Wert, normalerweise zwischen 0 und 21 (was eine sehr hohe `zoomLevel` ist), aber Werte bis 28 sind zulässig.

Mit Anmerkungen arbeiten

Erhalten Sie alle Anmerkungen

```

//following method returns all annotations object added on map
NSArray *allAnnotations = mapView.annotations;

```

Annotationsansicht abrufen

```

for (id<MKAnnotation> annotation in mapView.annotations)
{
    MKAnnotationView* annotationView = [mapView viewForAnnotation:annotation];
    if (annotationView)
    {
        // Do something with annotation view
        // for e.g change image of annotation view
        annotationView.image = [UIImage imageNamed:@"SelectedPin.png"];
    }
}

```

Alle Anmerkungen entfernen

```
[mapView removeAnnotations:mapView.annotations]
```

Einzelne Anmerkung entfernen

```
//getting all Annotation
NSArray *allAnnotations = self.myMapView.annotations;

if (allAnnotations.count > 0)
{
    //getting first annoation
    id <MKAnnotation> annotation=[allAnnotations firstObject];

    //removing annotation
    [mapView removeAnnotation:annotation];
}
}
```

Passen Sie die Sichtbarkeit der Kartenansicht an, um alle Anmerkungen anzuzeigen

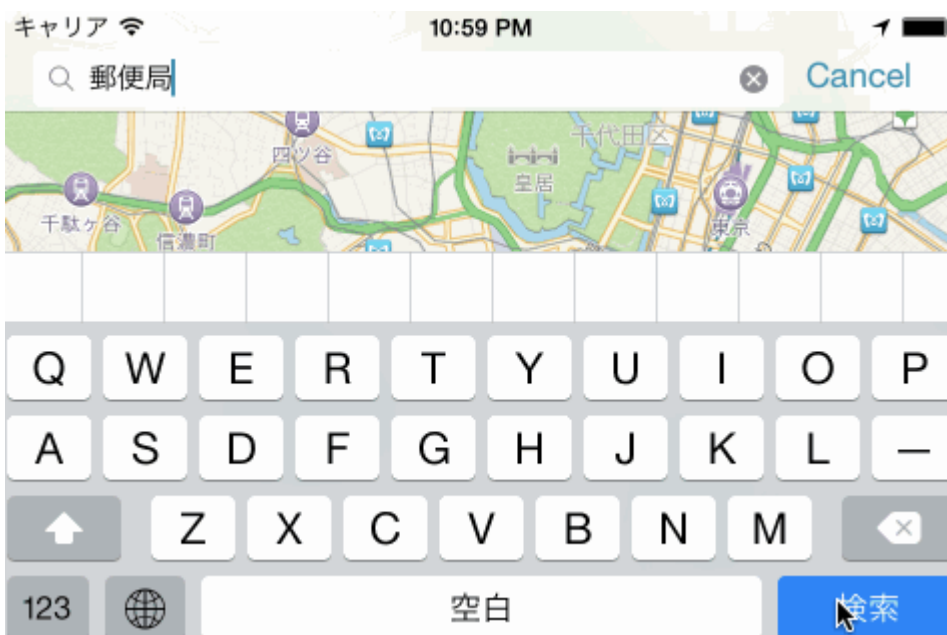
Schnell:

```
mapView.showAnnotations(mapView.annotations, animated: true)
```

Ziel c:

```
[mapView showAnnotations:mapView.annotations animated:YES];
```

Demo:



MKMapView online lesen: <https://riptutorial.com/de/ios/topic/915/mkmapview>

Kapitel 100: ModalPresentationStyles

Einführung

Modale Präsentationsstile werden verwendet, wenn Sie von einem View-Controller zu einem anderen wechseln. Es gibt zwei Möglichkeiten, diese Anpassung vorzunehmen. Eine ist durch Code und eine andere durch Interface Builder (unter Verwendung von Segmenten). Dieser Effekt wird erreicht, indem die Variable `modalPresentationStyle` auf eine Instanz der `UIModalPresentationStyle` . `modalPresentationStyle` Eigenschaft `modalPresentationStyle` ist eine Klassenvariable von `UIViewController` und wird verwendet, um anzugeben, wie ein `ViewController` auf dem Bildschirm dargestellt wird.

Bemerkungen

Denken Sie immer an die folgende Erwähnung von Apple.

In einer horizontal kompakten Umgebung werden Controller für modale Ansichten immer im Vollbildmodus angezeigt. In einer horizontal regulären Umgebung gibt es verschiedene Darstellungsoptionen.

Examples

Erkundung von ModalPresentationStyle mit dem Interface Builder

Dies ist eine sehr einfache App, die verschiedene `ModalpresentationStyle` in iOS veranschaulicht. Gemäß der [hier](#) gefundenen Dokumentation gibt es für `UIModalPresentationStyle` 9 verschiedene Werte, die wie folgt lauten:

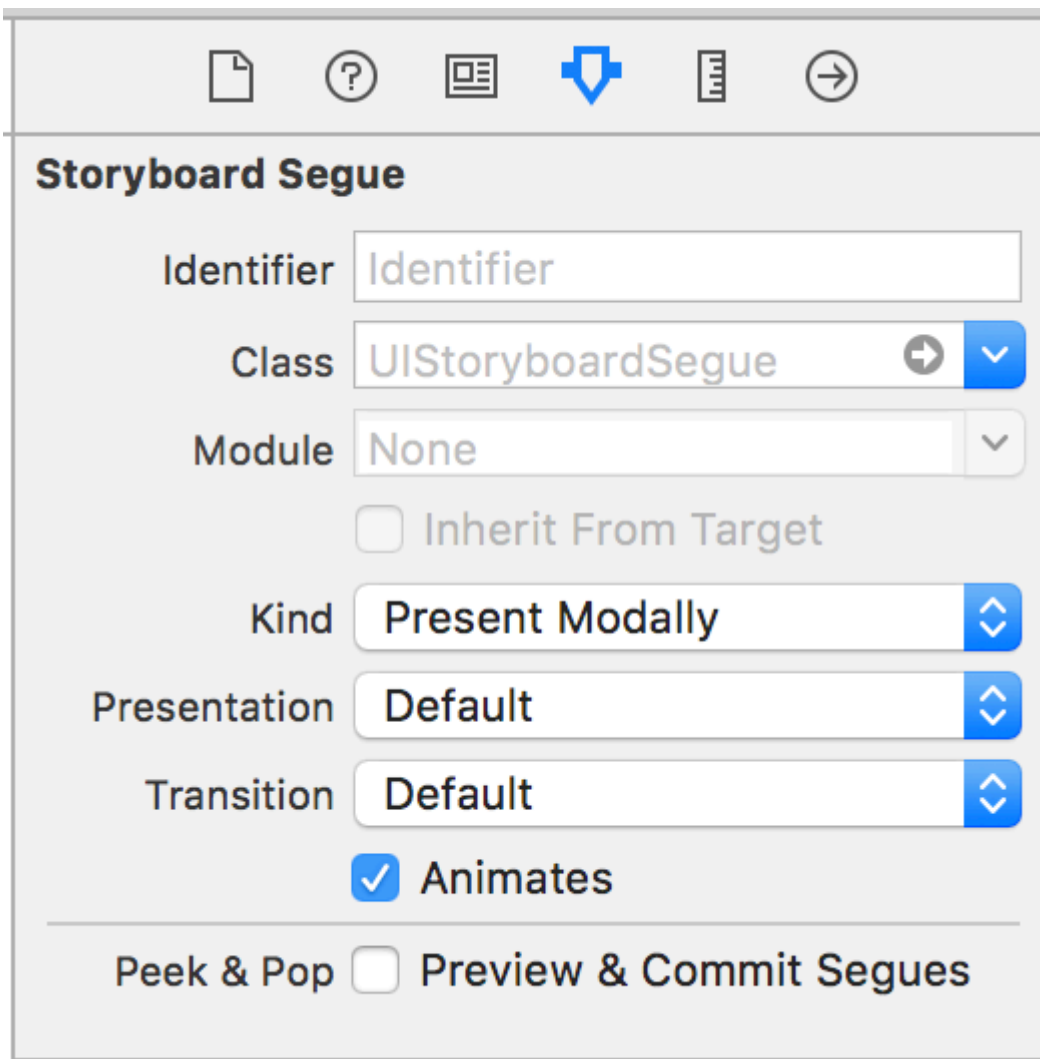
1. `fullScreen`
2. `pageSheet`
3. `formSheet`
4. `currentContext`
5. `custom`
6. `overFullScreen`
7. `overCurrentContext`
8. `popover`
9. `none`

Um ein Projekt `ViewControllers` , erstellen Sie einfach ein normales iOS-Projekt und fügen Sie 2 `ViewControllers` . `UIButton` Sie einen `UIButton` in `UIButton` ursprünglichen `ViewController` und verbinden Sie `ViewController` über einen `Target -> Action` Mechanismus mit dem 2. `ViewController` . Um beide `ViewControllers` zu unterscheiden, legen Sie die Hintergrundfarbe von `UIView` in `ViewController` einer anderen Farbe fest. Wenn alles gut geht, sollte Ihr Interface Builder etwas aussehen,



Button

ausführen (Weitere Informationen zu den Gründen für das iPad finden Sie im Abschnitt "Bemerkungen"). Wenn Sie mit der Einrichtung Ihres Projekts fertig sind, wählen Sie das Segment aus und gehen Sie zum `attributes inspector`. Sie sollten so etwas sehen können,



`Present Modally` Sie die `kind`-Eigenschaft auf `Present Modally`.


In diesem Beispiel werden wir nicht alle Effekte sehen, da einige von ihnen wenig Code benötigen.

Beginnen wir mit `fullscreen`. Dieser Effekt ist standardmäßig ausgewählt, wenn Sie die `Present Modally` in der Registerkarte `Kind` auswählen. Beim Erstellen und Ausführen belegt der 2. `ViewController` den gesamten Bildschirm `ViewController` iPad.



auswählen. Bei dieser Option ähnelt der 2. `ViewController` im Hochformat des Vollbildmodus, im `ViewController` ist der 2. `ViewController` jedoch sehr schmal. Inhalte, die nicht von 2nd `ViewController` werden, werden ebenfalls abgeblendet.



Carrier 

6





in der Mitte des Geräts platziert und die Größe ist kleiner als die des Geräts. Wenn sich das Gerät im `ViewController` befindet und die Tastatur sichtbar ist, wird die `ViewController` nach oben `ViewController` , um den `ViewController` .




Present as Popover in der Registerkarte Kind Present as Popover . Der 2. ViewController wird als kleines Popover dargestellt (Größe kann eingestellt werden). Der Hintergrundinhalt ist abgeblendet. Jede Berührung außerhalb des Popovers würde den Popover verwerfen. Ihr Attributes Inspector sollte in etwa so aussehen:


Storyboard Segue

Identifier


Class  


Module 

Inherit From Target

Kind 

Directions Up Down
 Left Right

Anchor 

Passthrough 

Animates

Peek & Pop Preview & Commit Segues

Anchor ist das Oberflächenelement, auf das der Popover-Pfeil zeigen soll. Directions sind die Richtungen, in die Ihr Anker Anchor zeigen darf.



Button

<https://riptutorial.com/de/ios/topic/10122/modelpresentationstyles>

Kapitel 101: Momentaufnahme von UIView

Examples

Schnappschuss abrufen

```
- (UIImage *)getSnapshot
{
    UIScreen *screen = [UIScreen mainScreen];
    CGRect bounds = [self.view bounds];
    UIGraphicsBeginImageContextWithOptions(bounds.size, false, screen.scale);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetInterpolationQuality(context, kCGInterpolationHigh);
    [self.view drawViewHierarchyInRect:bounds afterScreenUpdates:YES];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}
```

Schnell

```
var screenshot: UIImage
{
    UIGraphicsBeginImageContext(self.bounds.size);
    let context = UIGraphicsGetCurrentContext();
    self.layer.render(in: context)
    let screenShot = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return screenShot
}
```

Schnappschuss mit Unteransicht mit anderem Markup und Text

- Unterstützt Portrait und Querformat für beide Bildtypen
- Zeichnen und andere Unteransichten können in meinem Fall zusammengefügt werden

```
{
    CGSize fullSize = getImageForEdit.size;
    CGSize sizeInView = AVMakeRectWithAspectRatioInsideRect(imgViewFake.image.size,
imgViewFake.bounds).size;
    CGFloat orgScale = orgScale = fullSize.width/sizeInView.width;
    CGSize newSize = CGSizeMake(orgScale * img.image.size.width, orgScale *
img.image.size.height);
    if(newSize.width <= fullSize.width && newSize.height <= fullSize.height){
        newSize = fullSize;
    }
    CGRect offsetRect;
    if (getImageForEdit.size.height > getImageForEdit.size.width){
        CGFloat scale = newSize.height/fullSize.height;
        CGFloat offset = (newSize.width - fullSize.width*scale)/2;
        offsetRect = CGRectMake(offset, 0, newSize.width-offset*2, newSize.height);
    }
    else{
```

```
CGFloat scale = newSize.width/fullSize.width;
CGFloat offset = (newSize.height - fullSize.height*scale)/2;
offsetRect = CGRectMake(0, offset, newSize.width, newSize.height-offset*2);
}
 UIGraphicsBeginImageContextWithOptions(newSize, NO, getImageForEdit.scale);
 [getImageForEdit drawAtPoint:offsetRect.origin];
 //      [img.image drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
CGFloat oldScale = img.contentScaleFactor;
img.contentScaleFactor = getImageForEdit.scale;
 [img drawViewHierarchyInRect:CGRectMake(0, 0, newSize.width, newSize.height)
afterScreenUpdates:YES];
img.contentScaleFactor = oldScale;
UIImage *combImage = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
imageData = UIImageJPEGRepresentation(combImage, 1);
}
```

Momentaufnahme von UIView online lesen:

<https://riptutorial.com/de/ios/topic/4622/momentaufnahme-von-uiview>

Kapitel 102: MPMediaPickerDelegate

Bemerkungen

Weitere Informationen zum Datenschutz finden Sie in der [Apple-Dokumentation](#) .

Stellen Sie sicher, dass die Musik-App auf Ihrem iPhone verfügbar ist. Im Simulator funktioniert es nicht.

Examples

Laden Sie Musik mit MPMediaPickerControllerDelegate und spielen Sie sie mit AVAudioPlayer ab

Gehen Sie die Schritte durch:

- Fügen Sie Ihrer Info.plist "NSAppleMusicUsageDescription" für die Datenschutzbehörde hinzu.
- Stellen Sie sicher, dass Ihre Musik auf Ihrem iPhone verfügbar ist. Im Simulator funktioniert es nicht.

iOS 10.0.1

```
import UIKit
import AVFoundation
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {

    var avMusicPlayer: AVAudioPlayer!
    var mpMediaPicker: MPMediaPickerController!
    var mediaItems = [MPMediaItem]()
    let currentIndex = 0

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool){
        //What to do?
    }

    func mediaPicker(_ mediaPicker: MPMediaPickerController, didPickMediaItems
mediaItemCollection: MPMediaItemCollection) {
        mediaItems = mediaItemCollection.items
        updatePlayer()
        self.dismiss(animated: true, completion: nil)
    }

    func updatePlayer(){
        let item = mediaItems[currentIndex]
        // DO-TRY-CATCH try to setup AVAudioPlayer with the path, if successful, sets up the
```

```

AVMusicPlayer, and song values.
    if let path: NSURL = item.assetURL as NSURL? {
        do
        {
            avMusicPlayer = try AVAudioPlayer(contentsOf: path as URL)
            avMusicPlayer.enableRate = true
            avMusicPlayer.rate = 1.0
            avMusicPlayer.numberOfLoops = 0
            avMusicPlayer.currentTime = 0
        }
        catch
        {
            avMusicPlayer = nil
        }
    }
}

@IBAction func Play(_ sender: AnyObject) {
    //AVMusicPlayer.deviceCurrentTime
    avMusicPlayer.play()
}

@IBAction func Stop(_ sender: AnyObject) {
    avMusicPlayer.stop()
}

@IBAction func picker(_ sender: AnyObject) {
    mpMediapicker = MPMediaPickerController.self(mediaTypes:MPMediaType.music)
    mpMediapicker.allowsPickingMultipleItems = false
    mpMediapicker.delegate = self
    self.present(mpMediapicker, animated: true, completion: nil)
}
}

```

MPMediaPickerDelegate online lesen:

<https://riptutorial.com/de/ios/topic/7299/mpmediapickerdelegate>

Kapitel 103: MPVolumeView

Einführung

Bei der MPVolumeView-Klasse handelt es sich um eine Volume-Ansicht, die dem Benutzer einen Schieberegler zum Einstellen der Lautstärke des System-Audioausgangs und eine Schaltfläche zum Auswählen der Audioausgaberroute bietet.

Bemerkungen

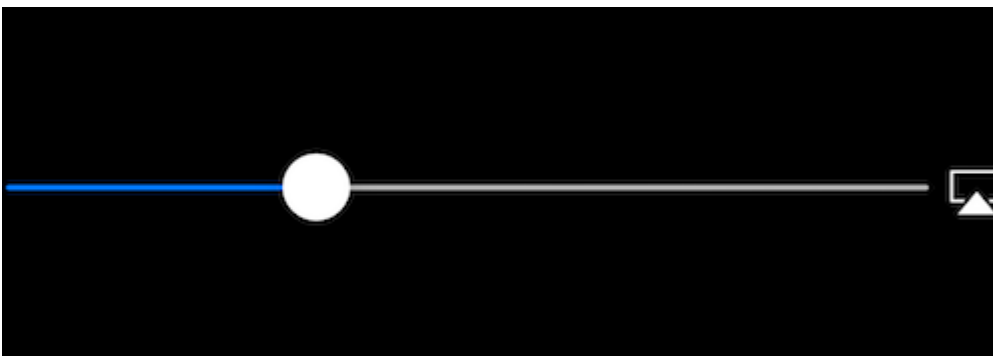
MPVolumeView wird nur beim Erstellen und Ausführen auf einem tatsächlichen iOS-Gerät angezeigt und funktioniert nicht in einem Simulator.

Examples

MPVolumeView hinzufügen

```
// Add MPVolumeView in a holder view
let mpVolumeHolderView = UIView(frame: CGRect(x: 0, y: view.bounds.midY, width:
view.bounds.width, height: view.bounds.height))
// Set the holder view's background color to transparent
mpVolumeHolderView.backgroundColor = .clear
let mpVolume = MPVolumeView(frame: mpVolumeHolderView.bounds)
mpVolume.showsRouteButton = true
mpVolumeHolderView.addSubview(mpVolume)
view.addSubview(mpVolumeHolderView)
// the volume view is white, set the parent background to black to show it better in this
example
view.backgroundColor = .black
```

!!! Ein sehr wichtiger Hinweis ist, dass der MPVolumeView nur auf einem tatsächlichen Gerät und nicht auf einem Simulator funktioniert.



MPVolumeView online lesen: <https://riptutorial.com/de/ios/topic/9038/mpvolumeview>

Kapitel 104: Multicast-Delegierte

Einführung

Muster zum Hinzufügen von Multicasting-Funktionen zu vorhandenen iOS-Steuerelementen. Durch das Hinzufügen von Multicasting wird die Übersichtlichkeit und Wiederverwendung von Code verbessert.

Examples

Multicast-Delegaten für alle Steuerelemente

Leiten Sie Nachrichten von einem Objekt an ein anderes durch Delegierte weiter, und senden Sie diese Nachrichten an mehrere Beobachter.

Schritt 1: - Erstellen Sie die `NSObject` Klasse von `RRMulticastDelegate`

Schritt 2: - Der folgende Code wird in der Datei `RRMulticastDelegate.h` implementiert

```
#import <Foundation/Foundation.h>

@interface RRMulticastDelegate : NSObject

{
    //Handle multiple observers of delegate
    NSMutableArray* _delegates;
}

// Delegate method implementation to the list of observers
- (void)addDelegate:(id)delegate;
- (void)removeDelegate:(id)delegate;

// Get multiple delegates
-(NSArray *)delegatesObjects;

@end
```

Schritt 3: - Der folgende Code wird in der Datei `RRMulticastDelegate.m` implementiert

```
#import "RRMulticastDelegate.h"

@implementation RRMulticastDelegate

- (id)init
{
    if (self = [super init])
    {
        _delegates = [NSMutableArray array];
    }
    return self;
}
```

```

- (NSArray *)delegatesObjects
{
    return _delegates;
}

- (void)removeDelegate:(id)delegate
{
    if ([_delegates containsObject:delegate])
        [_delegates removeObject:delegate];
}

- (void)addDelegate:(id)delegate
{
    if (![_delegates containsObject:delegate])
        [_delegates addObject:delegate];
}

- (BOOL)respondsToSelector:(SEL)aSelector
{
    if ([super respondsToSelector:aSelector])
        return YES;

    // if any of the delegates respond to this selector, return YES
    for(id delegate in _delegates)
    {
        if (!delegate)
            continue;

        if ([delegate respondsToSelector:aSelector])
        {
            return YES;
        }
    }
    return NO;
}

- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector
{
    // can this class create the signature?
    NSMethodSignature* signature = [super methodSignatureForSelector:aSelector];

    // if not, try our delegates
    if (!signature)
    {
        for(id delegate in _delegates)
        {
            if (!delegate)
                continue;

            if ([delegate respondsToSelector:aSelector])
            {
                return [delegate methodSignatureForSelector:aSelector];
            }
        }
    }
    return signature;
}

- (void)forwardInvocation:(NSInvocation *)anInvocation
{

```

```

// forward the invocation to every delegate
for(id delegate in _delegates)
{
    if (!delegate)
        continue;

    if ([delegate respondsToSelector:[anInvocation selector]])
    {
        [anInvocation invokeWithTarget:delegate];
    }
}
}

@end

```

Schritt 4: - Erstellen Sie die NSObject **Kategorieklasse** von RRProperty

Schritt 5: - Folgende NSObject+RRProperty.h Datei NSObject+RRProperty.h

```

#import <Foundation/Foundation.h>

#import "RRMulticastDelegate.h"

@interface NSObject (RRProperty)<UITextFieldDelegate,UITableViewDataSource>

-(void)setObject:(id)block forKey:(NSString *)key;
-(id)objectForKey:(NSString *)key;

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate;
- (RRMulticastDelegate *)multicastDatasource;

-(void)addDelegate:(id)delegate;
-(void)addDataSource:(id)datasource;

@end

```

Schritt 6: - Der folgende Code wird in der Datei NSObject+RRProperty.m implementiert

```

#import "NSObject+RRProperty.h"

#import <objc/message.h>
#import <objc/runtime.h>

#pragma GCC diagnostic ignored "-Wprotocol"

static NSString *const MULTICASTDELEGATE = @"MULTICASTDELEGATE";
static NSString *const MULTICASTDATASOURCE = @"MULTICASTDATASOURCE";

@implementation NSObject (RRProperty)

-(void)setObject:(id)block forKey:(NSString *)key
{
    objc_setAssociatedObject(self, (__bridge const void *) (key), block,
OBJC_ASSOCIATION_RETAIN);
}

```

```

-(id)objectForKey:(NSString *)key
{
    return objc_getAssociatedObject(self, (__bridge const void *) (key));
}

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate
{
    id multicastDelegate = [self objectForKey:MULTICASTDELEGATE];
    if (multicastDelegate == nil) {
        multicastDelegate = [[RRMulticastDelegate alloc] init];

        [self setObject:multicastDelegate forKey:MULTICASTDELEGATE];
    }

    return multicastDelegate;
}

- (RRMulticastDelegate *)multicastDatasource
{
    id multicastDatasource = [self objectForKey:MULTICASTDATASOURCE];
    if (multicastDatasource == nil) {
        multicastDatasource = [[RRMulticastDelegate alloc] init];

        [self setObject:multicastDatasource forKey:MULTICASTDATASOURCE];
    }

    return multicastDatasource;
}

-(void)addDelegate:(id)delegate
{
    [self.multicastDelegate addDelegate:delegate];

    UITextField *text = (UITextField *) self;
    text.delegate = self.multicastDelegate;
}

-(void)addDataSource:(id)datasource
{
    [self.multicastDatasource addDelegate:datasource];
    UITableView *text = (UITableView *) self;
    text.dataSource = self.multicastDatasource;
}

@end

```

Schließlich können Sie Multicast-Delegaten für alle Steuerelemente verwenden ...

Für ex ...

Importieren Sie Ihre Viewcontroller-Klasse in die Datei `NSObject+RRProperty.h`, um auf ihre Methoden zuzugreifen, **um den Multicast-Delegaten / die Datenquelle festzulegen**.

```

UITextView *txtView = [[UITextView alloc] initWithFrame:txtframe];
[txtView addDelegate:self];

UITableView *tblView = [[UITableView alloc] initWithFrame:tblframe];

```

```
[tblView addDelegate:self];  
[tblView addDataSource:self];
```

Multicast-Delegierte online lesen: <https://riptutorial.com/de/ios/topic/10081/multicast-delegierte>

Kapitel 105: MVP-Architektur

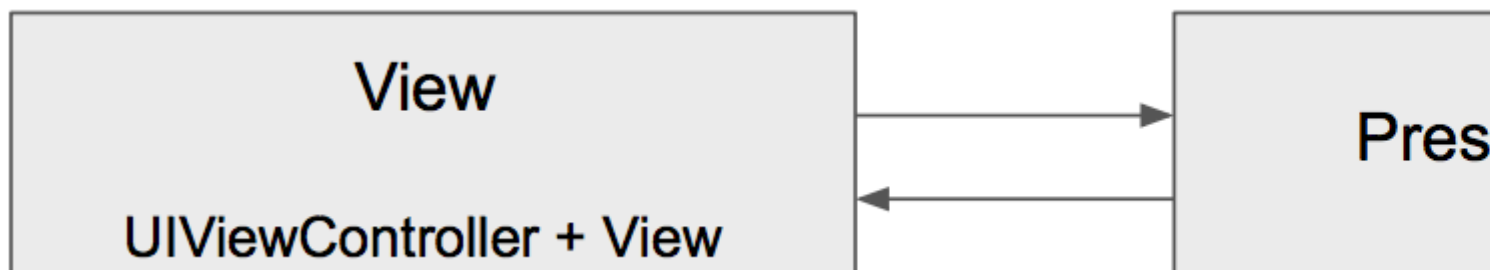
Einführung

MVP ist ein Architekturmuster, eine Ableitung des Modell-Ansicht-Controllers. Es wird durch drei verschiedene Komponenten dargestellt: Modell, Ansicht und Präsentator. Es wurde entwickelt, um automatisierte Unit-Tests zu ermöglichen und die Trennung von Anliegen in der Präsentationslogik zu verbessern.

In Beispielen finden Sie ein einfaches Projekt, das mit dem MVP-Muster erstellt wurde.

Bemerkungen

Komponenten:



- **Modell** ist eine Schnittstelle, die für die Domänendaten verantwortlich ist (die in der GUI angezeigt oder anderweitig bearbeitet werden sollen).
- **Ansicht** ist für die Präsentationsschicht (GUI) verantwortlich
- **Presenter** ist der "Mittelmensch" zwischen Modell und View. Es reagiert auf die Aktionen des Benutzers, die in der Ansicht ausgeführt werden, ruft Daten aus dem Modell ab und formatiert sie für die Anzeige in der Ansicht

Pflichten der Komponente:

Modell	Aussicht	Moderator
Kommuniziert mit der DB-Schicht	Rendert Daten	Führt Abfragen an das Modell aus
Geeignete Ereignisse auslösen	Empfängt Ereignisse	Formatiert Daten von Modell
	Sehr grundlegende Validierungslogik	Sendet formatierte Daten an die Ansicht
		Komplexe Validierungslogik

Unterschiede zwischen MVC und MVP :

- View in MVC ist eng mit dem Controller gekoppelt. Der View-Teil des MVP besteht aus UIViews und UIViewController
- MVP View ist so dumm wie möglich und enthält fast keine Logik (wie in MVVM). MVC View verfügt über einige Geschäftslogik und kann das Modell abfragen
- MVP View behandelt Benutzergesten und delegiert die Interaktion an den Presenter. In MVC verarbeitet der Controller Gesten und Befehle Modell
- MVP-Pattern unterstützt Unit-Tests stark, MVC hat nur eingeschränkte Unterstützung
- MVC Controller hat viele UIKit-Abhängigkeiten, MVP Presenter hat keine

Pros:

- MVP macht UIViewController zu einem Teil der View-Komponente, dumm, passiv und ... weniger massiv;]
- Der größte Teil der Geschäftslogik ist aufgrund der dummen Ansichten verkapselt. Dies ergibt eine hervorragende Testbarkeit. Zum Testen des Domänenteils können Mock-Objekte eingeführt werden.
- Getrennte Einheiten sind leichter im Kopf zu halten, die Verantwortlichkeiten sind klar aufgeteilt.

Cons

- Sie werden mehr Code schreiben.
- Barriere für unerfahrene Entwickler oder für diejenigen, die noch nicht mit dem Muster arbeiten.

Examples

Hundewechsel

```
import Foundation

enum Breed: String {
    case bulldog = "Bulldog"
    case doberman = "Doberman"
    case labrador = "Labrador"
}

struct Dog {
    let name: String
    let breed: String
    let age: Int
}
```

DoggyView.swift

```
import Foundation

protocol DoggyView: NSObjectProtocol {
```

```

func startLoading()
func finishLoading()
func setDoggies(_ doggies: [DoggyViewData])
func setEmpty()
}

```

DoggyService.swift

```

import Foundation

typealias Result = ([Dog]) -> Void

class DoggyService {

    func deliverDoggies(_ result: @escaping Result) {

        let firstDoggy = Dog(name: "Alfred", breed: Breed.labrador.rawValue, age: 1)
        let secondDoggy = Dog(name: "Vinny", breed: Breed.doberman.rawValue, age: 5)
        let thirdDoggy = Dog(name: "Lucky", breed: Breed.labrador.rawValue, age: 3)

        let delay = DispatchTime.now() + Double(Int64(Double(NSEC_PER_SEC)*2)) /
Double(NSEC_PER_SEC)

        DispatchQueue.main.asyncAfter(deadline: delay) {
            result([firstDoggy,
                    secondDoggy,
                    thirdDoggy])
        }
    }
}

```

DoggyPresenter.swift

```

import Foundation

class DoggyPresenter {

    // MARK: - Private
    fileprivate let dogService: DoggyService
    weak fileprivate var dogView: DoggyView?

    init(dogService: DoggyService){
        self.dogService = dogService
    }

    func attachView(_ attach: Bool, view: DoggyView?) {
        if attach {
            dogView = nil
        } else {
            if let view = view { dogView = view }
        }
    }

    func getDogs(){
        self.dogView?.startLoading()

        dogService.deliverDoggies { [weak self] doggies in

```



```

        self?.dogView?.finishLoading()

    if doggies.count == 0 {
        self?.dogView?.setEmpty()
    } else {
        self?.dogView?.setDoggies(doggies.map {
            return DoggyViewData(name: "\($0.name) \($0.breed)",
                age: "\($0.age)")
        })
    }
}

struct DoggyViewData {
    let name: String
    let age: String
}

```

DoggyListViewController.swift

```

import UIKit

class DoggyListViewController: UIViewController, UITableViewDataSource {

    @IBOutlet weak var emptyView: UIView?
    @IBOutlet weak var tableView: UITableView?
    @IBOutlet weak var spinner: UIActivityIndicatorView?

    fileprivate let dogPresenter = DoggyPresenter(dogService: DoggyService())
    fileprivate var dogsToDisplay = [DoggyViewData]()

    override func viewDidLoad() {
        super.viewDidLoad()

        tableView?.dataSource = self
        spinner?.hidesWhenStopped = true
        dogPresenter.attachView(true, view: self)
        dogPresenter.getDogs()
    }

    // MARK: DataSource
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return dogsToDisplay.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell {
        let cell = UITableViewCell(style: .subtitle, reuseIdentifier: "Cell")
        let userViewData = dogsToDisplay[indexPath.row]
        cell.textLabel?.text = userViewData.name
        cell.detailTextLabel?.text = userViewData.age
        return cell
    }
}

extension DoggyListViewController: DoggyView {

    func startLoading() {

```

```
        spinner?.startAnimating()
    }

    func finishLoading() {
        spinner?.stopAnimating()
    }

    func setDoggies(_ doggies: [DoggyViewData]) {
        dogsToDisplay = doggies
        tableView?.isHidden = false
        emptyView?.isHidden = true;
        tableView?.reloadData()
    }

    func setEmpty() {
        tableView?.isHidden = true
        emptyView?.isHidden = false;
    }
}
```

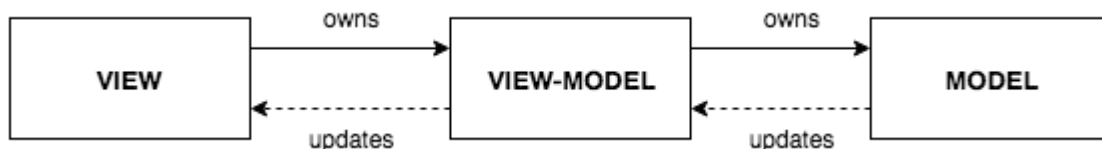
MVP-Architektur online lesen: <https://riptutorial.com/de/ios/topic/9467/mvp-architektur>

Kapitel 106: MVVM

Examples

MVVM ohne reaktive Programmierung

Ich fange mit einer wirklich kurzen Erklärung an, was ist und warum das Model-View-ViewModel (MVVM) -Designmuster in Ihren iOS-Apps verwendet wird. Als iOS zum ersten Mal auf den Markt kam, schlug Apple vor, MVC (Model-View-Controller) als Entwurfsmuster zu verwenden. Sie haben es in all ihren Beispielen gezeigt und alle ersten Entwickler waren glücklich damit, weil sie die Anliegen zwischen Geschäftslogik und Benutzeroberfläche gut voneinander trennten. Als Anwendungen immer umfangreicher und komplexer wurden, erschien ein neues Problem, das als Massive View Controller (MVC) bezeichnet wurde. Da die gesamte Geschäftslogik im ViewController hinzugefügt wurde, wurden sie mit der Zeit zu umfangreich und komplex. Um MVC-Probleme zu vermeiden, wurde ein neues Designmuster in die Welt des iOS eingeführt - das Model-View-ViewModel (MVVM) -Muster.



Das Diagramm oben zeigt, wie MVVM aussieht. Sie haben einen Standard-ViewController + View (in Storyboard, XIB oder Code), der als MVVM-View fungiert (in späterem Text - View verweist auf MVVM-View). Eine Ansicht hat einen Verweis auf ein ViewModel, in dem sich unsere Geschäftslogik befindet. Es ist wichtig zu wissen, dass ViewModel nichts über die Ansicht weiß und nie einen Verweis auf die Ansicht hat. ViewModel hat eine Referenz auf ein Modell. Dies reicht bei einem theoretischen Teil der MVVM aus. Mehr über sie gelesen werden kann [hier](#).

Eines der **Hauptprobleme bei MVVM** ist das Aktualisieren von View über das ViewModel, wenn ViewModel keine Referenzen hat und nicht einmal etwas über die View weiß.

Der Hauptteil dieses Beispiels soll zeigen, wie MVVM (genauer gesagt, wie ViewModel und View gebunden werden) ohne reaktive Programmierung (ReactiveCocoa, ReactiveSwift oder RxSwift) verwendet werden. Nur eine Anmerkung: Wenn Sie die reaktive Programmierung verwenden möchten, ist die Verwendung von MVVM-Bindungen noch einfacher. In diesem Beispiel wird jedoch beschrieben, wie MVVM ohne Reactive-Programmierung verwendet wird.

Lassen Sie uns ein einfaches Beispiel erstellen, um die Verwendung von MVVM zu veranschaulichen.

Unser `MVVMExampleViewController` ist ein einfacher ViewController mit einem Label und einem Button. Wenn die Taste gedrückt wird, sollte der Etikettentext auf "Hallo" gesetzt werden. Da die Entscheidung, was bei Benutzerbenutzerinteraktionen zu tun ist, Teil der Geschäftslogik ist, muss ViewModel entscheiden, was zu tun ist, wenn der Benutzer die Schaltfläche drückt. MVVMs View sollte keine Geschäftslogik ausführen.

```

class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}

```

MVVMExampleViewModel ist ein einfaches ViewModel.

```

class MVVMExampleViewModel {

    func userTriggeredSayHelloButton() {
        // How to update View's label when there is no reference to the View??
    }
}

```

Sie fragen sich vielleicht, wie Sie die ViewModel-Referenz in der Ansicht einstellen. Normalerweise mache ich es, wenn ViewController initialisiert wird oder bevor er angezeigt wird. Für dieses einfache Beispiel würde ich in AppDelegate so etwas AppDelegate :

```

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    if let rootVC = window?.rootViewController as? MVVMExampleViewController {
        let viewModel = MVVMExampleViewModel()
        rootVC.viewModel = viewModel
    }

    return true
}

```

Die eigentliche Frage lautet nun: Wie kann ich View von ViewModel aktualisieren, ohne einen Hinweis auf die View auf das ViewModel zu geben? (Denken Sie daran, dass wir keine der reaktiven Programmier-iOS-Bibliotheken verwenden werden.)

Sie könnten über die Verwendung von KVO nachdenken, aber das würde die Dinge zu kompliziert machen. Einige kluge Leute haben über das Thema nachgedacht und die [Bond-Bibliothek entworfen](#) . Die Bibliothek mag auf den ersten Blick kompliziert und ein wenig schwieriger zu verstehen sein, also nehme ich nur einen kleinen Teil davon und mache unsere MVVM voll funktionsfähig.

Wir stellen die `Dynamic` Klasse vor, die den Kern unseres einfachen, aber voll funktionsfähigen MVVM-Musters bildet.

```

class Dynamic<T> {
    typealias Listener = (T) -> Void
    var listener: Listener?
}

```

```

func bind(_ listener: Listener?) {
    self.listener = listener
}

func bindAndFire(_ listener: Listener?) {
    self.listener = listener
    listener?(value)
}

var value: T {
    didSet {
        listener?(value)
    }
}

init(_ v: T) {
    value = v
}
}

```

Dynamic Klasse verwendet Generics und Closures, um unser ViewModel mit unserem View zu verbinden. Ich werde nicht auf Details zu dieser Klasse eingehen, wir können es in den Kommentaren machen (um dieses Beispiel zu verkürzen). Lassen Sie uns nun unseren `MVVMExampleViewController` und `MVVMExampleViewModel`, um diese Klassen zu verwenden.

Unser aktualisierter `MVVMExampleViewController`

```

class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
        bindViewModel()
    }

    func bindViewModel() {
        if let viewModel = viewModel {
            viewModel.helloText.bind({ (helloText) in
                DispatchQueue.main.async {
                    // When value of the helloText Dynamic variable
                    // is set or changed in the ViewModel, this code will
                    // be executed
                    self.helloLabel.text = helloText
                }
            })
        }
    }

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}

```

`MVVMExampleViewModel` aktualisiert:

```
class MVVMExampleViewModel {  
  
    // we have to initialize the Dynamic var with the  
    // data type we want  
    var helloText = Dynamic("")  
  
    func userTriggeredSayHelloButton() {  
        // Setting the value of the Dynamic variable  
        // will trigger the closure we defined in the View  
        helloText.value = "Hello"  
    }  
}
```

Das ist es. Ihr `ViewModel` **jetzt** `View` aktualisieren, ohne dass es einen Verweis auf die `View` .

Dies ist ein wirklich einfaches Beispiel, aber ich denke, Sie haben eine Vorstellung davon, wie mächtig dies sein kann. Ich werde nicht näher auf die Vorteile der MVVM eingehen, aber wenn Sie von MVCM zu MVVM wechseln, werden Sie nicht mehr zurückkehren. Probieren Sie es einfach aus und überzeugen Sie sich selbst.

MVVM online lesen: <https://riptutorial.com/de/ios/topic/8775/mvvm>

Kapitel 107: Navigationsleiste

Examples

Anpassen der Standardansicht der Navigationsleiste.

```
// Default UINavigationController appearance throughout the app
[[UINavigationController appearance] setTitleTextAttributes:@{NSForegroundColorAttributeName:
[UIColor whiteColor],
                                                            NSFontAttributeName : [UIFont
fontWithName:@"HelveticaNeue-CondensedBold" size:17],
                                                            }
};

[[UINavigationController appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationController appearance] setBarTintColor:[UIColor KNGRed]];
[[UINavigationController appearance] setTranslucent:NO];
[[UINavigationController appearance] setBarStyle:UIBarStyleBlack];
[[UIBarButtonItem appearanceWhenContainedIn: [UISearchBar class], nil] setTintColor:[UIColor
KNGGray]];
```

SWIFT-Beispiel

```
navigationController?.navigationBar.titleTextAttributes = [NSForegroundColorAttributeName:
UIColor.white, NSFontAttributeName:UIFont(name: "HelveticaNeue-CondensedBold", size: 17)!,]
navigationController?.navigationBar.tintColor = .white
navigationController?.navigationBar.barTintColor = .red
navigationController?.navigationBar.isTranslucent = false
navigationController?.navigationBar.barStyle = .black
```

Navigationsleiste online lesen: <https://riptutorial.com/de/ios/topic/7066/navigationsleiste>

Kapitel 108: NSArray

Einführung

Im Folgenden finden Sie einige nützliche Hilfsprogrammfunktionen / -methoden, die wie bei der Array-Erweiterung verwendet werden können, damit der Entwickler bestimmte kritische Operationen an Arrays mit Hilfe von einzeiligem Code durchführen kann.

Bemerkungen

Sobald das aktuelle Dokument genehmigt wird, werden auch andere Array-Anwendungen so viele Verbesserungen erhalten. Dies ist mein erstes Dokument und ich brauche Ihre Unterstützung und Ihre Zustimmung.

Examples

Array in Json-String umwandeln

Rufen Sie diese Funktion mit Parameterargument als Array mit dem Typ 'any' auf. Es wird Ihnen Json String zurückgeben. Ein Json-String wird verwendet, um ein Array in einem Webdienst-Aufruf als Anforderungseingabeparameter in Swift zu senden.

// -----

```
let array = [{"one" : 1}, {"two" : 2}, {"three" : 3}, {"four" : 4}]

let jsonString = convertIntoJSONString(arrayObject: array)
print("jsonString - \(jsonString)")
```

// -----

```
func convertIntoJSONString(arrayObject: [Any]) -> String? {

    do {
        let jsonData: Data = try JSONSerialization.data(withJSONObject: arrayObject,
options: [])
        if let jsonString = NSString(data: jsonData, encoding:
String.Encoding.utf8.rawValue) {
            return jsonString as String
        }
    } catch let error as NSError {
        print("Array convertIntoJSON - \(error.description)")
    }
    return nil
}
```

NSArray online lesen: <https://riptutorial.com/de/ios/topic/9248/nsarray>

Kapitel 109: NSAttributedString

Bemerkungen

[Festlegen der Schriftfarbe mit NSAttributedString](#)

Examples

Erstellen einer Zeichenfolge mit benutzerdefiniertem Kerning (Buchstabenabstand)

`NSAttributedString` (und seinem veränderlichen Geschwister `NSMutableAttributedString`) können Sie Strings erstellen, die in ihrem Erscheinungsbild für den Benutzer komplex sind.

Eine gebräuchliche Anwendung besteht darin, eine Zeichenfolge anzuzeigen und benutzerdefinierte Kern- / Buchstabenabstände hinzuzufügen.

Dies würde folgendermaßen erreicht werden (wobei `label` ein `UILabel`), was für das Wort "Kerning" ein anderes Kerning ergibt.

Schnell

```
var attributedString = NSMutableAttributedString("Apply kerning")
attributedString.addAttribute(attribute: NSKernAttributeName, value: 5, range: NSRange(6, 7))
label.attributedString = attributedString
```

Ziel c

```
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply kerning"];
[attributedString addAttribute:NSKernAttributeName value:@5 range:NSMakeRange(6, 7)];
[label setAttributedString:attributedString];
```

Erstellen Sie eine Zeichenfolge mit durchgestrichenem Text

Ziel c

```
NSMutableAttributedString *attributeString = [[NSMutableAttributedString alloc]
initWithString:@"Your String here"];
[attributeString addAttribute:NSStrikethroughStyleAttributeName
value:@2
range:NSMakeRange(0, [attributeString length])];
```

Schnell

```
let attributeString: NSMutableAttributedString = NSMutableAttributedString(string: "Your
```

```
String here")
attributeString.addAttribute(NSStrikethroughStyleAttributeName, value: 2, range:
NSMakeRange(0, attributeString.length))
```

Dann können Sie dies zu Ihrem UILabel hinzufügen:

```
yourLabel.attributedText = attributeString;
```

Attributierte Zeichenfolgen und fetten Text in Swift anhängen

```
let someValue : String = "Something the user entered"
let text = NSMutableAttributedString(string: "The value is: ")
text.appendAttributedString(NSAttributedString(string: someValue, attributes:
[NSFontAttributeName:UIFont.boldSystemFontOfSize(UIFont.systemFontSize())]))
```

Das Ergebnis sieht so aus:

Der Wert ist: **Etwas, das der Benutzer eingegeben hat**

Ändern Sie die Farbe eines Wortes oder einer Zeichenfolge

Ziel c

```
UIColor *color = [UIColor redColor];
NSString *textToFind = @"redword";

NSMutableAttributedString *attrsString = [[NSMutableAttributedString alloc]
initWithAttributedString:yourLabel.attributedText];

// search for word occurrence
NSRange range = [yourLabel.text rangeOfString:textToFind];
if (range.location != NSNotFound) {
    [attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
}

// set attributed text
yourLabel.attributedText = attrsString;
```

Schnell

```
let color = UIColor.red;
let textToFind = "redword"

let attrsString = NSMutableAttributedString(string:yourlabel.text!);

// search for word occurrence
let range = (yourlabel.text! as NSString).range(of: textToFind)
if (range.length > 0) {
    attrsString.addAttribute(NSForegroundColorAttributeName,value:color,range:range)
}

// set attributed text
yourlabel.attributedText = attrsString
```

Hinweis :

Das wichtigste hier ist die Verwendung eines `NSMutableAttributedString` und des Selektors `addAttribute:value:range` mit dem Attribut `NSForegroundColorAttributeName` , um die Farbe eines `NSForegroundColorAttributeName` zu ändern:

```
NSMutableAttributedString *attrsString = [[NSMutableAttributedString alloc]
initWithAttributedString:label.attributedString];
[attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
```

Sie können den Bereich auch auf andere Weise abrufen, z. B. `NSRegularExpression`.

Alle Attribute entfernen

Ziel c

```
NSMutableAttributedString *mutAttString = @"string goes here";
NSRange range = NSMakeRange(0, mutAttString.length);
[mutAttString setAttributes:@{ } range:originalRange];
```

`setAttributes` Apple-Dokumentation verwenden wir `setAttributes` und nicht `addAttribute` .

Schnell

```
mutAttString.setAttributes([:], range: NSRange(0..
```

NSAttributedString online lesen: <https://riptutorial.com/de/ios/topic/979/nsattributedString>

Kapitel 110: NSBundle

Examples

Holen Sie sich das Hauptpaket

1. Verweis auf das Hauptpaket mit Cocoa erhalten.

Rufen Sie die **mainBundle**- Klassenmethode der **NSBundle**- Klasse auf, um das **Hauptpaket** in der Cocoa-Anwendung **abzurufen** .

```
NSBundle *mainBundle;
// Get the main bundle for the app;
mainBundle = [NSBundle mainBundle];
```

2. Abrufen eines Verweises auf das Hauptpaket mit Core Foundation.

Verwenden Sie die **CFBundleGetMainBundle**- Funktion, um das **Hauptpaket** für Ihre C-basierte Anwendung **abzurufen**.

```
CFBundleRef mainBundle;
// Get the main bundle for the app
mainBundle = CFBundleGetMainBundle();
```

Bundle per Pfad erhalten

1. Suchen eines Kakaobündels anhand seines Pfads

Rufen Sie die **BundleWithPath**: Klassenmethode von **NSBundle** auf, um das Bundle mit Cocoa an einem bestimmten Pfad zu **erhalten**

```
NSBundle *myBundle;
// obtain a reference to a loadable bundle
myBundle = [NSBundle bundleWithPath:@"~/Library/MyBundle.bundle"];
```

2. Suchen eines Cocoa Foundation-Pakets mithilfe seines Pfads

Rufen Sie die Funktion **CFBundleCreate** auf und müssen **Sie den Typ CFURLRef** verwenden, um das Bundle mit Core Foundation an einem bestimmten Pfad zu erhalten.

```
CFURLRef bundleURL;
CFBundleRef mainBundle;
// Make a CFURLRef from the CFString representation of the bundle's path.
bundleURL = CFURLCreateWithFileSystemPath(kCFAllocatorDefault,
CFSTR("~/Library/MyBundle.bundle"), kCFURLPOSIXPathStyle, true);
// Make a bundle instance using the URLRef.
mainBundle = CFBundleCreate(kCFAllocatorDefault, bundleURL);
```

```
// You can release the URL now.  
CFRelease(bundleURL);  
// Use the bundle ...  
// Release the bundle when done.  
CFRelease(myBundle);
```

NSBundle online lesen: <https://riptutorial.com/de/ios/topic/5862/nsbundle>

Kapitel 111: NSData

Bemerkungen

Nützliche Ressourcen

[Apple-Dokumentation \(NSData\)](#)

[NSData.dataWithContentsOfFile \(\)](#)

[NSData.bytes](#)

Examples

NSData-Objekte erstellen

Datei verwenden

Schnell

```
let data = NSData(contentsOfFile: filePath) //assuming filePath is a valid path
```

Ziel c

```
NSData *data = [NSData dataWithContentsOfFile:filePath]; //assuming filePath is a valid path
```

Verwenden eines String-Objekts

Schnell

```
let data = (string as NSString).dataUsingEncoding(NSUTF8StringEncoding) //assuming string is a String object
```

Ziel c

```
NSData *data = [string dataUsingEncoding:NSUTF8StringEncoding]; //assuming string is a String object
```

Konvertierung von NSData in andere Typen

Zu String

Schnell

```
let string = String(NSString(data: data, encoding: NSUTF8StringEncoding)) //assuming data is a valid NSData object
```

Ziel c

```
NSString *string = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];  
//assuming data is a valid NSData object  
[string release];
```

Zum Array

Schnell

```
let array = data.bytes as! NSMutableArray //assuming data is a valid NSData object
```

Ziel c

```
NSMutableArray *array = (NSMutableArray *)[data bytes]; //assuming data is a valid NSData object
```

In Byte-Array

Schnell

```
let byteArray = data.bytes as! UInt8 //assuming data is a valid NSData object
```

Ziel c

```
UInt8 *byteArray = (UInt8 *)data.bytes; //assuming data is a valid NSData object
```

Konvertierung von NSData in HEX-String

NSData

kann als hexadezimaler String dargestellt werden, ähnlich wie bei der `description`.

Schnell

```
extension NSData {  
  
    func hexString() -> String {  
        return UnsafeBufferPointer<UInt8>(start: UnsafePointer<UInt8>(bytes), count: length)  
            .reduce("") { $0 + String(format: "%02x", $1) }  
    }  
  
}
```

Ziel c

```
@implementation NSData (HexRepresentation)  
  
- (NSString *)hexString {  
    const unsigned char *bytes = (const unsigned char *)self.bytes;  
    NSMutableString *hex = [NSMutableString new];  
    for (NSInteger i = 0; i < self.length; i++) {  
        [hex appendFormat:@"%02x", bytes[i]];  
    }  
    return [hex copy];  
}  
  
@end
```

NSData online lesen: <https://riptutorial.com/de/ios/topic/5084/nsdata>

Kapitel 112: NSDate

Syntax

- NSDate () // NSDate-Objektinitialisierung auf das aktuelle Datum und die aktuelle Uhrzeit
- NSDate (). TimeIntervalSince1970 // Aktuelles Datum und Uhrzeit in Sekunden ab 00:00:00 UTC am 1. Januar 1970.
- NSDate (). Compare (other: NSDate) // Gibt einen Vergleich des aktuellen Datums mit einem anderen Datum zurück und gibt ein `NSComparisonResult`

Bemerkungen

Es gibt verschiedene Arten von Datumsformaten, die Sie einstellen können: Hier ist eine vollständige Liste davon.

Format	Bedeutung / Beschreibung	Beispiel 1	Beispiel2
y	Ein Jahr mit mindestens einer Ziffer.	175 n. Chr. → „175“	2016 AD → „2016“
yy	Ein Jahr mit genau 2 Ziffern.	5 n. Chr. “05”	2016 AD → “16”
yyy	Ein Jahr mit mindestens 3 Ziffern.	5 n. Chr. “005”	2016 AD → „2016“
yyyy	Ein Jahr mit mindestens 4 Ziffern.	5 n. Chr. → „0005“	2016 AD → „2016“
M	Ein Monat mit mindestens einer Ziffer.	Juli → „7“	"November" → "11"
MM	Ein Monat mit mindestens 2 Ziffern.	Juli → “07”	"November" → "11"
MMM	Abkürzung für drei Buchstaben.	Juli → “Jul”	"November" → "Nov"
MMMM	Vollständiger Name des Monats	Juli → “Juli”	"November" → "November"
MMMMM	Abkürzung für einen Buchstabenmonat (Jan, Juni, Juli wird alle "J" haben).	Juli → “J”	"November" → "N"
d	Tag mit mindestens einer Ziffer.	8 → "8"	29 → „29“

Format	Bedeutung / Beschreibung	Beispiel 1	Beispiel2
dd	Tag mit mindestens zwei Ziffern.	8 → "08"	29 → „29“
"E", "EE" oder "EEE"	3-stelliger Tageskürzel für den Namen des Tages.	Montag → "Mo"	Donnerstag → "Do"
EEEE	Ganztägiger Name	Montag → "Montag"	Donnerstag → „Donnerstag“
EEEEE	1-stelliger Tageskürzel für den Namen des Tages. (Do und Di sind 'T')	Montag → "M"	Donnerstag → "T"
EEEEEEE	2-tägige Abkürzung des Tagesnamens.	Montag → „Mo“	Donnerstag → "Do"
ein	Tageszeit (AM / PM).	22 Uhr → "PM"	02.00 Uhr → „AM“
h	Eine 1-12-basierte Stunde mit mindestens einer Ziffer.	22 Uhr → "10"	2 Uhr morgens → "2"
hh	Eine 1-12-basierte Stunde mit mindestens 2 Ziffern.	22 Uhr → "10"	2 Uhr morgens → "02"
H	Eine 0-23-basierte Stunde mit mindestens 1 Ziffer.	22 Uhr → „14“	2 Uhr morgens → "2"
HH	Eine 0-23-basierte Stunde mit mindestens 2 Ziffern.	22 Uhr → „14“	2 Uhr morgens → "02"
m	Eine Minute mit mindestens einer Ziffer.	7 → "7"	29 → „29“
mm	Eine Minute mit mindestens 2 Ziffern.	7 → "07"	29 → „29“
s	Eine Sekunde mit mindestens einer Ziffer.	7 → "7"	29 → „29“
ss	Eine Sekunde mit mindestens 2 Ziffern.	7 → "07"	29 → „29“

Es gibt viele weitere, um unterschiedliche Zeiten basierend auf Zone (z) zu erhalten, um Zeit mit Millisekunden-Details (S) usw. zu erhalten.

Examples

Aktuelles Datum abrufen

Das aktuelle Datum zu bekommen ist sehr einfach. Sie erhalten das NSDate-Objekt des aktuellen Datums in nur einer Zeile wie folgt:

Schnell

```
var date = NSDate()
```

Swift 3

```
var date = Date()
```

Ziel c

```
NSDate *date = [NSDate date];
```

NSDate-Objekt N Sekunden ab dem aktuellen Datum abrufen

Die Anzahl der Sekunden ab dem aktuellen Datum und der aktuellen Uhrzeit für das neue Datum. Verwenden Sie einen negativen Wert, um ein Datum vor dem aktuellen Datum anzugeben.

Dafür haben wir eine Methode namens `dateWithTimeIntervalSinceNow(seconds: NSTimeInterval) -> NSDate` (Swift) oder `+(NSDate*)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds` (Objective-C).

Wenn Sie beispielsweise ein Datum benötigen, das eine Woche vom aktuellen Datum und eine Woche bis zum aktuellen Datum beträgt, können wir es als tun.

Schnell

```
let totalSecondsInWeek:NSTimeInterval = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
let nextWeek = NSDate().dateWithTimeIntervalSinceNow(totalSecondsInWeek)

//Using positive value for future date from today
let lastWeek = NSDate().dateWithTimeIntervalSinceNow(-totalSecondsInWeek)
```

Swift 3

```
let totalSecondsInWeek:TimeInterval = 7 * 24 * 60 * 60;

//Using positive value to add to the current date
let nextWeek = Date(timeIntervalSinceNow: totalSecondsInWeek)

//Using negative value to get date one week from current date
let lastWeek = Date(timeIntervalSinceNow: -totalSecondsInWeek)
```

Ziel c

```
NSTimeInterval totalSecondsInWeek = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
NSDate *lastWeek = [NSDate dateWithTimeIntervalSinceNow:-totalSecondsInWeek];

//Using positive value for future date from today
NSDate *nextWeek = [NSDate dateWithTimeIntervalSinceNow:totalSecondsInWeek];

NSLog(@"Last Week: %@", lastWeek);
NSLog(@"Right Now: %@", now);
NSLog(@"Next Week: %@", nextWeek);
```

Datumsvergleich

Es gibt 4 Methoden zum Vergleichen von Datumsangaben:

Schnell

- `isEqualToDate(anotherDate: NSDate) -> Bool`
- `earlierDate(anotherDate: NSDate) -> NSDate`
- `laterDate(anotherDate: NSDate) -> NSDate`
- `compare(anotherDate: NSDate) -> NSComparisonResult`

Ziel c

- - (BOOL)isEqualToDate:(NSDate *)anotherDate
- - (NSDate *)earlierDate:(NSDate *)anotherDate
- - (NSDate *)laterDate:(NSDate *)anotherDate
- - (NSComparisonResult)compare:(NSDate *)anotherDate

Nehmen wir an, wir haben 2 Termine:

Schnell

```
let date1: NSDate = ... // initialized as July 7, 2016 00:00:00
let date2: NSDate = ... // initialized as July 2, 2016 00:00:00
```

Ziel c

```
NSDate *date1 = ... // initialized as July 7, 2016 00:00:00
NSDate *date2 = ... // initialized as July 2, 2016 00:00:00
```

Um sie zu vergleichen, versuchen wir diesen Code:

Schnell

```

if date1.isEqualToDate(date2) {
    // returns false, as both dates aren't equal
}

earlierDate: NSDate = date1.earlierDate(date2) // returns the earlier date of the two (date 2)
laterDate: NSDate = date1.laterDate(date2) // returns the later date of the two (date1)

result: NSComparisonResult = date1.compare(date2)

if result == .OrderedAscending {
    // true if date1 is earlier than date2
} else if result == .OrderedSame {
    // true if the dates are the same
} else if result == .OrderedDescending {
    // true if date1 is later than date1
}

```

Ziel c

```

if ([date1 isEqualToDate:date2]) {
    // returns false, as both date are not equal
}

NSDate *earlierDate = [date1 earlierDate:date2]; // returns date which comes earlier from both
date, here it will return date2
NSDate *laterDate = [date1 laterDate:date2]; // returns date which comes later from both date,
here it will return date1

NSComparisonResult result = [date1 compare:date2];
if (result == NSOrderedAscending) {
    // fails
    // comes here if date1 is earlier then date2, in our case it will not come here
} else if (result == NSOrderedSame){
    // fails
    // comes here if date1 is same as date2, in our case it will not come here
} else{ // NSOrderedDescending
    // succeeds
    // comes here if date1 is later than date2, in our case it will come here
}

```

Wenn Sie Datumsangaben vergleichen und Sekunden, Wochen, Monate und Jahre behandeln möchten:

Swift 3

```

let dateStringUTC = "2016-10-22 12:37:48 +0000"
let dateFormatter = DateFormatter()
dateFormatter.locale = Locale(identifier: "en_US_POSIX")
dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss X"
let date = dateFormatter.date(from: dateStringUTC)!

let now = Date()

let formatter = DateComponentsFormatter()
formatter.unitsStyle = .full
formatter.maximumUnitCount = 2

```

```
let string = formatter.string(from: date, to: Date())! + " " + NSLocalizedString("ago",
comment: "added after elapsed time to say how long before")
```

Oder Sie können dies für jede Komponente verwenden:

```
// get the current date and time
let currentDateTime = Date()

// get the user's calendar
let userCalendar = Calendar.current

// choose which date and time components are needed
let requestedComponents: Set<Calendar.Component> = [
    .year,
    .month,
    .day,
    .hour,
    .minute,
    .second
]

// get the components
let dateTimeComponents = userCalendar.dateComponents(requestedComponents, from:
currentDateTime)

// now the components are available
dateTimeComponents.year
dateTimeComponents.month
dateTimeComponents.day
dateTimeComponents.hour
dateTimeComponents.minute
dateTimeComponents.second
```

Holen Sie sich Unix-Epoche-Zeit

Um die [Unix-](#) `timeIntervalSince1970` zu erhalten, verwenden Sie die konstante

`timeIntervalSince1970`:

Schnell

```
let date = NSDate() // current date
let unixtime = date.timeIntervalSince1970
```

Ziel c

```
NSDate *date = [NSDate date]; // current date
int unixtime = [date timeIntervalSince1970];
```

NSDateFormatter

Das Konvertieren eines `NSDate` Objekts in einen String erfolgt in nur 3 Schritten.

1. Erstellen Sie ein `NSDateFormatter` Objekt

Schnell

```
let dateFormatter = NSDateFormatter()
```

Swift 3

```
let dateFormatter = DateFormatter()
```

Ziel c

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
```

2. Legen Sie das Datumsformat fest, in dem Sie Ihre Zeichenfolge möchten

Schnell

```
dateFormatter.dateFormat = "yyyy-MM-dd 'at' HH:mm"
```

Ziel c

```
dateFormatter.dateFormat = @"yyyy-MM-dd 'at' HH:mm";
```

3. Holen Sie sich die formatierte Zeichenfolge

Schnell

```
let date = NSDate() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Swift 3

```
let date = Date() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Ziel c

```
NSDate *date = [NSDate date]; // your NSDate object
NSString *dateString = [dateFormatter stringFromDate:date];
```

Dies ergibt eine Ausgabe in etwa wie 2001-01-02 at 13:00 : 2001-01-02 at 13:00

Hinweis

Das Erstellen einer `NSDateFormatter` Instanz ist ein teurer Vorgang. `NSDateFormatter` wird empfohlen, sie einmal zu erstellen und nach Möglichkeit wiederzuverwenden.

Nützliche Erweiterung zum Konvertieren des Datums in einen String.

```
extension Date {
    func toString() -> String {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MMMM dd yyyy"
        return dateFormatter.string(from: self)
    }
}
```

Nützliche Links für die schnelle [Datumsbildung, die schnell lesbar werden kann](#) .

Zum Erstellen von Datumsformaten siehe [Muster für Datumsformate](#) .

Konvertieren Sie NSDate, das (nur) aus Stunde und Minute besteht, in ein vollständiges NSDate

Es gibt viele Fälle, in denen ein `NSDate` nur aus einem Stunden- und Minutenformat erstellt wurde, z. B. 08:12, das von einem Server als String zurückgegeben wird, und Sie initiieren eine `NSDate`-Instanz nur mit diesen **Werten**.

Der Nachteil für diese Situation ist, dass Ihr `NSDate` fast vollständig "nackt" ist und Sie Folgendes erstellen müssen: Tag, Monat, Jahr, Sekunde und Zeitzone, damit dieses Objekt mit anderen `NSDate`-Typen "zusammenspielen" kann.

Nehmen wir an, dass `hourAndMinute` der `NSDate`-Typ ist, der sich aus dem Stunden- und Minutenformat zusammensetzt.

Ziel c


```

NSDateComponents *hourAndMinuteComponents = [calendar components:NSCalendarUnitHour |
NSDateComponents *componentsOfDate = [[NSCalendar currentCalendar]
components:NSCalendarUnitDay | NSCalendarUnitMonth | NSCalendarUnitYear
NSDateComponents *components = [[NSDateComponents alloc] init];
[components setDay: componentsOfDate.day];
[components setMonth: componentsOfDate.month];
[components setYear: componentsOfDate.year];
[components setHour: [hourAndMinuteComponents hour]];
[components setMinute: [hourAndMinuteComponents minute]];
[components setSecond: 0];
[calendar setTimeZone: [NSTimeZone defaultTimeZone]];

NSDate *yourFullNSDateObject = [calendar dateFromComponents:components];

```

Jetzt ist dein Objekt das totale Gegenteil von "nackt".

UTC Zeitversatz von NSDate mit TimeZone

Hier wird der UTC Zeitversatz aus den aktuellen Daten in der gewünschten Zeitzone berechnet.

```

+(NSTimeInterval)getUTCOffsetIntervalWithCurrentTimeZone:(NSTimeZone *)current forDate:(NSDate
*)date {
    NSTimeZone *utcTimeZone = [NSTimeZone timeZoneWithAbbreviation:@"UTC"];
    NSInteger currentGMTOffset = [current secondsFromGMTForDate:date];
    NSInteger gmtOffset = [utcTimeZone secondsFromGMTForDate:date];
    NSTimeInterval gmtInterval = currentGMTOffset - gmtOffset;
    return gmtInterval;
}

```

Zeitzyklusart abrufen (12 Stunden oder 24 Stunden)

Prüfen, ob das aktuelle Datum das Symbol für AM oder PM enthält

Ziel c

```

NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
[formatter setLocale:[NSLocale currentLocale]];
[formatter setDateStyle:NSDateFormatterNoStyle];
[formatter setTimeStyle:NSDateFormatterShortStyle];
NSString *dateString = [formatter stringFromDate:[NSDate date]];
NSRange amRange = [dateString rangeOfString:[formatter AMSymbol]];
NSRange pmRange = [dateString rangeOfString:[formatter PMSymbol]];
BOOL is24h = (amRange.location == NSNotFound && pmRange.location == NSNotFound);

```

Anforderung des Zeitzyklus-Typs von `NSDateFormatter`

Ziel c

```
NSString *formatStringForHours = [NSDateFormatter dateFormatFromTemplate:@"j" options:0
locale:[NSLocale currentLocale]];
NSRange containsA = [formatStringForHours rangeOfString:@"a"];
BOOL is24h = containsA.location == NSNotFound;
```

Hierbei wird eine spezielle Datumsvorlagenzeichenfolge namens "j" verwendet, die gemäß der [ICU-Spezifikation](#) ...

[...] fordert das bevorzugte Stundenformat für das Gebietsschema (h, H, K oder k) an, das durch das bevorzugte Attribut des Stundenelements in den Zusatzdaten bestimmt wird. [...] Beachten Sie, dass die Verwendung von 'j' in einem an eine API übergebenen Skelett die einzige Möglichkeit ist, dass ein Skelett den bevorzugten Zeitzyklus-Typ eines Gebietsschemas (12 Stunden oder 24 Stunden) anfordert.

Dieser letzte Satz ist wichtig. "Es ist die einzige Möglichkeit, ein Skelett den bevorzugten Zeitzyklustyp eines Gebietsschemas anzufordern". Da `NSDateFormatter` und `NSCalendar` auf der ICU-Bibliothek `NSCalendar`, gilt dies auch hier.

Referenz

Die zweite Option wurde aus [dieser Antwort abgeleitet](#).

NSDate von JSON-Datumsformat abrufen `"/Date(1268123281843)/"`

Vor `Json.NET 4.5` wurden Datumsangaben im Microsoft-Format geschrieben: `"/Date(1198908717056)/"`. Wenn Ihr Server ein Datum in diesem Format sendet, können Sie den folgenden Code verwenden, um ihn in `NSDate` zu serialisieren:

Ziel c

```
(NSDate*) getDateFromJSON:(NSString *)dateString
{
    // Expect date in this format "/Date(1268123281843)/"
    int startPos = [dateString rangeOfString:@"("].location+1;
    int endPos = [dateString rangeOfString:@")"].location;
    NSRange range = NSMakeRange(startPos, endPos-startPos);
    unsigned long long milliseconds = [[dateString substringWithRange:range] longLongValue];
    NSLog(@"%llu", milliseconds);
    NSTimeInterval interval = milliseconds/1000;
    NSDate *date = [NSDate dateWithTimeIntervalSince1970:interval];
    // add code for date formatter if need NSDate in specific format.
    return date;
}
```

Holen Sie sich historische Zeit von NSDate (zB: vor 5s, 2m, 3h)

Dies kann in verschiedenen Chat-Anwendungen, RSS-Feeds und sozialen Apps verwendet werden, wo Sie aktuelle Feeds mit Zeitstempeln benötigen:

Ziel c

```
- (NSString *)getHistoricTimeText:(NSDate *)since
{
    NSString *str;
    NSTimeInterval interval = [[NSDate date] timeIntervalSinceDate:since];
    if(interval < 60)
        str = [NSString stringWithFormat:@"%is ago", (int)interval];
    else if(interval < 3600)
    {
        int minutes = interval/60;
        str = [NSString stringWithFormat:@"%im ago", minutes];
    }
    else if(interval < 86400)
    {
        int hours = interval/3600;

        str = [NSString stringWithFormat:@"%ih ago", hours];
    }
    else
    {
        NSDateFormatter *dateFormatter=[[NSDateFormatter alloc]init];
        [dateFormatter setLocale:[NSLocale currentLocale]];
        NSString *dateFormat = [NSDateFormatter dateFormatFromTemplate:@"MMM d, YYYY"
options:0 locale:[NSLocale currentLocale]];
        [dateFormatter setDateFormat:dateFormat];
        str = [dateFormatter stringFromDate:since];
    }
    return str;
}
```

NSDate online lesen: <https://riptutorial.com/de/ios/topic/1502/nsdate>

Kapitel 113: NSHTTPCookieStorage

Examples

Speichern und lesen Sie die Cookies von UserDefaults

```
import Foundation

class CookiesSingleton {

static let instance : CookiesSingleton = CookiesSingleton()
static var enableDebug = true

func loadCookies() {
    if let cookiesDetails =
        UserDefaults.standardUserDefaults().objectForKey("customeWebsite") {
        for (keys,_) in cookiesDetails as! NSDictionary{
            if let cookieDict = UserDefaults.standardUserDefaults().objectForKey(keys
as! String){
                if let cookie = NSHTTPCookie(properties:cookieDict as! [String:AnyObject])
{
                    NSHTTPCookieStorage.sharedHTTPCookieStorage().setCookie(cookie)
                    if(CookiesSingleton.enableDebug){
                        print("Each Cookies",cookieDict)
                    }
                }
            }
        }
    }
}

func removeCookies(){
    NSURLCache.sharedURLCache().removeAllCachedResponses()
    NSURLCache.sharedURLCache().diskCapacity = 0
    NSURLCache.sharedURLCache().memoryCapacity = 0

    let storage : NSHTTPCookieStorage = NSHTTPCookieStorage.sharedHTTPCookieStorage()
    for cookie in storage.cookies! {
        storage.deleteCookie(cookie as NSHTTPCookie)
    }

    UserDefaults.standardUserDefaults().setValue("", forKey: "customeWebsite")
    UserDefaults.standardUserDefaults().synchronize()

    if(CookiesSingleton.enableDebug){
        print("Cookies Removed")
    }
}

func saveCookies() {

    let cookieArray = NSMutableArray()
    let savedC = NSHTTPCookieStorage.sharedHTTPCookieStorage().cookies

    let allCookiesDic:NSMutableDictionary = NSMutableDictionary()
```

```

for c : NSHTTPCookie in savedC! {

    let cookieProps = NSMutableDictionary()
    cookieArray.addObject(c.name)
    cookieProps.setValue(c.name, forKey: NSHTTPCookieName)
    cookieProps.setValue(c.value, forKey: NSHTTPCookieValue)
    cookieProps.setValue(c.domain, forKey: NSHTTPCookieDomain)
    cookieProps.setValue(c.path, forKey: NSHTTPCookiePath)
    cookieProps.setValue(c.version, forKey: NSHTTPCookieVersion)
    cookieProps.setValue(NSDate().dateByAddingTimeInterval(2629743), forKey:
NSHTTPCookieExpires)

    allCookiesDic.setValue(cookieProps, forKey: c.name)

}
NSUserDefaults.standardUserDefaults().setValue(allCookiesDic, forKey: "customeWebsite")
NSUserDefaults.standardUserDefaults().synchronize()

if(CookiesSingleton.enableDebug){
    print("Cookies Saved")
}
}
}

```

NSHTTPCookieStorage online lesen: <https://riptutorial.com/de/ios/topic/7312/nshttpcookiestorage>

Kapitel 114: NSInvocation

Examples

NSInvocation Objective-C

Siehe diesen [ursprünglichen Beitrag](#) von [e.James](#)

Laut [Apples NSInvocation-Klassenreferenz](#) :

Eine `NSInvocation` ist eine Objective-C-Nachricht, die statisch dargestellt wird, d.
`NSInvocation` Eine Aktion, die in ein Objekt umgewandelt wird.

Und *etwas* ausführlicher:

Der Begriff der Botschaften ist zentral für die Ziel-Philosophie. Immer wenn Sie eine Methode aufrufen oder auf eine Variable eines Objekts zugreifen, senden Sie ihr eine Nachricht.

`NSInvocation` wenn Sie eine Nachricht zu einem anderen Zeitpunkt an ein Objekt senden oder dieselbe Nachricht mehrmals senden möchten. `NSInvocation` können Sie die Nachricht *beschreiben*, die Sie senden `NSInvocation` , und sie später *aufrufen* (tatsächlich an das Zielobjekt senden).

Angenommen, Sie möchten einem Array einen String hinzufügen. Sie würden die Nachricht `addObject:` normalerweise wie folgt senden:

```
[myArray addObject:myString];
```

`NSInvocation` , Sie möchten `NSInvocation` , um diese Nachricht zu einem anderen Zeitpunkt zu senden:

Zuerst würden Sie ein `NSInvocation` Objekt für die Verwendung mit dem `addObject:` Selector von `NSMutableArray` `addObject:` :

```
NSMethodSignature * mySignature = [NSMutableArray  
    instanceMethodSignatureForSelector:@selector(addObject:)];  
NSInvocation * myInvocation = [NSInvocation  
    invocationWithMethodSignature:mySignature];
```

Als Nächstes geben Sie an, an welches Objekt die Nachricht gesendet werden soll:

```
[myInvocation setTarget:myArray];
```

Geben Sie die Nachricht an, die Sie an dieses Objekt senden möchten:

```
[myInvocation setSelector:@selector(addObject:)];
```

Und geben Sie alle Argumente für diese Methode ein:

```
[myInvocation setArgument:&myString atIndex:2];
```

Beachten Sie, dass Objektargumente vom Zeiger übergeben werden müssen. Vielen Dank an [Ryan McCuaig](#) für diesen Hinweis. Weitere Informationen finden Sie in [der Dokumentation von Apple](#) .

Zu diesem Zeitpunkt ist `myInvocation` ein vollständiges Objekt, das eine Nachricht beschreibt, die gesendet werden kann. Um die Nachricht tatsächlich zu senden, rufen Sie an:

```
[myInvocation invoke];
```

In diesem letzten Schritt wird die Nachricht gesendet, wobei im Wesentlichen `[myArray addObject:myString];` .

Stellen Sie sich das als eine E-Mail vor. Sie öffnen eine neue E-Mail (`NSInvocation` Objekt), geben die Adresse der Person (Objekt) ein, an die Sie die E-Mail senden möchten, geben eine Nachricht für den Empfänger ein (geben Sie einen `selector` und Argumente an) und klicken Sie auf "Senden". (Aufruf `invoke`).

Weitere Informationen finden Sie unter [Verwenden von NSInvocation](#) .

`NSUndoManager` verwendet `NSInvocation` Objekte, *um Befehle umzukehren* . Im Wesentlichen erstellen Sie ein `NSInvocation` Objekt, um `NSInvocation` zu sagen: "Hey, wenn Sie rückgängig machen wollen, was ich gerade getan habe, senden Sie diese Nachricht mit diesen Argumenten an dieses Objekt." Sie übergeben das `NSInvocation` Objekt an den `NSUndoManager` und fügen dieses Objekt einem Array von rückgängig gemachten Aktionen hinzu. Wenn der Benutzer "Undo" `NSUndoManager` , sucht `NSUndoManager` einfach nach der neuesten Aktion im Array und ruft das gespeicherte `NSInvocation` Objekt auf, um die erforderliche Aktion auszuführen.

Weitere Informationen finden Sie unter [Registrieren von Rückgängig-Operationen](#) .

NSInvocation online lesen: <https://riptutorial.com/de/ios/topic/8276/nsinvocation>

Kapitel 115: NSNotificationCenter

Einführung

iOS-Benachrichtigungen sind eine einfache und leistungsstarke Methode, um Daten lose gekoppelt zu senden. Das heißt, der Absender einer Benachrichtigung muss sich nicht darum kümmern, von wem (wenn jemand) die Benachrichtigung erhalten wird. Er wird nur an den Rest der App weitergeleitet, und er kann von vielen Dingen abgerufen werden oder abhängig davon, was nicht Der Status Ihrer App.

Quelle : - [HACKING mit Swift](#)

Parameter

Parameter	Einzelheiten
Name	Name der Benachrichtigung, für die der Beobachter registriert werden soll; Das heißt, nur Benachrichtigungen mit diesem Namen werden verwendet, um den Block zur Operationswarteschlange hinzuzufügen. Wenn Sie null übergeben, verwendet das Benachrichtigungscenter nicht den Namen einer Benachrichtigung, um zu entscheiden, ob der Block der Operationswarteschlange hinzugefügt werden soll.
obj	Das Objekt, dessen Benachrichtigungen der Beobachter erhalten möchte; es werden also nur Benachrichtigungen dieses Absenders an den Beobachter gesendet. Wenn Sie null übergeben, verwendet die Benachrichtigungszentrale nicht den Absender einer Benachrichtigung, um zu entscheiden, ob sie dem Beobachter zugestellt werden soll.
Warteschlange	Die Operationswarteschlange, zu der der Block hinzugefügt werden soll. Wenn Sie nil übergeben, wird der Block synchron im Buchungsthread ausgeführt.
Block	Der Block, der ausgeführt werden soll, wenn die Benachrichtigung empfangen wird. Der Block wird vom Benachrichtigungscenter kopiert und (die Kopie) gehalten, bis die Beobachterregistrierung entfernt wird.

Bemerkungen

Ein NSNotificationCenter-Objekt (oder einfach eine Benachrichtigungszentrale) stellt einen Mechanismus zum Senden von Informationen innerhalb eines Programms bereit. Ein NSNotificationCenter-Objekt ist im Wesentlichen eine Benachrichtigungstabelle.

Für mehr Informationen, überprüfen Sie die Apple - Dokumentation heraus [hier](#)

Examples

Beobachter hinzufügen

Namenskonvention

Benachrichtigungen werden durch globale NSString-Objekte identifiziert, deren Namen folgendermaßen zusammengesetzt sind:

Name of associated class + Did | Will + UniquePartOfName + Notification

Zum Beispiel:

- UIApplicationDidBecomeActiveNotification
- UIWindowDidMiniaturizeNotification
- NSTextViewDidChangeSelectionNotification
- NSColorPanelColorDidChangeNotification

Schnell 2.3

```
NSNotificationCenter.defaultCenter().addObserver(self,
                                                selector:
#selector(self.testNotification(_:)),
                                                name: "TestNotification",
                                                object: nil)
```

Swift 3

```
NSNotificationCenter.default.addObserver(self,
                                        selector: #selector(self.testNotification(_:)),
                                        name: NSNotification.Name(rawValue:
"TestNotification"),
                                        object: nil)
```

Ziel c

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                        selector:@selector(testNotification:)
                                        name:@"TestNotification"
                                        object:nil];
```

PS: Es ist auch erwähnenswert, dass die Anzahl der Hinzufügungen eines Beobachters genau der

Anzahl der Entfernungen des Beobachters entsprechen muss. Ein Anfängerfehler ist das Hinzufügen des Beobachters in `viewWillAppear:` eines `UIViewController`s, das Entfernen des Beobachters in `viewDidUnload:` bewirkt jedoch eine ungerade Anzahl von `viewDidUnload:`, wodurch der Beobachter und der Benachrichtigungsselektor überflüssig werden.

Beobachter entfernen

Schnell 2.3

```
//Remove observer for single notification
NSNotificationCenter defaultCenter().removeObserver(self, name: "TestNotification", object: nil)

//Remove observer for all notifications
NSNotificationCenter defaultCenter().removeObserver(self)
```

Swift 3

```
//Remove observer for single notification
NotificationCenter.default.removeObserver(self, name: NSNotification.Name(rawValue: "TestNotification"), object: nil)

//Remove observer for all notifications
NotificationCenter.default.removeObserver(self)
```

Ziel c

```
//Remove observer for single notification
[[NSNotificationCenter defaultCenter] removeObserver:self name:@"TestNotification" object:nil];

//Remove observer for all notifications
[[NSNotificationCenter defaultCenter] removeObserver:self];
```

Benachrichtigung versenden

Schnell

```
NSNotificationCenter defaultCenter().postNotificationName("TestNotification", object: self)
```

Ziel c

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"TestNotification" object:nil];
```

Benachrichtigung mit Daten buchen

Schnell

```
let userInfo: [String: AnyObject] = ["someKey": myObject]
NSNotificationCenter.defaultCenter().postNotificationName("TestNotification", object: self,
userInfo: userInfo)
```

Ziel c

```
NSDictionary *userInfo = [NSDictionary dictionaryWithObject:myObject forKey:@"someKey"];
[[NSNotificationCenter defaultCenter] postNotificationName: @"TestNotification" object:nil
userInfo:userInfo];
```

Benachrichtigung beobachten

Schnell

```
func testNotification(notification: NSNotification) {
    let userInfo = notification.userInfo
    let myObject: MyObject = userInfo["someKey"]
}
```

Ziel c

```
- (void)testNotification:(NSNotification *)notification {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}
```

Beobachter mit Block hinzufügen / entfernen

Anstatt einen Beobachter mit einem Selektor hinzuzufügen, kann ein Block verwendet werden:

```
id testObserver = [[NSNotificationCenter defaultCenter] addObserverForName:@"TestNotification"
                                                                    object:nil
                                                                    queue:nil
                                                                    usingBlock:^(NSNotification*
notification) {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}]];
```

Der Beobachter kann dann entfernt werden mit:

```
[[NSNotificationCenter defaultCenter] removeObserver:testObserver
                                     name:@"TestNotification"
                                     object:nil];
```

Beobachter für Namen hinzufügen und entfernen

```
// Add observer
let observer =
NSNotificationCenter.defaultCenter().addObserverForName("nameOfTheNotification", object: nil,
queue: nil) { (notification) in
    // Do operations with the notification in this block
}

// Remove observer
NSNotificationCenter.defaultCenter().removeObserver(observer)
```

NSNotificationCenter online lesen: <https://riptutorial.com/de/ios/topic/1601/nsnotificationcenter>

Kapitel 116: NSPredicate

Syntax

- Zeichenketten-Substitutionen des Prädikats
 - Zeichenkettenbezeichner für das C-Format: % d, % s, % f usw
 - Objektersetzung: % @
 - Keypath-Substitution: % K
- Vergleichselemente für Vergleichselemente
 - =, ==: Der Ausdruck auf der linken Seite entspricht dem Ausdruck auf der rechten Seite
 - > =, ==>: Der Ausdruck auf der linken Seite ist größer oder gleich dem Ausdruck auf der rechten Seite
 - < =, = <: Der Ausdruck auf der linken Seite ist kleiner oder gleich dem Ausdruck auf der rechten Seite
 - >: Der Ausdruck auf der linken Seite ist größer als der Ausdruck auf der rechten Seite
 - <: Der Ausdruck auf der linken Seite ist weniger als der Ausdruck auf der rechten Seite
 - !=, <>: Der Ausdruck auf der linken Seite ist nicht gleich dem Ausdruck auf der rechten Seite
 - BETWEEN: Der Ausdruck auf der linken Seite liegt zwischen oder gleich einem der Werte im rechten Ausdruck, der untere und obere Grenzen angibt. Beispiel: BETWEEN {0, 5}.
- Prädikatsverbundoperatoren
 - AND, &&: Logisches AND
 - ODER, ||: Logisches ODER
 - NICHT!: Logisch NICHT
- Vergleichselemente für Vergleichselemente
 - BEGINSWITH: Der Ausdruck auf der linken Seite beginnt mit dem Ausdruck auf der rechten Seite
 - ENDSWITH: Der Ausdruck auf der linken Seite endet mit dem Ausdruck auf der rechten Seite
 - CONTAINS: Der Ausdruck auf der linken Seite enthält den Ausdruck auf der rechten Seite
 - LIKE: Der Ausdruck auf der linken Seite entspricht dem Ausdruck auf der rechten Seite mit Platzhalter
 - *: Entspricht keinem oder mehreren Zeichen
 - ?: Stimmt mit einem Zeichen überein

Examples

Erstellen eines NSPredicate mit predicateWithBlock

Ziel c

```
NSPredicate *predicate = [NSPredicate predicateWithBlock:^(BOOL(id item,
                                                                    NSDictionary *bindings) {
    return [item isKindOfClass:[UILabel class]];
}]];
```

Schnell

```
let predicate = NSPredicate { (item, bindings) -> Bool in
    return item.isKindOfClass(UILabel.self)
}
```

In diesem Beispiel `UILabel` das Prädikat mit Elementen der Klasse `UILabel`.

Erstellen eines NSPredicate mit predicateWithFormat

Ziel c

```
NSPredicate *predicate = [NSPredicate predicateWithFormat: @"self[SIZE] = %d", 5];
```

Schnell

```
let predicate = NSPredicate(format: "self[SIZE] >= %d", 5)
```

In diesem Beispiel stimmt das Prädikat mit Elementen überein, die Arrays mit einer Länge von mindestens 5 sind.

Ein NSPredicate mit Substitutionsvariablen erstellen

Ein `NSPredicate` kann Substitutionsvariablen verwenden, um Werte fliegend zu binden.

Ziel c

```
NSPredicate *template = [NSPredicate predicateWithFormat: @"self BEGINSWITH $letter"];
NSDictionary *variables = @{@"letter": @"r"};
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables: variables];
```

Schnell

```
let template = NSPredicate(format: "self BEGINSWITH $letter")
let variables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(variables)
```

Das Vorlagenprädikat wird von `predicateWithSubstitutionVariables` nicht geändert. Stattdessen

wird eine Kopie erstellt und diese Kopie erhält die Substitutionsvariablen.

Verwenden von NSPredicate zum Filtern eines Arrays

Ziel c

```
NSArray *heroes = @[@"tracer", @"bastion", @"reaper", @"junkrat", @"roadhog"];

NSPredicate *template = [NSPredicate predicateWithFormat:@"self BEGINSWITH $letter"];

NSDictionary *beginsWithRVariables = @{ @"letter": @"r"};
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables:
beginsWithRVariables];

NSArray *beginsWithRHeroes = [heroes filteredArrayUsingPredicate: beginsWithR];
// ["reaper", "roadhog"]

NSDictionary *beginsWithTVariables = @{ @"letter": @"t"};
NSPredicate *beginsWithT = [template predicateWithSubstitutionVariables: beginsWithTVariables];

NSArray *beginsWithTHeroes = [heroes filteredArrayUsingPredicate: beginsWithT];
// ["tracer"]
```

Schnell

```
let heroes = ["tracer", "bastion", "reaper", "junkrat", "roadhog"]

let template = NSPredicate(format: "self BEGINSWITH $letter")

let beginsWithRVariables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(beginsWithRVariables)

let beginsWithRHeroes = heroes.filter { beginsWithR.evaluateWithObject($0) }
// ["reaper", "roadhog"]

let beginsWithTVariables = ["letter": "t"]
let beginsWithT = template.predicateWithSubstitutionVariables(beginsWithTVariables)

let beginsWithTHeroes = heroes.filter { beginsWithT.evaluateWithObject($0) }
// ["tracer"]
```

Formularvalidierung mit NSPredicate

```
NSString *emailRegex = @"[A-Z0-9a-z]([A-Z0-9a-z._-]{0,64})+[A-Z0-9a-z]+@[A-Z0-9a-z]+([A-Za-z0-9.-]{0,64})+([A-Z0-9a-z])+\.[A-Za-z]{2,4}";
NSString *firstNameRegex = @"[0-9A-Za-z]{2,32}$";
NSString *lastNameRegex = @"[0-9A-Za-z\''-]{2,32}$";
NSString *mobileNumberRegex = @"^[0-9]{10}$";
NSString *zipcodeRegex = @"^[0-9]{5}$";
NSString *SSNRegex = @"^\d{3}-?\d{2}-?\d{4}$";
NSString *addressRegex = @"^[ A-Za-z0-9]{2,32}$";
NSString *cityRegex = @"^[ A-Za-z0-9]{2,25}$";
```

```

NSString *PINRegex = @"^[0-9]{4}$";
NSString *driversLiscRegex = @"^[0-9a-zA-Z]{5,20}$";

-(BOOL)validateEmail {
    //Email address field should give an error when the email address begins with ".", "-", "_"
    .
    NSPredicate *emailPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
emailRegex];
    return ([emailPredicate evaluateWithObject:self.text] && self.text.length <= 64 &&
([self.text rangeOfString:@".."].location == NSNotFound));
}

- (BOOL)validateFirstName {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateLastName {
    NSPredicate *lastNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
lastNameRegex];
    return [lastNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateAlphaNumericMin2Max32 {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateMobileNumber {
    NSString *strippedMobileNumber = [[[[self.text stringByReplacingOccurrencesOfString:@"("
withString:@""]
                                stringByReplacingOccurrencesOfString:@")"
withString:@""]
                                stringByReplacingOccurrencesOfString:@"-"
withString:@""]
                                stringByReplacingOccurrencesOfString:@" "
withString:@""];

    NSPredicate *mobileNumberPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
mobileNumberRegex];

    return [mobileNumberPredicate evaluateWithObject:strippedMobileNumber];
}

- (BOOL)validateZipcode {
    NSPredicate *zipcodePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
zipcodeRegex];

    return [zipcodePredicate evaluateWithObject:self.text];
}

- (BOOL)validateSSN {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", SSNRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateAddress {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",

```



```

addressRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateCity {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", cityRegex];
    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validatePIN {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", PINRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateDriversLiscNumber {
    if([self.text length] > 20) {
        return NO;
    }
    NSPredicate *driversLiscPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
driversLiscRegex];

    return [driversLiscPredicate evaluateWithObject:self.text];
}

```

NSPredicate mit "AND", "OR" und "NOT" -Bedingung

Das bedingte Prädikat wird sauberer und sicherer, wenn Sie die Klasse `NSCompoundPredicate`, die grundlegende boolesche Operatoren für die angegebenen Prädikate bereitstellt.

Ziel c

UND - Bedingung

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
NSPredicate *combinedPredicate = [NSCompoundPredicate andPredicateWithSubpredicates:
@[predicate,anotherPredicate]];

```

ODER - Bedingung

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
NSPredicate *combinedPredicate = [NSCompoundPredicate orPredicateWithSubpredicates:
@[predicate,anotherPredicate]];

```

NICHT - Bedingung

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];

```

```
NSPredicate *combinedPredicate = [NSCompoundPredicate notPredicateWithSubpredicate:  
@[predicate, anotherPredicate]];
```

NSPredicate online lesen: <https://riptutorial.com/de/ios/topic/5796/nspredicate>

Kapitel 117: NSTimer

Parameter

Parameter	Einzelheiten
<code>interval</code>	Die Wartezeit in Sekunden, bevor der Timer ausgelöst wird. oder bei sich wiederholenden Timern die Zeit zwischen den Zündungen.
<code>target</code>	Das Objekt, in dem der <code>selector</code> aufgerufen werden soll
<code>selector</code>	In Swift ein <code>Selector</code> Objekt, das die Methode angibt, die auf dem <code>target</code> aufgerufen werden soll
<code>repeats</code>	Wenn <code>false</code> , feuert die Timer nur einmal. Wenn <code>true</code> , Feuer, um die Timer alle <code>interval</code> Sekunden.

Bemerkungen

Mit einem `NSTimer` können Sie eine Nachricht nach `NSTimer` einer bestimmten Zeit an ein Ziel senden.

Examples

Timer erstellen

Dadurch wird ein Timer erstellt, um die `doSomething` Methode innerhalb von 5 Sekunden `self` aufzurufen.

Schnell

```
let timer = NSTimer.scheduledTimerWithTimeInterval(5,
    target: self,
    selector: Selector(doSomething()),
    userInfo: nil,
    repeats: false)
```

Swift 3

```
let timer = Timer.scheduledTimer(timeInterval: 1,
    target: self,
    selector: #selector(doSomething()),
    userInfo: nil,
    repeats: true)
```

Ziel c

```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self
selector:@selector(doSomething) userInfo:nil repeats:NO];
```

Wenn Sie die Wiederholungen auf `false/NO` bedeutet dies, dass der Timer nur einmal ausgelöst werden soll. Wenn wir dies auf `true/YES`, wird es alle fünf Sekunden `true/YES`, bis es manuell deaktiviert wird.

Einen Timer manuell auslösen

Schnell

```
timer.fire()
```

Ziel c

```
[timer fire];
```

Das Aufrufen von `fire` bewirkt eine NSTimer die Aufgabe auszuführen wäre es in der Regel nach einem Zeitplan durchgeführt haben.

Bei einem sich **nicht wiederholenden Timer** wird der Timer dadurch automatisch ungültig. Das heißt, das Aufrufen eines `fire` vor Ablauf des Zeitintervalls führt nur zu einem Aufruf.

In einem sich **wiederholenden Zeitgeber** wird die Aktion einfach aufgerufen, ohne den üblichen Zeitplan zu unterbrechen.

Timer ungültig machen

Schnell

```
timer.invalidate()
```

Ziel c

```
[timer invalidate];
```

Dadurch wird der Timer angehalten. **Muss der Timer aus dem Thread aufgerufen werden erstellt wurde**, siehe [Apples Notizen](#) :

Sie müssen diese Nachricht von dem Thread senden, in dem der Timer installiert wurde. Wenn Sie diese Nachricht von einem anderen Thread senden, wird die mit dem Zeitgeber verknüpfte Eingabequelle möglicherweise nicht aus der Laufschleife entfernt, wodurch der Thread möglicherweise nicht ordnungsgemäß beendet wird.

Hinweise: Nachdem der Timer ungültig gemacht wurde, ist es unmöglich, den gleichen ungültigen Timer auszulösen. Stattdessen müssen Sie den ungültigen Timer erneut initialisieren und die Feuermethode auslösen.

Wiederholtes Timer-Ereignis

Schnell

```
class ViewController: UIViewController {  
  
    var timer = NSTimer()  
  
    override func viewDidLoad() {  
        NSTimer.scheduledTimerWithTimeInterval(1.0, target: self, selector:  
Selector(self.timerMethod()), userInfo: nil, repeats: true)  
    }  
  
    func timerMethod() {  
        print("Timer method called")  
    }  
  
    func endTimer() {  
        timer.invalidate()  
    }  
}
```

Swift 3

```
class ViewController: UIViewController {  
  
    var timer = Timer()  
  
    override func viewDidLoad() {  
        Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:  
#selector(self.timerMethod()), userInfo: nil, repeats: true)  
    }  
  
    func timerMethod() {  
        print("Timer method called")  
    }  
  
    func endTimer() {  
        timer.invalidate()  
    }  
}
```

Muss manuell entwertet werden, falls gewünscht.

Schnell

Nicht wiederholtes verzögertes Timer-Ereignis

```
NSTimer.scheduledTimerWithTimeInterval(3.0, target: self, selector:  
Selector(self.timerMethod()), userInfo: nil, repeats: false)
```

Swift 3

```
Timer.scheduledTimer(timeInterval: 3.0, target: self, selector:
#selector(self.timerMethod()), userInfo: nil, repeats: false)
```

Der Timer wird einmal 3 Sekunden nach der Ausführung ausgelöst. Wird automatisch gelöscht, sobald sie ausgelöst wird.

Weitergabe von Daten mit Timer

Wenn Sie Daten mit dem Timer-Trigger übergeben möchten, können Sie dies mit dem Parameter `userInfo` tun.

Hier ist der einfache Ansatz, der einen kurzen Überblick darüber gibt, wie Sie die Daten vom Timer an die ausgelöste Methode übergeben können.

[*Swift 3*]

```
Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:#selector(iGotCall(sender:)),
userInfo: ["Name": "i am iOS guy"], repeats:true)
```

[*Ziel - C*]

```
NSTimer* timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                    target:self
                    selector:@selector(iGotCall:)
                    userInfo:@"i am iOS guy" repeats:YES];
```

Die obige Codezeile führt `["Name": "i am iOS guy"]` in die `userInfo`. Wenn nun der `iGotCall` angerufen wird, können Sie den übergebenen Wert als Code-Snippet abrufen.

[*Swift 3*]

```
func iGotCall(sender: Timer) {
    print((sender.userInfo!))
}
```

[*Ziel - C*]

```
- (void)iGotCall:(NSTimer*)theTimer {
    NSLog(@"%@", (NSString*)[theTimer userInfo]);
}
```

NSTimer online lesen: <https://riptutorial.com/de/ios/topic/2624/nstimer>

Kapitel 118: NSURL

Examples

So erhalten Sie die letzte Zeichenfolgekomponente aus NSURL-Zeichenfolge.

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/images/apple-tree.jpg"];
NSString *fileName = [url lastPathComponent];
// fileName = "apple-tree.jpg"
```

So erhalten Sie die letzte Zeichenfolgekomponente von der URL (NSURL) in Swift

Schnell 2.3

```
let url = NSURL(string: "http://google.com/lastPath")
let lastPath = url?.lastPathComponent
```

Swift 3,0

```
let url = URL(string: "http://google.com/lastPath")
let lastPath = url?.lastPathComponent
```

NSURL online lesen: <https://riptutorial.com/de/ios/topic/4610/nsurl>

Kapitel 119: NSURLConnection

Examples

Methoden delegieren

// das NSURLConnectionDelegate-Protokoll anpassen.

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

// Implementierung der NSURLConnection-Protokollmethoden.

```
#pragma mark NSURLConnection Delegate Methods

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    // A response has been received, this is where we initialize the instance var you created
    // so that we can append data to it in the didReceiveData method
    // Furthermore, this method is called each time there is a redirect so reinitializing it
    // also serves to clear it
    _responseData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    // Append the new data to the instance variable you declared
    [_responseData appendData:data];
}

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
    willCacheResponse:(NSCachedURLResponse*)cachedResponse {
    // Return nil to indicate not necessary to store a cached response for this connection
    return nil;
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    // The request is complete and data has been received
    // You can parse the stuff in your instance variable now
}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    // The request has failed for some reason!
    // Check the error var
}
```

Synchrone Anforderung

```
NSURLRequest * urlRequest = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];
NSURLResponse * response = nil;
```



```
NSError * error = nil;
NSData * data = [NSURLConnection sendSynchronousRequest:urlRequest
                                returningResponse:&response
                                error:&error];

if (error == nil)
{
    // Parse data here
}
```

Asynchrone Anforderung

```
// Create the request instance.
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];

// Create url connection and fire request
NSURLConnection *conn = [[NSURLConnection alloc] initWithRequest:request delegate:self];
```

NSURLConnection online lesen: <https://riptutorial.com/de/ios/topic/6004/nsurlconnection>

Kapitel 120: NSURLSession

Bemerkungen

Die **NSURLSession**- Klasse und verwandte Klassen stellen eine API zum Herunterladen von Inhalten bereit. Diese API bietet eine Vielzahl von Delegatmethoden für die Unterstützung der Authentifizierung und gibt Ihrer App die Möglichkeit, Hintergrund-Downloads durchzuführen, wenn Ihre App nicht ausgeführt wird oder in iOS, wenn Ihre App angehalten ist.

Auf hoher Ebene basiert **NSURLSession** auf dem Konzept von Sitzungen und Aufgaben. Eine Aufgabe stellt eine einzelne Anforderung für eine einzelne URL (oder einen einzelnen Upload zu einer einzelnen URL) dar. Eine Sitzung ist eine Gruppe verwandter Anforderungen.

Das Betriebssystem stellt eine einzige bereits vorhandene Sitzung bereit - die gemeinsam genutzte Sitzung, die im Wesentlichen wie `NSURLConnection` funktioniert. Darüber hinaus können Sie nach Bedarf eigene Sitzungen in Ihrer App erstellen.

Verschiedene Apps verwenden Sitzungen auf unterschiedliche Weise. Viele Apps erstellen beim Start eine einzige Sitzung, die sie einfach wiederverwenden. Andere Apps profitieren von der Möglichkeit, eine Gruppe verwandter Aufgaben abzubrechen (z. B. ein Webbrowser, der alle ausstehenden Anforderungen abbricht, wenn Sie eine Registerkarte schließen) und somit eine Sitzung für jede Gruppe verwandter Anforderungen erstellen.

Der erste Schritt bei der Verwendung von `NSURLSession` ist das Erstellen eines Sitzungskonfigurationsobjekts. Das (normalerweise) wiederverwendbare Objekt enthält verschiedene Sitzungseinstellungen, die Sie für Ihre speziellen Bedürfnisse anpassen können, z. B. maximale Parallelität, zusätzliche Header, die mit jeder Anfrage gesendet werden sollen, ob Anfragen über das Mobilfunkgerät (nur iOS) gesendet werden dürfen, Timeouts, Anmeldeinformationsspeicher, minimale TLS-Version und sogar Proxy-Einstellungen.

Es gibt drei Arten von Sitzungskonfigurationen, je nachdem, wie sich die resultierende Sitzung verhalten soll:

- **Standardkonfigurationen** erstellen Sitzungen, die ähnlich wie `NSURLConnection` funktionieren.
- **Hintergrundkonfigurationen** erstellen Sitzungen, in denen Anforderungen außerhalb des Prozesses ausgeführt werden, sodass Downloads auch dann fortgesetzt werden können, wenn die App nicht mehr ausgeführt wird.
- **Ephemere Konfigurationen** erstellen Sitzungen, in denen nichts auf der Festplatte zwischengespeichert wird, keine Cookies auf der Festplatte usw. gespeichert werden und sich daher zum Sichern von Objekten wie inkognito Browserfenstern eignen.

Wenn Sie eine Hintergrundkonfiguration erstellen, müssen Sie einen Sitzungsbezeichner angeben, mit dem Sie die Hintergrundsituation später erneut zuordnen können (wenn Ihre App beendet ist oder vom Betriebssystem angehalten oder beendet wird). Es darf nicht mehr als eine Instanz einer Sitzung mit der gleichen ID in Ihrer App aktiv sein. Diese Konfigurationen sind in der

Regel nicht wiederverwendbar. Alle anderen Sitzungskonfigurationen können wiederverwendet werden, um beliebig viele Sitzungen zu erstellen. Wenn Sie also mehrere Sitzungen mit ähnlichen Einstellungen erstellen müssen, können Sie die Konfiguration einmal erstellen und bei jeder neuen Sitzung erneut verwenden.

Nachdem Sie eine Sitzung erstellt haben, können Sie in dieser Sitzung Aufgaben erstellen. Es gibt drei Arten von Aufgaben:

- **Datenaufgaben** geben Daten als **NSData**- Objekt zurück. Diese sind für die allgemeine Verwendung geeignet, werden jedoch in Hintergrundsitzungen nicht unterstützt.
- **Download-Aufgaben** geben Daten als Datei auf der Festplatte zurück. Diese eignen sich für größere Anforderungen oder für die Verwendung in Hintergrundsitzungen.
- **Aufgaben** hochladen Daten von einem NSData-Objekt oder von einer Datei auf der Festplatte hochladen. Sie stellen ein Datenobjekt oder eine Datei bereit, die den POST-Body bereitstellt. Die Body-Daten / -Datei, die Sie für die Task bereitstellen, überschreibt alle Body-Daten / -Dateien, die im NSURLRequest-Objekt (falls vorhanden) bereitgestellt werden.

Mit jedem dieser Typen können Sie die Antwortdaten auf verschiedene Arten abrufen, indem Sie blockbasierte Rückrufe verwenden oder einen Delegaten für die Sitzung bereitstellen und Delegatmethoden implementieren.

Darüber hinaus können Sie mit NSURLSession Delegierungsmethoden für die Authentifizierung, die benutzerdefinierte TLS-Zertifikatsverarbeitung (sowohl für Clientzertifikate als auch für die Serverüberprüfung), das Ändern des Cache-Verhaltens usw. bereitstellen.

Examples

Einfache GET-Anfrage

```
// define url
let url = NSURL(string: "https://urlToGet.com")

//create a task to get data from a url
let task = NSURLSession.sharedSession().dataTaskWithURL(url!)
{
    /*inside this block, we have access to NSData *data, NSURLResponse *response, and
    NSError *error returned by the dataTaskWithURL() function*/
    (data, response, error) in

    if error == nil
    {
        // Data from the request can be manipulated here
    }
    else
    {
        // An error occurred
    }
}

//make the request
```

```
task.resume()
```

Ziel-C Erstellen Sie eine Session- und Datenaufgabe

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/"];
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration
defaultSessionConfiguration];

// Configure the session here.

NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration];

[[session dataTaskWithURL:url
    completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
{
    // The response object contains the metadata (HTTP headers, status code)

    // The data object contains the response body

    // The error object contains any client-side errors (e.g. connection
    // failures) and, in some cases, may report server-side errors.
    // In general, however, you should detect server-side errors by
    // checking the HTTP status code in the response object.
}] resume];
```

Hintergrundkonfiguration einrichten

So erstellen Sie eine Hintergrundsitzung

```
// Swift:
let mySessionID = "com.example.bgSession"
let bgSessionConfig =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(mySessionID)

let session = NSURLSession(configuration: bgSessionConfig)

// add tasks here

// Objective-C:
NSString *mySessionID = @"com.example.bgSession";
NSURLSessionConfiguration *configuration =
    [NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier: mySessionID];
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration
                        delegate:self]
```

In iOS müssen Sie außerdem die Unterstützung für die Behandlung von Hintergrund-App-Relaunch einrichten. Wenn die

`application:handleEventsForBackgroundURLSession:completionHandler:` Ihrer `application:handleEventsForBackgroundURLSession:completionHandler:` Methode (Objective-C) oder `application(_:handleEventsForBackgroundURLSession:completionHandler:)` aufgerufen wird, bedeutet dies, dass Ihre App im Hintergrund neu gestartet wurde, um die Aktivität in einer Sitzung `application(_:handleEventsForBackgroundURLSession:completionHandler:)` .

In dieser Methode sollten Sie eine neue Sitzung mit dem angegebenen Bezeichner erstellen und diese mit einem Delegaten konfigurieren, um Ereignisse so zu behandeln, wie Sie es normalerweise im Vordergrund tun würden. Außerdem sollten Sie den bereitgestellten Beendigungs-Handler in einem Wörterbuch speichern, wobei die Sitzung als Schlüssel verwendet wird.

Wenn die `NSURLSessionDidFinishEventsForBackgroundNSURLSession:` des Delegaten

`NSURLSessionDidFinishEventsForBackgroundNSURLSession:` (Obj-C) /

`NSURLSessionDidFinishEventsForBackgroundNSURLSession` (Swift), um Ihnen mitzuteilen, dass keine weiteren Ereignisse verarbeitet werden müssen, sollte Ihre

`NSURLSessionDidFinishEventsForBackgroundNSURLSession` den Abschluss-Handler für diese Sitzung suchen. Rufen Sie den Completion-Handler auf und teilen Sie dem Betriebssystem mit, dass Sie keine ausstehende Verarbeitung mehr für die Sitzung haben. (Wenn Sie aus irgendeinem Grund immer noch etwas tun, wenn Sie diesen Delegatenanruf erhalten, warten Sie, bis er fertig ist.) Sobald Sie diese Methode aufrufen, wird die Hintergrundsession sofort ungültig.

Wenn Ihre Anwendung dann eine `application:application:didFinishLaunchingWithOptions:`

empfängt `application:application:didFinishLaunchingWithOptions:` call (wahrscheinlich, dass der Benutzer Ihre `application:application:didFinishLaunchingWithOptions:` den Vordergrund gestellt hat, während Sie Hintergrundereignisse verarbeiten), ist es sicher, eine Hintergrundsession mit derselben Kennung zu erstellen, da die alte Sitzung diese verwendet. Kennung existiert nicht mehr.

Wenn Sie neugierig sind auf Details, wenn Sie eine Hintergrundsession erstellen, tun Sie zwei Dinge:

- Erstellen einer Sitzung in einem externen Dämon (`nsurlsessiond`) zur Abwicklung der Downloads
- Erstellen Sie eine Sitzung in Ihrer App, die über NSXPC mit diesem externen Dämon kommuniziert

Normalerweise ist es gefährlich, zwei Sitzungen mit derselben Sitzungs-ID bei einem einzigen Start der App zu erstellen, da beide versuchen, mit derselben Sitzung im Hintergrund-Daemon zu sprechen. Daher heißt es in der offiziellen Dokumentation, niemals mehrere Sitzungen mit derselben Kennung zu erstellen. Wenn es sich bei der ersten Sitzung jedoch um eine temporäre Sitzung handelt, die als Teil eines `handleEventsForBackgroundNSURLSession` Aufrufs erstellt wurde, ist die Zuordnung zwischen der nun ungültig gewordenen In-App-Sitzung und der Sitzung im Hintergrunddämon nicht mehr vorhanden.

Senden einer POST-Anforderung mit Argumenten mithilfe von `NSURLSession` in Objective-C

Es gibt zwei Möglichkeiten, einen POST-Request-Body zu codieren: URL-Codierung (`application/x-www-form-urlencoded`) und Formulardaten (`Multipart- / Formulardaten`). Ein Großteil des Codes ist ähnlich, aber die Art und Weise, wie Sie die Körperdaten erstellen, ist unterschiedlich.

Senden einer Anfrage mit URL-Kodierung

Egal, ob Sie einen Server für Ihre kleine Anwendung haben oder in einem Team mit einem

kompletten Back-End-Ingenieur arbeiten, Sie möchten mit diesem Server an einem Punkt mit Ihrer iOS-Anwendung sprechen.

Im folgenden Code werden wir eine Folge von Argumenten zusammenstellen, die das Ziel-Server-Skript verwendet, um etwas zu tun, das sich je nach Fall ändert. Zum Beispiel möchten wir die Zeichenfolge senden:

```
name = Brendon & Passwort = abcde
```

Auf dem Server, wenn sich ein Benutzer bei Ihrer Anwendung anmeldet, damit der Server diese Informationen in einer Datenbank speichern kann.

Lass uns anfangen. Sie möchten eine NSURLSession-POST-Anforderung mit dem folgenden Code erstellen.

```
// Create the configuration, which is necessary so we can cancel cacheing amongst other things.
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
// Disables cacheing
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration:defaultConfigObject
delegate:self delegateQueue:[NSOperationQueue mainQueue]];

NSString * scriptURL = [NSString stringWithFormat:@"https://server.io/api/script.php"];
//Converts the URL string to a URL usable by NSURLSession
NSMutableURLRequest * urlRequest = [NSMutableURLRequest requestWithURL:[NSURL
URLWithString:scriptURL]];
NSString * postDataString = [NSString stringWithFormat:@"name=%@&password=%@", [self
nameString], [self URLEncode:passwordString]];
[urlRequest setHTTPMethod:@"POST"];
[urlRequest setHTTPBody:[postDataString dataUsingEncoding:NSUTF8StringEncoding]];

NSURLSessionDataTask * dataTask = [defaultSession dataTaskWithRequest:urlRequest];
// Fire the data task.
[dataTask resume];
```

Der obige Code hat die POST-Anforderung an den Server erstellt und ausgelöst. Denken Sie daran, dass sich die Skript-URL und die POST-Datenzeichenfolge je nach Ihrer Situation ändern. Wenn Sie dies lesen, wissen Sie, womit Sie diese Variablen füllen sollen.

Sie müssen auch eine kleine Methode hinzufügen, die die URL-Kodierung übernimmt:

```
- (NSString *)URLEncode:(NSString *)originalString encoding:(NSStringEncoding)encoding
{
    return (__bridge_transfer NSString *)CFURLCreateStringByAddingPercentEscapes(
        kCFAllocatorDefault,
        (__bridge CFStringRef)originalString,
        NULL,
        CFSTR(":/?#[!$&'()*+,-;="),
        CFStringConvertNSStringEncodingToEncoding(encoding));
}
```

Wenn der Server die Verarbeitung dieser Daten abgeschlossen hat, sendet er eine Rückmeldung

an Ihre iOS-App. Also müssen wir diese Rückkehr bearbeiten, aber wie?

Wir verwenden ereignisgesteuerte Programmierung und verwenden die Delegat-Methoden von NSURLSession. Wenn der Server eine Antwort zurücksendet, werden diese Methoden ausgelöst. Die folgenden 5 Methoden werden während der gesamten ENTIRE-Anforderung jedes Mal ausgelöst, wenn eine gemacht wird:

```
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveResponse:(NSURLResponse *)response
completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler;

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error;

- (void)URLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*))completionHandler;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler;
```

Nachfolgend sehen Sie die oben im Kontext verwendeten Methoden. Jeder ihrer Zwecke ist dank Apple ziemlich selbsterklärend, aber ich habe ihre Verwendung trotzdem kommentiert:

```
// Response handling delegates
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveResponse:(NSURLResponse *)response
completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler{
    // Handler allows us to receive and parse responses from the server
    completionHandler(NSURLSessionResponseAllow);
}

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data{

    // Parse the JSON that came in into an NSDictionary
    NSError * err = nil;
    NSDictionary * jsonDict = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingAllowFragments error:&err];

    if (!err){ // if no error occurred, parse the array of objects as normal
        // Parse the JSON dictionary 'jsonDict' here
    }else{ // an error occurred so we need to let the user know
        // Handle your error here
    }
}

// Error handling delegate
- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error{
    if(error == nil){
        // Download from API was successful
        NSLog(@"Data Network Request Did Complete Successfully.");
    }else{
```

```

    // Describes and logs the error preventing us from receiving a response
    NSLog(@"Error: %@", [error userInfo]);

    // Handle network error, letting the user know what happened.
}
}

// When the session receives a challenge (because of iOS 9 App Transport Security blocking
non-valid SSL certificates) we use the following methods to tell NSURLSession "Chill out, I
can trust me".
// The following is not necessary unless your server is using HTTP, not HTTPS

- (void)NSURLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*))completionHandler{
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential, credential);
        }
    }
}

- (void)NSURLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler{
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential, credential);
        }
    }
}
}

```

So, das war es! Das ist alles Code, den Sie zum Senden, Empfangen und Analysieren einer API-Anforderung in iOS 9 benötigen! Okay ... es war eine Menge Code. Aber wie oben beschrieben, ist es ausfallsicher! Vergewissern Sie sich, dass Sie Fehler immer behandeln, wenn dies oben vorgeschlagen wurde.

Senden einer Anfrage mit Formularverschlüsselung

Die URL-Codierung ist eine weitgehend kompatible Methode zum Codieren beliebiger Daten. Das Hochladen binärer Daten (z. B. Fotos) ist jedoch relativ ineffizient, da jedes Nicht-ASCII-Byte zu einem dreistelligen Code wird. Es unterstützt auch keine Dateianhänge, daher müssen Sie Dateinamen und Dateidaten als separate Felder übergeben.

Angenommen, wir möchten ein Foto so hochladen, dass es effizient ist und tatsächlich auf Server-Seite wie eine Datei aussieht. Eine Möglichkeit, dies zu tun, besteht darin, stattdessen die Formulkodierung zu verwenden. Bearbeiten Sie dazu den Code, der die NSURLSession erstellt, wie folgt:

```
UIImage * imgToSend;
```



```

// 2nd parameter of UIImageJPEGRepresentation represents compression quality. 0 being most
compressed, 1 being the least
// Using 0.4 likely stops us hitting the servers upload limit and costs us less server space
NSData * imageData = UIImageJPEGRepresentation(imgToSend, 0.4f);

// Alternatively, if the photo is on disk, you can retrieve it with
// [NSData dataWithContentsOfURL:...]
```

```

// Set up the body of the POST request.

// This boundary serves as a separator between one form field and the next.
// It must not appear anywhere within the actual data that you intend to
// upload.
NSString * boundary = @"-----14737809831466499882746641449";

// Body of the POST method
NSMutableData * body = [NSMutableData data];

// The body must start with the boundary preceded by two hyphens, followed
// by a carriage return and newline pair.
//
// Notice that we prepend two additional hyphens to the boundary when
// we actually use it as part of the body data.
//
[body appendData:[NSString stringWithFormat:@"\r\n--%@\r\n",boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// This is followed by a series of headers for the first field and then
// TWO CR-LF pairs.
[body appendData:[NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"%tag_name%\r\n\r\n"] dataUsingEncoding:NSUTF8StringEncoding]];

// Next is the actual data for that field (called "tag_name") followed by
// a CR-LF pair, a boundary, and another CR-LF pair.
[body appendData:[strippedCompanyName dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[NSString stringWithFormat:@"\r\n--%@\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Encode the filename and image data as the "userfile" CGI parameter.
// This is similar to the previous field, except that it is being sent
// as an actual file attachment rather than a blob of data, which means
// it has both a filename and the actual file contents.
//
// IMPORTANT: The filename MUST be plain ASCII (and if encoded like this,
// must not include quotation marks in the filename).
//
NSString * picFileName = [NSString stringWithFormat:@"photoName"];
NSString * appendDataString = [NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"userfile\"; filename=\"%%.jpg\"\r\n", picFileName];
[body appendData:[appendDataString dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[@"Content-Type: application/octet-stream\r\n\r\n"
dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[NSData dataWithData:imageData]];

// Close the request body with one last boundary with two
// additional hyphens prepended **and** two additional hyphens appended.
[body appendData:[NSString stringWithFormat:@"\r\n--%@\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Create the session

```

```
// We can use the delegate to track upload progress and disable cacheing
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration: defaultConfigObject
delegate: self delegateQueue: [NSOperationQueue mainQueue]];

// Data uploading task.
NSURL * url = [NSURL URLWithString:@"https://server.io/api/script.php"];
NSMutableURLRequest * request = [NSMutableURLRequest requestWithURL:url];
NSString * contentType = [NSString stringWithFormat:@"multipart/form-data;
boundary=%@",boundary];
[request addValue:contentType forHTTPHeaderField:@"Content-Type"];
request.HTTPMethod = @"POST";
request.HTTPBody = body;
NSURLSessionDataTask * uploadTask = [defaultSession dataTaskWithRequest:request];
[uploadTask resume];
```

Dadurch wird die `NSURLSession`-Anforderung wie zuvor erstellt und ausgelöst, und die Delegat-Methoden verhalten sich folglich genauso. Stellen Sie sicher, dass das Skript, an das das Bild gesendet wird (an der URL in der Variablen `url`), ein Bild erwartet und es korrekt analysieren kann.

NSURLSession online lesen: <https://riptutorial.com/de/ios/topic/2009/nsurlsession>

Kapitel 121: NSUserActivity

Einführung

Ein `NSUserActivity`-Objekt kann verwendet werden, um wichtige Ereignisse in einer App mit dem System zu koordinieren. Es ist die Basis für die [Übergabe](#) zwischen verschiedenen Geräten, auf denen iOS und macOS ausgeführt wird. Darüber hinaus kann es auch verwendet werden, um die öffentliche Indexierung zu verbessern und Spotlight Search-Ergebnisse für eine App zu erweitern oder zu erstellen. Ab iOS 10 kann es auch verwendet werden, um Interaktionen zwischen Ihrer App und Siri mithilfe von SiriKit zu koordinieren.

Bemerkungen

Aktivitätsarten

Unterstützte Aktivitätstypen müssen in der `Info.plist` Datei Ihrer App unter dem Schlüssel `NSUserActivityTypes` . Aktivitäten sind an Ihre Entwicklerteam-ID gebunden. Dies bedeutet, dass die Aktivitätskoordination zwischen Apps mit derselben Team-ID eingeschränkt ist (z. B. "Safari" konnte keine Handoff-Aktivität von "Chrome" akzeptieren oder umgekehrt).

Aktive Aktivität einstellen / kündigen

Durch das Markieren einer Aktivität als aktuelle Aktivität wird die `becomeCurrent` für Handoff- oder Spotlight-Indizierung verfügbar. Es kann jeweils nur eine Aktivität aktuell sein. Sie können eine Aktivität als inaktiv markieren, ohne sie ungültig zu machen, indem Sie `resignCurrent` aufrufen.

Wenn Sie eine Aktivität für `invalidate` , wird dieselbe Instanz möglicherweise nicht mehr aktuell gemacht.

Kennzeichnen Sie eine Aktivität nicht als aktuell, wenn Sie sie für [SiriKit bereitstellen](#) .

Suchindizierung

Aktivitäten dürfen **nicht** als allgemeiner Indexierungsmechanismus in Ihrer App verwendet werden. Sie sollten stattdessen nur als Antwort auf vom Benutzer initiierte Aktionen verwendet werden. Verwenden Sie `CoreSpotlight`, um alle Inhalte in Ihrer App zu indizieren.

Examples

NSUserActivity erstellen

Um ein `NSUserActivity` Objekt erstellen zu `NSUserActivity` , muss Ihre App die unterstützten Aktivitätenarten in ihrer `Info.plist` Datei `Info.plist` . Unterstützte Aktivitäten werden von Ihrer Anwendung definiert und sollten eindeutig sein. Eine Aktivität wird unter Verwendung eines Namensschemas für eine umgekehrte Domäne definiert (z. B. `"com.companyName.productName.activityName"`). So könnte ein Eintrag in Ihrer `Info.plist` aussehen:

Schlüssel	Wert
<code>NSUserActivityTypes</code>	[Array]
- item0	<code>com.companyName.productName.activityName01</code>
- Gegenstand 1	<code>com.companyName.productName.activityName02</code>

Nachdem Sie alle unterstützten Aktivitätstypen definiert haben, können Sie damit beginnen, auf diese zuzugreifen und sie im Code Ihrer Anwendung zu verwenden.

Um ein `NSUserActivity` Objekt zu erstellen, müssen Sie folgende `NSUserActivity`

```
// Initialize the activity object and set its type from one of the ones specified in your
app's plist
NSUserActivity *currentActivity = [[NSUserActivity alloc]
initWithActivityType:@"com.companyName.productName.activityName01"];

// Set the title of the activity.
// This title may be displayed to the user, so make sure it is localized and human-readable
currentActivity.title = @"Current Activity";

// Configure additional properties like userInfo which will be included in the activity
currentActivity.userInfo = @{@"informationKey" : @"value"};

// Configure the activity so the system knows what may be done with it
// It is important that you only set YES to tasks that your application supports
// In this example, we will only enable the activity for use with Handoff
[currentActivity setEligibleForHandoff:YES];
[currentActivity setEligibleForSearch:NO]; // Defaults to NO
[currentActivity setEligibleForPublicIndexing:NO]; // Defaults to NO

// Set this activity as the current user activity
// Only one activity may be current at a time on a device. Calling this method invalidates any
other current activities.
[currentActivity becomeCurrent];
```

Danach sollte die oben genannte Aktivität für Handoff verfügbar sein (obwohl mehr Arbeit erforderlich ist, um die "Handoff" richtig abzuwickeln).

NSUserActivity online lesen: <https://riptutorial.com/de/ios/topic/10716/nsuseractivity>

Kapitel 122: UserDefaults

Syntax

- `UserDefaults.standard.set(dic, forKey: "LoginSession") //Save value inside userdefaults`
 - `UserDefaults.standard.object(forKey: "LoginSession") as? [String:AnyObject] ?? [:]`
`//Get value from UserDefaults`

Bemerkungen

NSUserDefaults, die zum Speichern aller Datentypen verwendet werden, und der Wert kann an einer beliebigen Stelle in der App-Klasse abgerufen werden. [NSUserDefaults](#)

Examples

Einstellwerte

Um einen Wert in `NSUserDefaults`, können Sie die folgenden Funktionen verwenden:

Schnell <3

```
setBool(_:forKey:)
setFloat(_:forKey:)
setInteger(_:forKey:)
setObject(_:forKey:)
setDouble(_:forKey:)
setURL(_:forKey:)
```

Swift 3

In Swift 3 die Namen der Funktion geändert `set` von `insted set` durch den Typ `folloed`.

```
set(_:forKey:)
```

Ziel c

```
-(void)setBool:(BOOL)value forKey:(nonnull NSString *)defaultName;
-(void)setFloat:(float)value forKey:(nonnull NSString *)defaultName;
-(void)setInteger:(NSInteger)value forKey:(nonnull NSString *)defaultName;
-(void)setObject:(nullable id)value forKey:(nonnull NSString *)defaultName;
-(void)setDouble:(double)value forKey:(nonnull NSString *)defaultName;
-(void)setURL:(nullable NSURL *)value forKey:(nonnull NSString *)defaultName;
```

Ein Beispiel wäre:

Schnell <3

```
NSUserDefaults.standardUserDefaults.setObject("Netherlands", forKey: "HomeCountry")
```

Swift 3

```
UserDefaults.standard.set("Netherlands", forKey: "HomeCountry")
```

Ziel c

```
[[NSUserDefaults standardUserDefaults] setObject:@"Netherlands" forKey:@"HomeCountry"];
```

Benutzerdefinierte Objekte

Um benutzerdefinierte Objekte in den "NSUserDefaults" zu speichern, müssen Sie Ihre CustomClass mit dem Protokoll "NSCoding" bestätigen. Sie müssen die folgenden Methoden implementieren:

Schnell

```
public func encodeWithCoder(aCoder: NSCoder) {
    aCoder.encodeObject(name, forKey:"name")
    aCoder.encodeObject(unitId, forKey: "unitId")
}

required public init(coder aDecoder: NSCoder) {
    super.init()
    name = aDecoder.decodeObjectForKey("name") as? String
    unitId = aDecoder.decodeIntegerForKey("unitId") as? NSInteger
}
```

Ziel c

```
- (id)initWithCoder:(NSCoder *)coder {
    self = [super init];
    if (self) {
        name = [coder decodeObjectForKey:@"name"];
        unitId = [coder decodeIntegerForKey:@"unitId"];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder*)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeInteger:unitId forKey:@"unitId"];
}
```

```
}
```

Standardwerte abrufen

Um einen Wert in `NSUserDefaults` zu erhalten, können Sie die folgenden Funktionen verwenden:

Schnell

```
arrayForKey(_:)  
boolForKey(_:)  
dataForKey(_:)  
dictionaryForKey(_:)  
floatForKey(_:)  
integerForKey(_:)  
objectForKey(_:)  
stringArrayForKey(_:)  
stringForKey(_:)  
doubleForKey(_:)  
URLForKey(_:)
```

Ziel c

```
-(nullable NSArray *)arrayForKey:(nonnull NSString *)defaultName;  
-(BOOL)boolForKey:(nonnull NSString *)defaultName;  
-(nullable NSData *)dataForKey:(nonnull NSString *)defaultName;  
-(nullable NSDictionary<NSString *, id> *)dictionaryForKey:(nonnull NSString *)defaultName;  
-(float)floatForKey:(nonnull NSString *)defaultName;  
-(NSInteger)integerForKey:(nonnull NSString *)defaultName;  
-(nullable id)objectForKey:(nonnull NSString *)key;  
-(nullable NSArray<NSString *> *)stringArrayForKey:(nonnull NSString *)defaultName;  
-(nullable NSString *)stringForKey:(nonnull NSString *)defaultName;  
-(double)doubleForKey:(nonnull NSString *)defaultName;  
-(nullable NSURL *)URLForKey:(nonnull NSString *)defaultName;
```

Ein Beispiel wäre:

Schnell

```
let homeCountry = NSUserDefaults.standardUserDefaults().stringForKey("HomeCountry")
```

Ziel c

```
NSString *homeCountry = [[NSUserDefaults standardUserDefaults] stringForKey:@"HomeCountry"];
```

Werte speichern

`NSUserDefaults` werden vom System regelmäßig auf die Festplatte geschrieben. Es kann jedoch vorkommen, dass Ihre Änderungen sofort gespeichert werden sollen, z. Dies geschieht durch

Aufruf von `synchronize` .

Schnell

```
NSUserDefaults.standardUserDefaults().synchronize()
```

Ziel c

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Verwenden Sie Manager zum Speichern und Lesen von Daten

Sie können die `NSUserDefaults` Methoden zwar überall verwenden, es kann jedoch manchmal besser sein, einen Manager zu definieren, der `NSUserDefaults` speichert und von `NSUserDefaults` liest, und diesen Manager dann zum Lesen oder Schreiben Ihrer Daten zu verwenden.

Angenommen, wir möchten die Punktzahl eines Benutzers in `NSUserDefaults` . Wir können eine Klasse wie die folgende erstellen, die zwei Methoden hat: `setHighScore` und `highScore` . Erstellen Sie eine Instanz dieser Klasse, wenn Sie auf die Highscores zugreifen möchten.

Schnell

```
public class ScoreManager: NSObject {

    let highScoreDefaultKey = "HighScoreDefaultKey"

    var highScore = {
        set {
            // This method includes your implementation for saving the high score
            // You can use NSUserDefaults or any other data store like CoreData or
            // SQLite etc.

            NSUserDefaults.standardUserDefaults().setInteger(newValue, forKey:
highScoreDefaultKey)
            NSUserDefaults.standardUserDefaults().synchronize()
        }
        get {
            //This method includes your implementation for reading the high score

            let score =
NSUserDefaults.standardUserDefaults().objectForKey(highScoreDefaultKey)

            if (score != nil) {
                return score.integerValue;
            } else {
                //No high score available, so return -1
                return -1;
            }
        }
    }
}
```


Ziel c

```
#import "ScoreManager.h"

#define HIGHSCORE_KEY @"highScore"

@implementation ScoreManager

- (void)setHighScore:(NSInteger) highScore {
    // This method includes your implementation for saving the high score
    // You can use UserDefaults or any other data store like CoreData or
    // SQLite etc.

    [[NSUserDefaults standardUserDefaults] setInteger:highScore forKey:HIGHSCORE_KEY];
    [[NSUserDefaults standardUserDefaults] synchronize];
}

- (NSInteger)highScore
{
    //This method includes your implementation for reading the high score

    NSNumber *highScore = [[NSUserDefaults standardUserDefaults] objectForKey:HIGHSCORE_KEY];
    if (highScore) {
        return highScore.integerValue;
    }else
    {
        //No high score available, so return -1

        return -1;
    }
}

@end
```

Die Vorteile sind:

1. Die Implementierung Ihres Lese- und Schreibprozesses ist nur an einem Ort und Sie können ihn jederzeit ändern (z. B. von `NSUserDefaults` zu Core Data wechseln), ohne sich Sorgen zu machen, alle Orte zu ändern, an denen Sie mit dem Highscore arbeiten.
2. Rufen Sie einfach nur eine Methode auf, wenn Sie auf die Partitur zugreifen oder sie schreiben möchten.
3. Debuggen Sie es einfach, wenn Sie einen Fehler oder ähnliches sehen.

Hinweis

Wenn Sie sich Sorgen um die Synchronisierung machen, ist es besser, eine Singleton-Klasse zu verwenden, die die Synchronisation verwaltet.

NSUserDefaults löschen

Schnell

```
let bundleIdentifier = NSBundle.mainBundle().bundleIdentifier()

NSUserDefaults.standardUserDefaults().removePersistentDomainForName(bundleIdentifier)
```

Ziel c

```
NSString *bundleIdentifier = [[NSBundle mainBundle] bundleIdentifier];

[[NSUserDefaults standardUserDefaults] removePersistentDomainForName: bundleIdentifier];
```

UserDefaults wird in Swift 3 verwendet

Jede Anwendung musste zum Speichern von Benutzersitzungen oder benutzerbezogenen Details in der Anwendung in UserDefaults verwendet werden. Daher haben wir die gesamte Logik innerhalb einer Klasse für die Verwaltung von UserDefaults verbessert.

Swift 3

```
import Foundation

public struct Session {

    fileprivate static let defaults = UserDefaults.standard

    enum userValues: String {
        case auth_token
        case email
        case fname
        case mobile
        case title
        case userId
        case userType
        case OTP
        case isApproved
    }

    //MARK: - Getting here User Details
    static func getSessionDetails() -> [String:AnyObject]? {
        let dictionary = defaults.object(forKey: "LoginSession") as? [String:AnyObject]
        return dictionary
    }

    //MARK: - Saving Device Token
    static func saveDeviceToken(_ token:String){
        guard (getSessionDetails() ?? "").isEmpty else {
            return
        }
        defaults.removeObject(forKey: "deviceToken")
        defaults.set(token, forKey: "deviceToken")
        defaults.synchronize()
    }
}
```

```

//MARK: - Getting Token here
static func gettingDeviceToken()->String?{
    let token = defaults.object(forKey: "deviceToken") as? String
    if token == nil{
        return ""
    }else{ return token}
}

//MARK: - Setting here User Details
static func setUserSessionDetails(_ dic :[String : AnyObject]){
    defaults.removeObject(forKey: "LoginSession")
    defaults.set(dic, forKey: "LoginSession")
    defaults.synchronize()
}

//MARK:- Removing here all Default Values
static func userSessionLogout(){
    //Set Activity
    defaults.removeObject(forKey: "LoginSession")
    defaults.synchronize()
}

//MARK: - Get value from session here
static func getUserValues(value: userValues) -> String? {
    let dic = getUserSessionDetails() ?? [:]
    guard let value = dic[value.rawValue] else{
        return ""
    }
    return value as? String
}
}
}

```

Verwendung der UserDefaults-Klasse

```

//Saving user Details
Session.setUserSessionDetails(json ?? [:])

//Retriving user Details
let userId = Session.getUserValues(value: .userId) ?? ""

```

NSUserDefaults online lesen: <https://riptutorial.com/de/ios/topic/3150/nsuserdefaults>

Kapitel 123: Objective-C Zugeordnete Objekte

Einführung

Zunächst in iOS 3.1 als Teil der Objective-C-Laufzeit eingeführt, bieten verknüpfte Objekte eine Möglichkeit, Instanzvariablen zu einem vorhandenen Klassenobjekt hinzuzufügen (ohne Unterklassifizierung).

Dies bedeutet, dass Sie jedes Objekt ohne Unterklassifizierung an jedes andere Objekt anhängen können.

Syntax

- `void objc_setAssociatedObject (id-Objekt, void * key, id-Wert, objc_AssociationPolicy-Richtlinie)`
- `id objc_getAssociatedObject (id-Objekt, ungültiger * Schlüssel)`
- `void objc_removeAssociatedObjects (ID-Objekt)`

Parameter

Param	Einzelheiten
Objekt	Das vorhandene Objekt, das Sie ändern möchten
Schlüssel	Dies kann grundsätzlich jeder Zeiger sein, der eine konstante Speicheradresse hat, aber es ist eine gute Praxis, hier eine berechnete Eigenschaft (Getter) zu verwenden.
Wert	Das Objekt, das Sie hinzufügen möchten
Politik	Die Speicherrichtlinie für diesen neuen <code>value</code> dh sollte er beibehalten / zugewiesen, kopiert usw. werden, genau wie bei jeder anderen Eigenschaft, die Sie deklarieren

Bemerkungen

Weitere Details hier:

[NSHipster](#)

[@kostiakoval](#)

[Kingscocoa](#)

Examples

Beispiel für ein assoziiertes Basisobjekt

Nehmen wir an, wir müssen Nectring-Objekt zu `SomeClass` (wir können keine Unterklasse bilden).

In diesem Beispiel erstellen wir ein zugeordnetes Objekt nicht nur, sondern hüllen es in eine Kategorie ein, um die Ordentlichkeit zu erhöhen

```
#import <objc/runtime.h>

@interface SomeClass (MyCategory)
// This is the property wrapping the associated object. below we implement the setter and
getter which actually utilize the object association
@property (nonatomic, retain) NSString *associated;
@end

@implementation SomeClass (MyCategory)

- (void)setAssociated:(NSString *)object {
    objc_setAssociatedObject(self, @selector(associated), object,
                            OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (NSString *)associated {
    return objc_getAssociatedObject(self, @selector(associated));
}
```

Nun wäre es so einfach, die Immobilie zu nutzen

```
SomeClass *instance = [SomeClass alloc] init];
instance.associated = @"this property is an associated object under the hood";
```

Objective-C Zugeordnete Objekte online lesen: <https://riptutorial.com/de/ios/topic/9102/objective-c-zugeordnete-objekte>

Kapitel 124: Online-Bilder zwischenspeichern

Examples

AlamofireImage

Online-Bilder mit AlamofireImage . Es funktioniert auf Alamofire in Swift. Installieren Sie AlamofireImage mit cocoapods

```
pod 'AlamofireImage', '~> 3.1'
```

Konfiguration:

1. Importieren Sie AlamofireImage und Alamofire
2. Setup den Image-Cache: `let imageCache = AutoPurgingImageCache(memoryCapacity: 111_111_111, preferredMemoryUsageAfterPurge: 90_000_000)`
3. Eine Anfrage machen und das Bild zum Cache hinzufügen:

```
Alamofire.request(self.nameUrl[i]).responseImage { response in
    if response.result.value != nil {
        let image = UIImage(data: response.data!, scale: 1.0)!
        imageCache.add(image, withIdentifier: self.nameUrl[i])
    }
}
```

4. Abrufen von Bildern aus dem Cache:

```
if let image = imageCache.image(withIdentifier: self.nameUrl[self.a])
{
    self.localImageView.image = image
}
```

Für weitere Informationen folgen Sie [diesem Link](#)

Online-Bilder zwischenspeichern online lesen: <https://riptutorial.com/de/ios/topic/9450/online-bilder-zwischenspeichern>

Kapitel 125: OpenGL

Einführung

OpenGL ES ist eine Grafikkbibliothek, die iOS für das 3D-Rendering verwendet.

Examples

Beispielprojekt

Ein [Beispielprojekt \(Git-Repo\)](#) , das als Ausgangspunkt für das 3D-Rendering verwendet werden kann. Der Code zum Einrichten von OpenGL und der Shader ist ziemlich langwierig und passt daher nicht in dieses Beispielformat. Spätere Teile davon können in separaten Beispielen gezeigt werden, in denen detailliert beschrieben wird, was genau mit jedem Teil des Codes passiert.

OpenGL online lesen: <https://riptutorial.com/de/ios/topic/9324/opengl>

Kapitel 126: Parallelität

Einführung

Verwandte Themen: [Grand Central Dispatch](#)

Syntax

- `dispatch_async` - Führt einen Codeblock in einer separaten Warteschlange aus und stoppt die aktuelle Warteschlange nicht. Wenn sich die Warteschlange in einem anderen Thread befindet als der, für den `dispatch_async` aufgerufen wurde, wird der Code im Block ausgeführt, während der Code nach dem `dispatch_async` ebenfalls ausgeführt wird
- `dispatch_sync` - Führt einen Codeblock in einer separaten Warteschlange und die aktuelle Warteschlange *nicht* zu stoppen. Wenn sich die Warteschlange in einem anderen Thread befindet als der, für den `dispatch_async` aufgerufen wurde, wird der Code im Block ausgeführt, und die Ausführung des Threads, in dem die Methode aufgerufen wurde, wird erst wieder aufgenommen, wenn sie abgeschlossen ist

Parameter

Warteschlange	<p>Die Warteschlange, in der der Code im Dispatch-Block ausgeführt wird. Eine <i>Warteschlange</i> ist (aber nicht genau wie ein Thread). Code in verschiedenen Warteschlangen kann parallel ausgeführt werden. Verwenden Sie <code>dispatch_get_main_queue</code>, um die Warteschlange für den Hauptthread <code>dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)</code> Zum Erstellen einer neuen Warteschlange, die wiederum einen neuen Thread erstellt, verwenden Sie <code>dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)</code>. Der erste Parameter ist der Name der Warteschlange, der im Debugger angezeigt wird, wenn Sie anhalten, während der Block noch läuft. Der zweite Parameter ist unwichtig, es sei denn, Sie möchten dieselbe Warteschlange für mehrere <code>dispatch_async</code> oder <code>dispatch_sync</code> Aufrufe verwenden. Es beschreibt, was passiert, wenn ein anderer Block in dieselbe Warteschlange gestellt wird. <code>DISPATCH_QUEUE_CONCURRENT</code> bewirkt, dass beide Blöcke gleichzeitig ausgeführt werden, während <code>DISPATCH_QUEUE_SERIAL</code> den zweiten Block auf den Abschluss des ersten Blocks warten lässt</p>
Block	<p>Code in diesem Block wird in der <code>queue</code>. Geben Sie hier den Code ein, den Sie in der separaten Warteschlange ausführen möchten. Ein hilfreicher Tipp: Wenn Sie dies in Xcode schreiben und das Blockargument blau umrahmt ist, doppelklicken Sie auf das Argument, und Xcode erstellt automatisch einen leeren Block (dies gilt für alle Blockargumente in einer Funktion oder Methode).</p>

Bemerkungen

Wenn Sie etwas in einem separaten Thread tun, was bei der Verwendung von Warteschlangen der Fall ist, ist es wichtig, die Thread-Sicherheit zu gewährleisten. Einige Methoden, insbesondere für `UIView`, funktionieren möglicherweise nicht und stürzen nicht in anderen Threads als dem Haupt-Thread ab. Stellen Sie außerdem sicher, dass Sie nichts (Variablen, Eigenschaften usw.) ändern, das auch im Hauptthread verwendet wird, es sei denn, Sie berücksichtigen diese Änderung

Examples

Gleichzeitiger Ausführen von Code - Ausführen von Code, während anderer Code ausgeführt wird

Angenommen, Sie möchten eine Aktion ausführen (in diesem Fall "Foo" protokollieren), während Sie etwas anderes tun ("Balken" protokollieren). Wenn Sie keine Parallelität verwenden, wird eine dieser Aktionen normalerweise vollständig ausgeführt, und der andere Lauf wird erst ausgeführt, wenn er vollständig abgeschlossen ist. Mit Parallelität können Sie jedoch beide Aktionen gleichzeitig ausführen:

```
dispatch_async(dispatch_queue_create("Foo", DISPATCH_QUEUE_CONCURRENT), ^{
    for (int i = 0; i < 100; i++) {
        NSLog(@"Foo");
        usleep(100000);
    }
});

for (int i = 0; i < 100; i++) {
    NSLog(@"Bar");
    usleep(50000);
}
```

Dadurch wird "Foo" 100-mal protokolliert und jedes Mal, wenn es protokolliert, für 100 ms angehalten. Dies geschieht jedoch in einem separaten Thread. Während der `Foo` wird "Bar" gleichzeitig im 50-ms-Intervall protokolliert. Im Idealfall sollte eine Ausgabe mit "Foo" s und "Bars" gemischt angezeigt werden

Ausführung im Hauptthread

Wenn Aufgaben asynchron ausgeführt werden, muss normalerweise sichergestellt werden, dass im Haupt-Thread Code ausgeführt wird. Beispielsweise möchten Sie eine REST-API asynchron schlagen, das Ergebnis jedoch in einem UILabel auf dem Bildschirm anzeigen. Bevor Sie das UILabel aktualisieren, müssen Sie sicherstellen, dass Ihr Code im Hauptthread ausgeführt wird:

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    //Perform expensive tasks
    //...

    //Now before updating the UI, ensure we are back on the main thread
```

```

dispatch_async(dispatch_get_main_queue(), ^{
    label.text = //....
});
}

```

Wenn Sie Ansichten auf dem Bildschirm aktualisieren, stellen Sie immer sicher, dass Sie dies im Haupt-Thread tun, da andernfalls undefiniertes Verhalten auftreten kann.

Versandgruppe - Warten auf andere Threads abgeschlossen.

```

dispatch_group_t preapreWaitingGroup = dispatch_group_create();

dispatch_group_enter(preapreWaitingGroup);
[self doAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    // Notify that this task has been completed.
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_enter(preapreWaitingGroup);
[self doOtherAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_notify(preapreWaitingGroup, dispatch_get_main_queue(), ^{
    // This block will be executed once all above threads completed and call
    dispatch_group_leave
    NSLog(@"Prepare completed. I'm readyyyy");
});

```

Update 1. Swift 3 Version.

```

let prepareGroup = DispatchGroup()
prepareGroup.enter()
doAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.enter()
doOtherAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.notify(queue: DispatchQueue.main) {
    // This block will be executed once all above threads completed and call
    dispatch_group_leave
    print("Prepare completed. I'm readyyyy")
}

```

Parallelität online lesen: <https://riptutorial.com/de/ios/topic/1090/parallelitat>

Kapitel 127: PDF-Erstellung in iOS

Examples

PDF erzeugen

```
UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 612, 792), nil);

[self drawText];

UIGraphicsEndPDFContext();
```

Dateiname ist die Dokumentdatei, an die Sie anhängen oder anhängen möchten

```
NSString* temporaryFile = @"firstIOS.PDF";
NSArray *arrayPaths =
    NSSearchPathForDirectoriesInDomains(
        NSDocumentDirectory,
        NSUserDomainMask,
        YES);

NSString *path = [arrayPaths objectAtIndex:0];

NSString* fileName = [path stringByAppendingPathComponent:fileName];
```

Wo DrawText ist

```
(void)drawText
{
    NSString* textToDraw = @"Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown printer took a galley of type and scrambled it to make a type specimen book.";

    CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

    CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);

    CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);

    CGRect frameRect = CGRectMake(0, 0, 300, 100);

    CGMutablePathRef framePath = CGPathCreateMutable();

    CGPathAddRect(framePath, NULL, frameRect);

    CFRange currentRange = CFRangeMake(0, 0);

    CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
    NULL);
    CGPathRelease(framePath);

    CGContextRef currentContext = UIGraphicsGetCurrentContext();
```

```
CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);

CGContextTranslateCTM(currentContext, 0, 450);

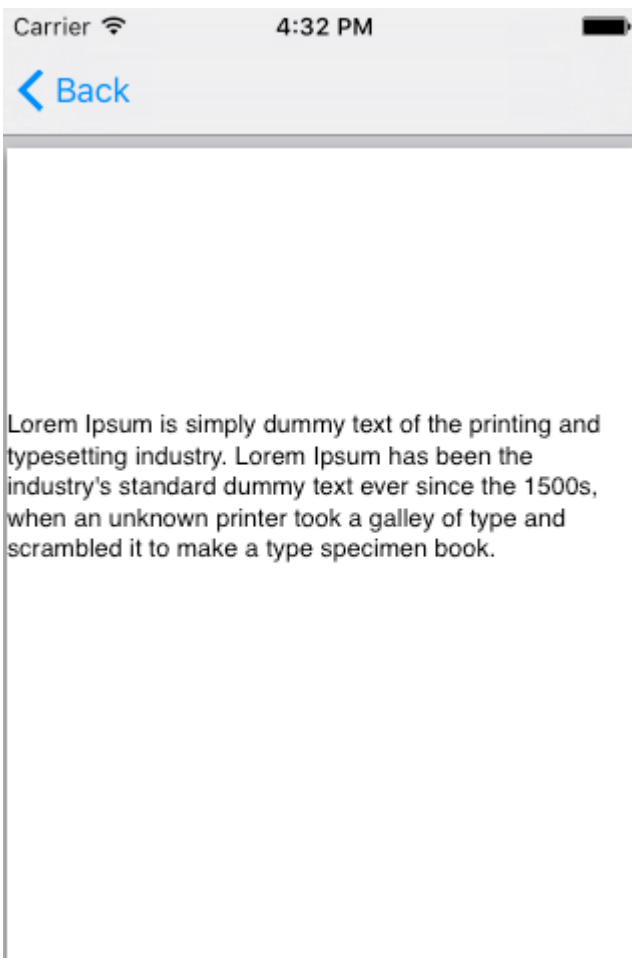
CGContextScaleCTM(currentContext, 2, -2);

CTFrameDraw(frameRef, currentContext);

CFRelease(frameRef);

CFRelease(stringRef);

CFRelease(framesetter);
}
```



PDF anzeigen

```
NSString* fileName = @"firstIOS.PDF";

NSArray *arrayPaths =
NSSearchPathForDirectoriesInDomains(
    NSDocumentDirectory,
    NSUserDomainMask,
    YES);
```

```

NSString *path = [arrayPaths objectAtIndex:0];

NSString* pdfFileName = [path stringByAppendingPathComponent:fileName];

UIWebView* webView = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

NSURL *url = [NSURL fileURLWithPath:pdfFileName];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

[webView setScalesPageToFit:YES];

[webView loadRequest:request];

[self.view addSubview:webView];

```

Mehrseitiges PDF

```

 UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsEndPDFContext();

```

Erstellen Sie ein PDF aus einem beliebigen in UIWebView geladenen Microsoft-Dokument

```
#define kPaperSizeA4 CGSizeMake(595.2, 841.8)
```

Implementieren Sie zunächst das UIPrintPageRenderer-Protokoll

```

@interface UIPrintPageRenderer (PDF)

- (NSData*) printToPDF;

@end

@implementation UIPrintPageRenderer (PDF)

- (NSData*) printToPDF
{
    NSMutableData *pdfData = [NSMutableData data];
    UIGraphicsBeginPDFContextToData(pdfData, self.paperRect, nil);
    [self prepareForDrawingPages:NSMakeRange(0, self.numberOfPages)];
    CGRect bounds = UIGraphicsGetPDFContextBounds();
    for (int i = 0; i < self.numberOfPages; i++)
    {
        UIGraphicsBeginPDFPage();
        [self drawPageAtIndex:i inRect:bounds];
    }
    UIGraphicsEndPDFContext();
}

```

```
    return pdfData;
}
@end
```

Rufen Sie anschließend die Methode unten auf, nachdem das Dokument in `UIWebView`

```
-(void)createPDF:(UIWebView *)webView {

    UIPrintPageRenderer *render = [[UIPrintPageRenderer alloc] init];
    [render addPrintFormatter:webView.viewPrintFormatter startingAtPageAtIndex:0];

    float padding = 10.0f;
    CGRect paperRect = CGRectMake(0, 0, kPaperSizeA4.width, kPaperSizeA4.height);
    CGRect printableRect = CGRectMake(padding, padding, kPaperSizeA4.width-(padding * 2),
    kPaperSizeA4.height-(padding * 2));

    [render setValue:[NSValue valueWithCGRect:paperRect] forKey:@"paperRect"];
    [render setValue:[NSValue valueWithCGRect:printableRect] forKey:@"printableRect"];

    NSData *pdfData = [render printToPDF];

    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{

        if (pdfData) {
            [pdfData writeToFile:directoryPath atomically: YES];
        }
        else
        {
            NSLog(@"PDF couldnot be created");
        }
    });}
};}
```

PDF-Erstellung in iOS online lesen: <https://riptutorial.com/de/ios/topic/2416/pdf-erstellung-in-ios>

Kapitel 128: plist iOS

Einführung

Plist wird zum Speichern von Daten in der iOS-App verwendet. Speichern Sie die Daten in Form von Arrays und Wörterbüchern. In plist können wir Daten speichern als: 1. Statische Daten, die in App verwendet werden sollen. 2. Daten, die vom Server kommen.

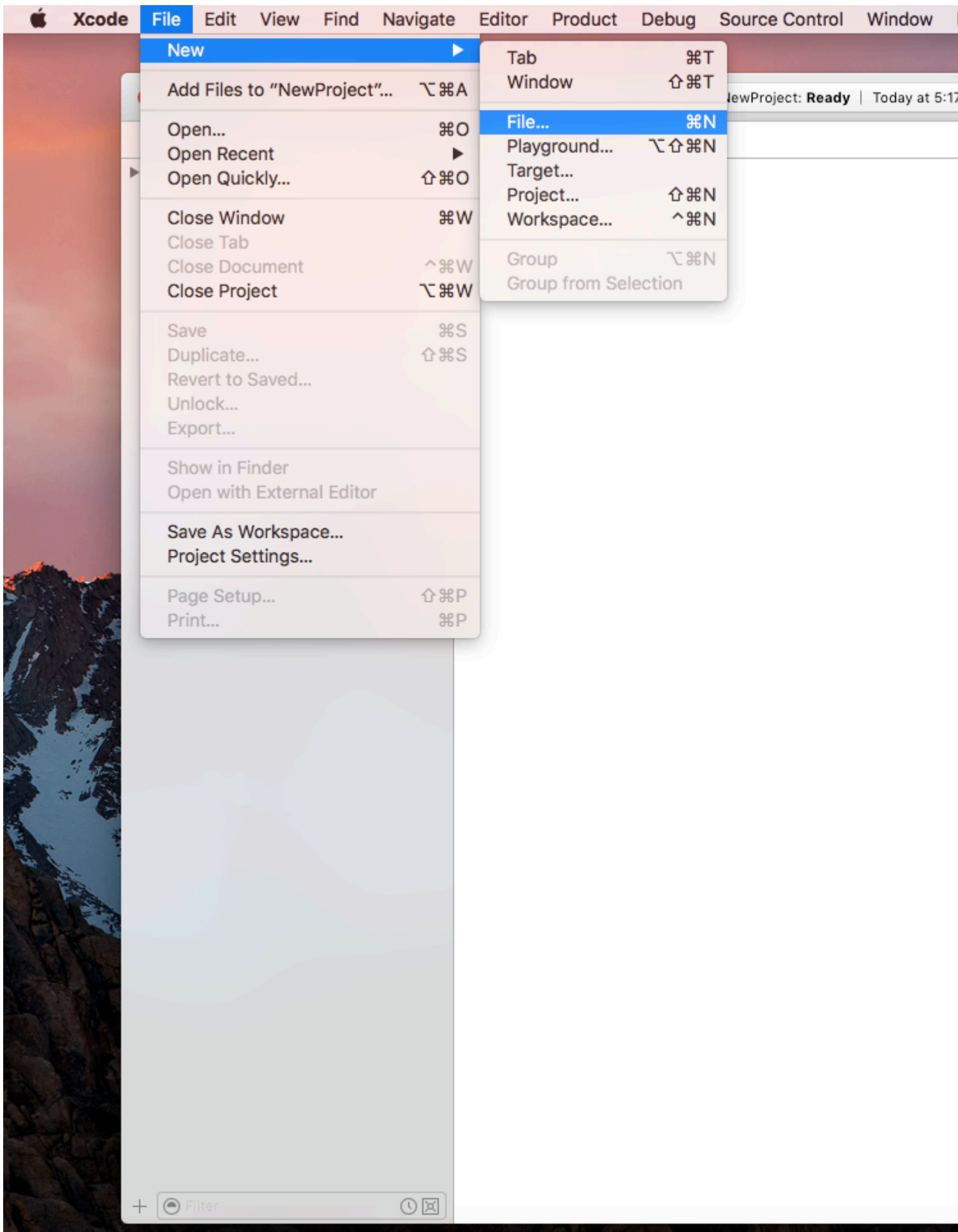
Examples

Beispiel:

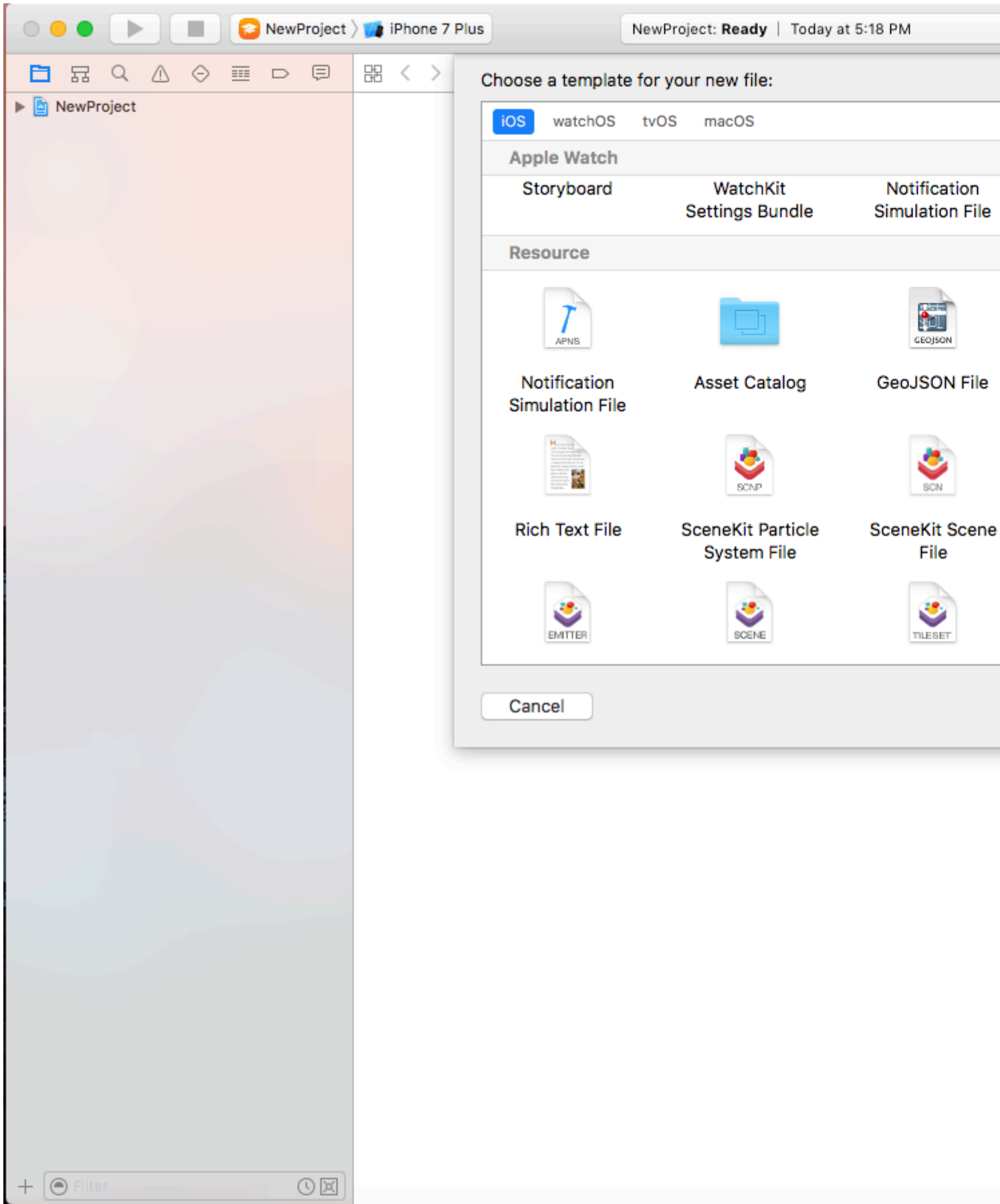
1. Statische Daten zur Verwendung in App.

Um statische Daten in plist zu speichern, gehen Sie folgendermaßen vor:

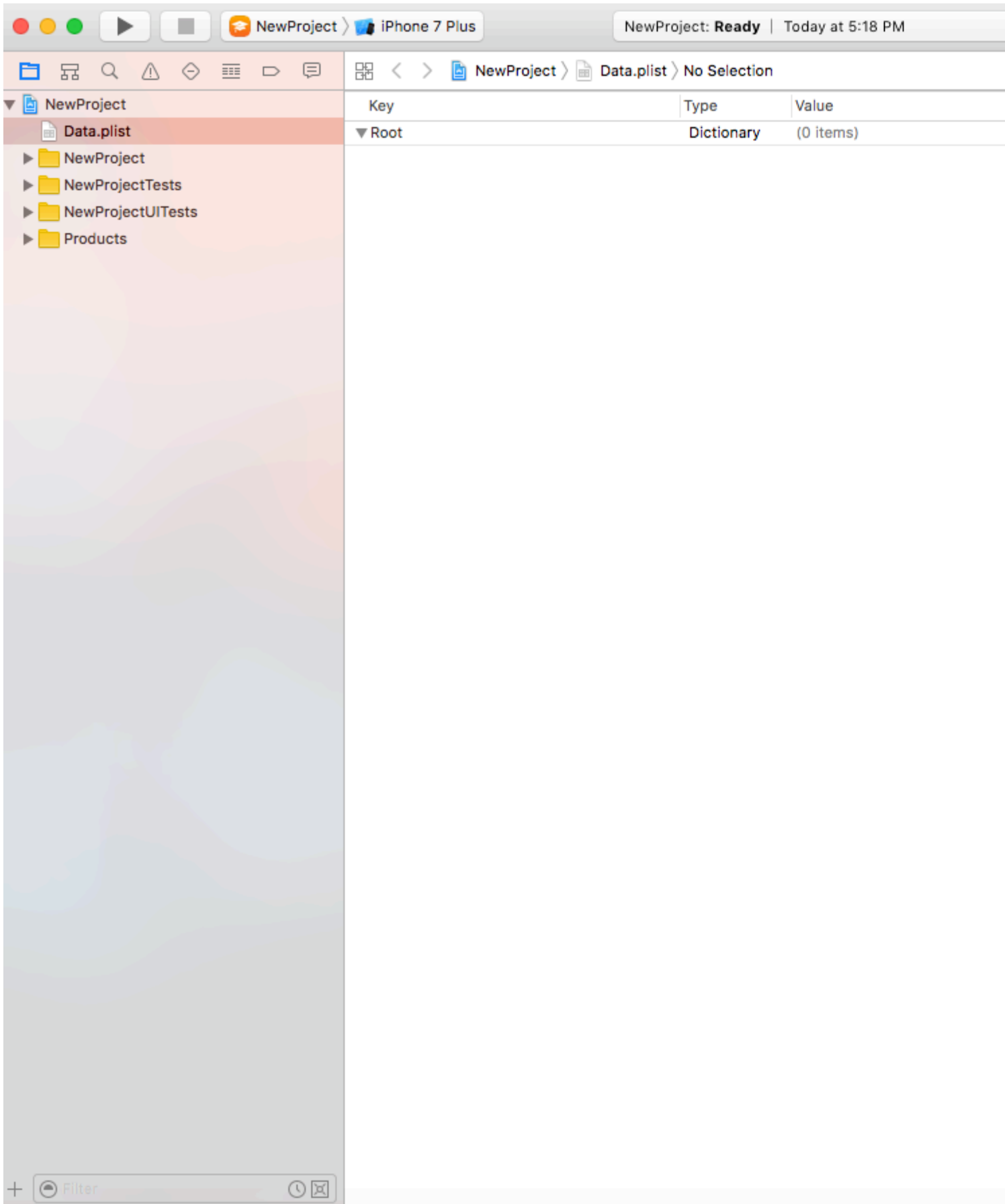
a) Fügen Sie eine neue Datei hinzu



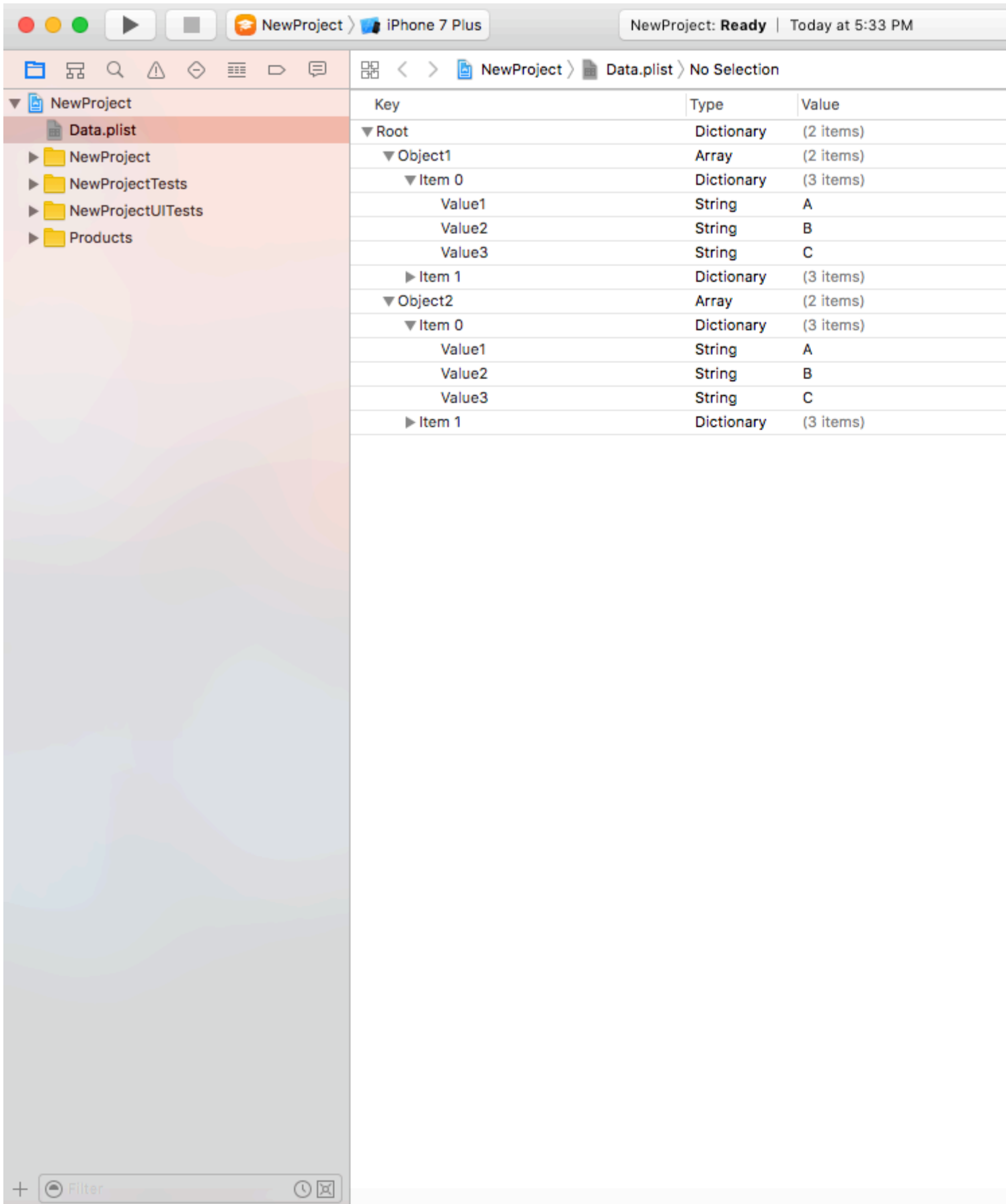
b) Klicken Sie in den Ressourcen auf Eigenschaftsliste



c) Benennen Sie die Eigenschaftsliste, und eine Datei wird erstellt als



d) Sie können eine Liste von Arrays und Wörterbüchern erstellen als:



// Lies die plist vom Bundle und hol das Root Dictionary raus

```
NSMutableDictionary *dictRoot = [NSMutableDictionary dictionaryWithContentsOfFile:[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"];
```

// Ihr Wörterbuch enthält ein Array von Wörterbüchern. // Ziehen Sie jetzt ein Array heraus.

```
NSArray *arrayList = [NSArray arrayWithArray:[dictRoot objectForKey:@"Object1"]];

for(int i=0; i< [arrayList count]; i++)
{
    NSMutableDictionary *details=[arrayList objectAtIndex:i];
}
```

Speichern und Bearbeiten / Löschen von Daten in Plist

Sie haben bereits eine Liste erstellt. Diese Liste wird in App unverändert bleiben. Wenn Sie die Daten in dieser Plist bearbeiten möchten, neue Daten in Plist hinzufügen oder Daten aus Plist entfernen möchten, können Sie keine Änderungen an dieser Datei vornehmen.

Zu diesem Zweck müssen Sie Ihre Liste im Document Directory speichern. Sie können Ihre im Dokumentverzeichnis gespeicherte Pliste bearbeiten.

Plist im Dokumentverzeichnis speichern als:

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"plist"];

NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:filePath];

NSDictionary *plistDict = dict;

NSFileManager *fileManager = [NSFileManager defaultManager];

NSString *error = nil;

NSData *plistData = [NSPropertyListSerialization dataFromPropertyList:plistDict
format:NSPropertyListXMLFormat_v1_0 errorDescription:&error];

if (![fileManager fileExistsAtPath: plistPath]) {

    if(plistData)
    {
        [plistData writeToFile:plistPath atomically:YES];
    }
}
else
{
}
}
```

Retreive-Daten von Plist als:

```
NSArray *paths = NSSearchPathForDirectoriesInDomains (NSDocumentDirectory,
NSUserDomainMask, YES);
NSString *documentsPath = [paths objectAtIndex:0];
NSString *plistPath = [documentsPath stringByAppendingPathComponent:@"Data.plist"];
NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:plistPath];

NSArray *usersArray = [dict objectForKey:@"Object1"];
```

Sie können die Dateien entfernen, neue Daten hinzufügen und die Liste erneut im Dokumentverzeichnis speichern.

plist iOS online lesen: <https://riptutorial.com/de/ios/topic/8141/plist-ios>

Kapitel 129: Profil mit Instrumenten

Einführung

Xcode enthält eine Anwendung zur Leistungsoptimierung mit dem Namen Instruments, mit der Sie Ihre Anwendung mit allen möglichen unterschiedlichen Metriken profilieren können. Sie verfügen über Tools zur Überprüfung der CPU-Nutzung, der Speicherauslastung, der Undichtigkeiten, der Datei- / Netzwerkaktivität und des Energieverbrauchs, um nur einige zu nennen. Es ist sehr einfach, ein Profil Ihrer App von Xcode aus zu erstellen, aber manchmal ist es nicht so einfach zu verstehen, was Sie beim Profilieren sehen. Dies kann einige Entwickler davon abhalten, dieses Tool voll zu nutzen.

Examples

Zeitprofiler

Das erste Instrument, das Sie betrachten, ist der `Time Profiler`. In gemessenen Intervallen stoppt Instruments die Ausführung des Programms und führt für jeden laufenden Thread eine Stapelablaufverfolgung durch. Stellen Sie sich das so vor, als drücken Sie die Pause-Taste im Xcode-Debugger. Hier ist eine Vorschau auf den Time Profiler: -



Running Time	Self	Symbol Name
5838.0ms 46.9%	0.0	▼Main Thread 0xa2db0
5234.0ms 42.0%	0.0	▶ ext.InstrumentsTutorial.ObjectiveC.CIContext
315.0ms 2.5%	0.0	▶ top_level_code InstrumentsTutorial
115.0ms 0.9%	0.0	▶ <Unknown Address>
63.0ms 0.5%	0.0	▶ InstrumentsTutorial.FlickrPhoto
15.0ms 0.1%	0.0	▶ @!objc ext.UIKit.ObjectiveC.CIContext
15.0ms 0.1%	0.0	▶ InstrumentsTutorial.ViewContro
12.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
10.0ms 0.0%	0.0	▶ @!objc ObjectiveC.UImage.init
10.0ms 0.0%	0.0	▶ @!objc ObjectiveC.CIContext.____
8.0ms 0.0%	0.0	▶ InstrumentsTutorial.FlickrPhoto
8.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro
2.0ms 0.0%	0.0	▶ InstrumentsTutorial.Flickr.searc
2.0ms 0.0%	0.0	▶ InstrumentsTutorial.FlickrPhoto
1.0ms 0.0%	0.0	▶ swift_getEnumCaseSinglePaylo
1.0ms 0.0%	1.0	▶ Swift.HeapBufferStorage.__dea
1.0ms 0.0%	0.0	▶ swift_getExistentialTypeMetada
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
1.0ms 0.0%	1.0	▶ _swift_retain_(swift::HeapObjec
1.0ms 0.0%	1.0	▶ swift_unknownRelease libswif
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
1.0ms 0.0%	0.0	▶ swift_getGenericClassObjCNam
1.0ms 0.0%	1.0	▶ _swift_release_(swift::HeapObje
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro

zeigt an, wie viel Zeit in verschiedenen Methoden innerhalb einer App ausgeführt wird. Jede Zeile ist eine andere Methode, der der Ausführungspfad des Programms gefolgt ist. Die Zeit, die für jede Methode aufgewendet wird, kann aus der Anzahl der Stopps des Profilers bei jeder Methode bestimmt werden. Wenn beispielsweise **100 Samples in Intervallen von 1 Millisekunde durchgeführt werden** und eine bestimmte Methode in 10 Samples am oberen Ende des Stapels gefunden wird, können Sie davon ausgehen, dass ungefähr **10%** der gesamten Ausführungszeit - **10 Millisekunden** - aufgewendet wurden in dieser Methode. Es ist eine ziemlich grobe Annäherung, aber es funktioniert!

press `⌘I` Sie in der Xcode's Menüleiste `Product\Profile` oder `press ⌘I` . Dadurch wird die App erstellt und Instruments gestartet. Sie werden mit einem Auswahlfenster begrüßt, das folgendermaßen aussieht:

Choose a profiling template for: iPhone 6 (8.2 Simu

Standard

Custom

Recent



Leaks



Multicore



Network



System Usage



Time Profiler



UI Recorder



Time Profiler

Performs low-overhead time-based sampling of proces

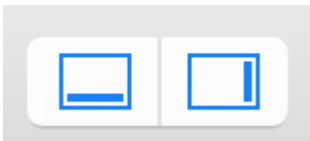
Dies sind alles verschiedene Vorlagen, die mit Instruments geliefert werden.

Wählen Sie das `Time Profiler` Instrument aus und klicken Sie auf Auswählen. Dadurch wird ein neues Instrumentendokument geöffnet. Klicken Sie oben links auf die rote **Aufnahmeschaltfläche**, um die Aufnahme zu starten und die App zu starten. Möglicherweise werden Sie nach Ihrem Passwort gefragt, um **Instruments** zur Analyse anderer Prozesse zu

autorisieren. Im **Instrumente-Fenster sehen** Sie die aufsteigende Zeit und einen kleinen Pfeil, der sich von links nach rechts über der **Grafik** in der Mitte des Bildschirms bewegt. Dies zeigt an, dass die App ausgeführt wird.

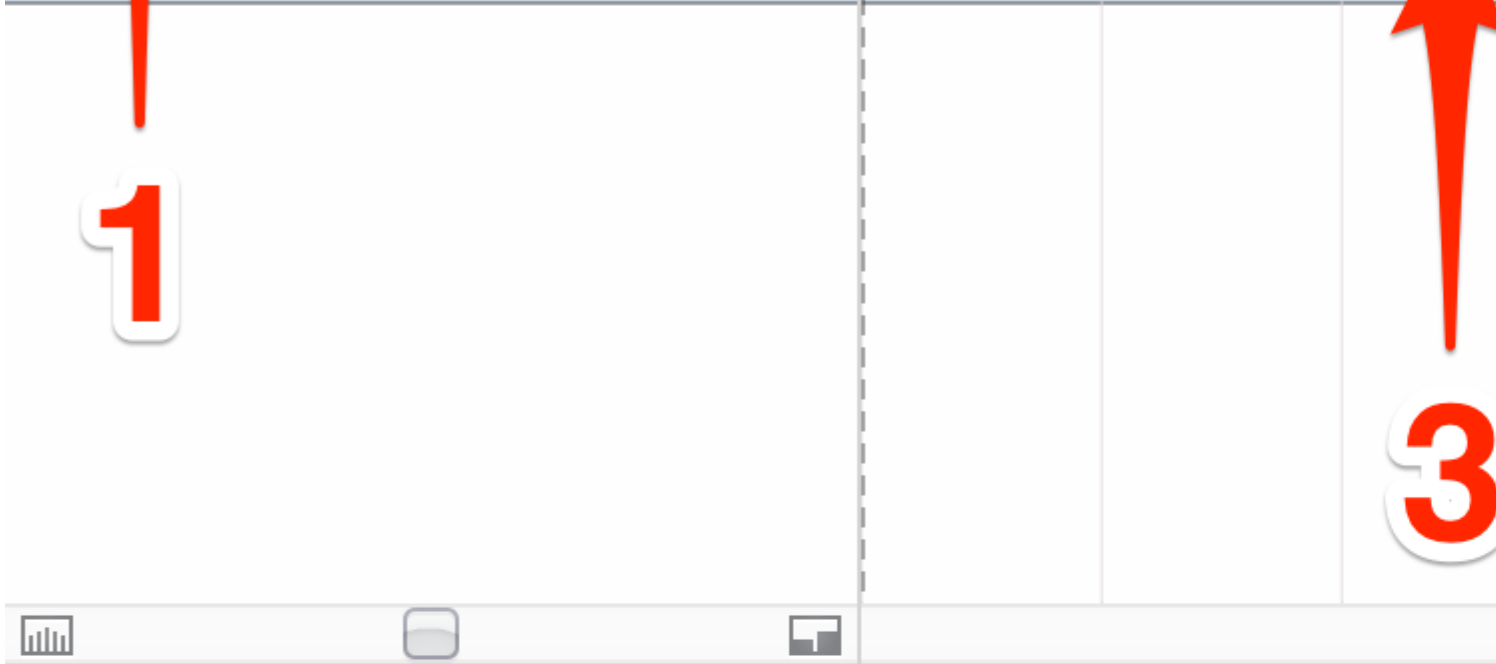
Starten Sie jetzt die App. Suchen Sie nach Bildern, und zeigen Sie ein oder mehrere Suchergebnisse an. Sie haben wahrscheinlich gemerkt, dass das Einsteigen in ein Suchergebnis langwierig ist und das Blättern durch eine Liste von Suchergebnissen auch unglaublich ärgerlich ist - es ist eine furchtbar unübersichtliche App!

Nun, Sie haben Glück, denn Sie werden gleich mit der Reparatur beginnen! Sie werden jedoch zunächst einen Überblick darüber bekommen, was Sie in **Instruments sehen** . Stellen Sie zunächst sicher, dass in der Ansichtsauswahl auf der rechten Seite der Symbolleiste beide Optionen ausgewählt sind:



Dadurch wird sichergestellt, dass alle Felder geöffnet sind. Studieren Sie nun den Screenshot unten und die Erklärung zu jedem Abschnitt darunter:

Time Profiler



Running Time	Self	Symbol Name
4500.0ms 48.6%	0.0	▼ Main Thread 0xa4f45
3903.0ms 42.1%	0.0	▶ ext.InstrumentsTutorial.ObjectiveC.CIContext.__CIContextInit
387.0ms 4.1%	0.0	▶ top_level_code InstrumentsTutorial
76.0ms 0.8%	0.0	▶ <Unknown Address>
39.0ms 0.4%	0.0	▶ InstrumentsTutorial.FlickrPhotoView
16.0ms 0.1%	0.0	▶ @!objc ext.UIKit.ObjectiveC.CIContext.__CIContextInit
10.0ms 0.1%	0.0	▶ InstrumentsTutorial.ViewController
7.0ms 0.0%	0.0	▶ @!objc ObjectiveC.CIContext.__CIContextInit
6.0ms 0.0%	0.0	▶ @!objc ObjectiveC.UIImage.init
5.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResultsView
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResultsView

. Die rote Schaltfläche "Aufnahme" stoppt und startet die App, die gerade profiliert wird, wenn auf diese geklickt wird (sie wechselt zwischen einem Aufnahme- und Stoppsymbol). Die Pause-Schaltfläche macht genau das, was Sie erwarten, und stoppt die aktuelle Ausführung der App.

2. Dies ist der Run-Timer. Der Timer zählt, wie lange die App ausgeführt wird und wie oft sie ausgeführt wurde. Wenn Sie die App mit den Aufnahmesteuerelementen stoppen und dann erneut starten, wird ein neuer Lauf gestartet, und auf dem Display wird Run 2 von 2 angezeigt.

3. Dies wird als Spur bezeichnet. In der von Ihnen ausgewählten Zeitprofiler-Vorlage gibt es nur ein Instrument, also nur eine Spur. Sie erfahren später in diesem Tutorial mehr über die Besonderheiten des hier gezeigten Diagramms.

4. Dies ist das Detailfenster. Es zeigt die wichtigsten Informationen zu dem jeweiligen Instrument, das Sie verwenden. In diesem Fall werden die "heißesten" Methoden gezeigt, dh die Methoden, die die meiste CPU-Zeit verbraucht haben. Wenn Sie oben auf die Leiste mit der Aufrufstruktur (die linke Hand) klicken und Probenliste auswählen, wird eine andere Ansicht der Daten angezeigt. Diese Ansicht zeigt jede einzelne Probe. Klicken Sie auf einige Beispiele, um die erfasste Stack-Ablaufverfolgung im Inspektor für erweiterte Details anzuzeigen.

5. Dies ist das Inspektorenfeld. Es gibt drei Inspektoren: Aufnahmeeinstellungen, Anzeigeeinstellungen und Erweiterte Details. In Kürze erfahren Sie mehr über einige dieser Optionen.

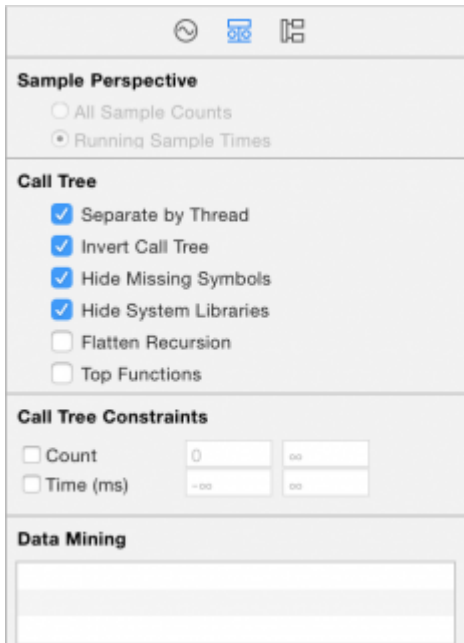
Tief bohren

Führen Sie eine Bildersuche durch und untersuchen Sie die Ergebnisse. Ich persönlich suche gerne nach "Hund", aber wähle, was du willst - du könntest einer dieser Katzenmenschen sein!

Scrollen Sie nun einige Male in der Liste nach oben und unten, damit im `Time Profiler` eine gute Datenmenge `Time Profiler` . Sie sollten bemerken, dass sich die Zahlen in der Mitte des Bildschirms ändern und der **Graph** ausgefüllt wird. Dies zeigt an, dass **CPU- Zyklen** verwendet werden.

Sie würden wirklich nicht erwarten, dass eine Benutzeroberfläche so klobig ist, da keine `table view` zum Versand bereit ist, bis sie wie Butter rollt! Um das Problem zu lokalisieren, müssen Sie einige Optionen festlegen.

Wählen Sie auf der rechten Seite den **Inspektor für Anzeigeeinstellungen aus** (or press `⌘+2`) . Im **Inspektoren** unter der `Call Tree` Wählen Sie im Abschnitt **Separate nach Thema**, **Invert Call Tree** , ausblenden Symbole und Ausblenden Systembibliotheken fehlt. Es wird so aussehen:



Die einzelnen Optionen machen jeweils die in der Tabelle links angezeigten Daten:

Getrennt nach Thread: Jeder Thread sollte separat betrachtet werden. Auf diese Weise können Sie verstehen, welche Threads für die größte **CPU**- Auslastung verantwortlich sind.

Anrufbaum umkehren: Mit dieser Option wird die `stack trace` von oben nach unten betrachtet. Dies ist normalerweise das, was Sie möchten, da Sie die tiefsten Methoden sehen möchten, in denen die **CPU** ihre Zeit verbringt.

Fehlende Symbole ausblenden: Wenn die `dsym` Datei für Ihre App oder ein `system framework` nicht gefunden werden kann, werden anstelle der Methodennamen (Symbole) in der Tabelle nur Hex-Werte angezeigt, die den Adressen in der Binärdatei entsprechen. Wenn diese Option ausgewählt ist, werden nur vollständig aufgelöste Symbole angezeigt und die nicht aufgelösten **Hex**- Werte werden ausgeblendet. Dies hilft, die angezeigten Daten zu entschlüsseln.

Systembibliotheken ausblenden: Wenn diese Option ausgewählt ist, werden nur Symbole Ihrer eigenen App angezeigt. Es ist oft nützlich, diese Option zu wählen, da Sie normalerweise nur darauf achten, wo die **CPU** Zeit in Ihrem eigenen Code verbringt - Sie können nicht viel darüber tun, wie viel **CPU** die `system libraries` verwenden!

Rekursion reduzieren: Diese Option behandelt **rekursive** Funktionen (solche, die sich selbst aufrufen) als einen Eintrag in jeder `stack trace` und nicht als mehrere.

Top-Funktionen: Wenn Sie dies `Instruments` berücksichtigt `Instruments` die Gesamtzeit einer Funktion als Summe der Zeit direkt in dieser Funktion sowie die Zeit, die in Funktionen verbracht wird, die von dieser Funktion aufgerufen werden.

Wenn die Funktion A also B anruft, wird die Zeit von A als die in A PLUS verbrachte Zeit und die in B verbrachte Zeit angegeben. Dies kann sehr nützlich sein, da Sie bei jedem Abstieg in den Aufrufstapel die größte Zeitangabe wählen können in Ihre zeitaufwändigsten Methoden.

Wenn Sie eine `Objective-C` App ausführen, gibt es auch die Option **Nur Obj-C** anzeigen: Wenn

diese Option ausgewählt ist, werden nur Objective-C Methoden und keine C oder C++ Funktionen angezeigt. Es gibt keine in Ihrem Programm, aber wenn Sie sich eine OpenGL App angesehen haben, könnte sie beispielsweise C++ .

Obwohl einige Werte leicht abweichen können, sollte die Reihenfolge der Einträge der folgenden Tabelle ähneln, sobald Sie die obigen Optionen aktiviert haben:

Running Time	Self	Symbol Name
12682.0ms	48.0%	0.0
11858.0ms	44.8%	11858.0
428.0ms	1.6%	0.0
186.0ms	0.7%	0.0
120.0ms	0.4%	0.0
23.0ms	0.0%	0.0
12.0ms	0.0%	0.0
9.0ms	0.0%	0.0
7.0ms	0.0%	0.0
5.0ms	0.0%	0.0
4.0ms	0.0%	0.0
4.0ms	0.0%	0.0
4.0ms	0.0%	0.0
4.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
2.0ms	0.0%	0.0
1.0ms	0.0%	1.0
1.0ms	0.0%	0.0
1.0ms	0.0%	1.0
1.0ms	0.0%	1.0
1.0ms	0.0%	0.0

Nun, das sieht auf keinen Fall gut aus. Die überwiegende Mehrheit der Zeit wird in der Methode aufgewendet, die den Filter "Ton" auf die Miniaturbilder anwendet. Das sollte für Sie kein allzu großer Schock sein, da das Laden und Scrollen der Tabellen die rauesten Bereiche der Benutzeroberfläche waren, und dann werden die Tabellenzellen ständig aktualisiert.

Um mehr über die Vorgänge in dieser Methode zu erfahren, doppelklicken Sie auf die entsprechende Zeile in der Tabelle. Dadurch wird die folgende Ansicht angezeigt:

```

15
16 class var sharedCache: ImageCache {
17     return _sharedCache
18 }
19
20 func setImage(image: UIImage, forKey key: String) {
21     images[key] = image
22 }
23
24 func imageForKey(key: String) -> UIImage? {
25     return images[key]
26 }
27 }
28
29 extension UIImage {
30     func applyTonalFilter() -> UIImage? {
31         let context = CIContext(options:nil)
32         let filter = CIFilter(name:"CIPhotoEffectTonal")
33         let input = CoreImage.CIImage(image: self)
34         filter.setValue(input, forKey: kCIInputImageKey)
35         let outputImage = filter.outputImage
36
37         let outImage = context.createCGImage(outputImage, fromRect: outputImage.extent())
38         let returnImage = UIImage(CGImage: outImage)
39         return returnImage
40     }
41 }
42

```

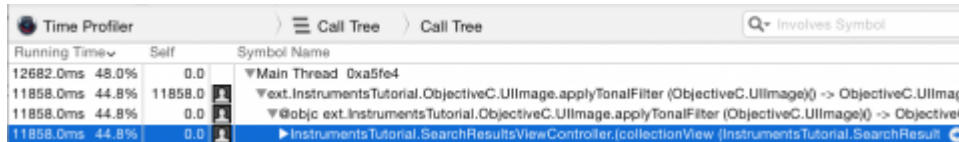
Nun, das ist interessant, nicht wahr? `applyTonalFilter()` ist eine zu `UIImage` in einer Erweiterung hinzugefügte Methode, und nach dem Anwenden des Bildfilters wird fast **100 %** der Zeit damit verbracht, die `CGImage`-Ausgabe zu erstellen.

Es lässt sich nicht viel tun, um dies zu beschleunigen: Das Erstellen des Images ist ein recht intensiver Prozess und dauert so lange, wie es dauert. Lassen Sie uns einen Schritt zurückgehen und sehen, von wo aus `applyTonalFilter()` aufgerufen wird. **Klicken** `Call Tree` in der

Navigationsleiste oben in der Codeansicht auf `Call Tree` , um zum vorherigen Bildschirm zurückzukehren:



Klicken Sie nun auf den kleinen Pfeil links neben der Zeile `applyTonalFilter` oben in der Tabelle. Dadurch entfaltet sich die Aufrufstruktur, um den Aufrufer von `applyTonalFilter` anzuzeigen. Möglicherweise müssen Sie auch die nächste Reihe aufklappen. Beim Profilieren von Swift werden in der Aufrufstruktur manchmal doppelte Zeilen angezeigt, denen `@objc` vorangestellt ist. Sie interessieren sich für die erste Zeile, der der Zielname Ihrer App vorangestellt ist (`InstrumentsTutorial`):

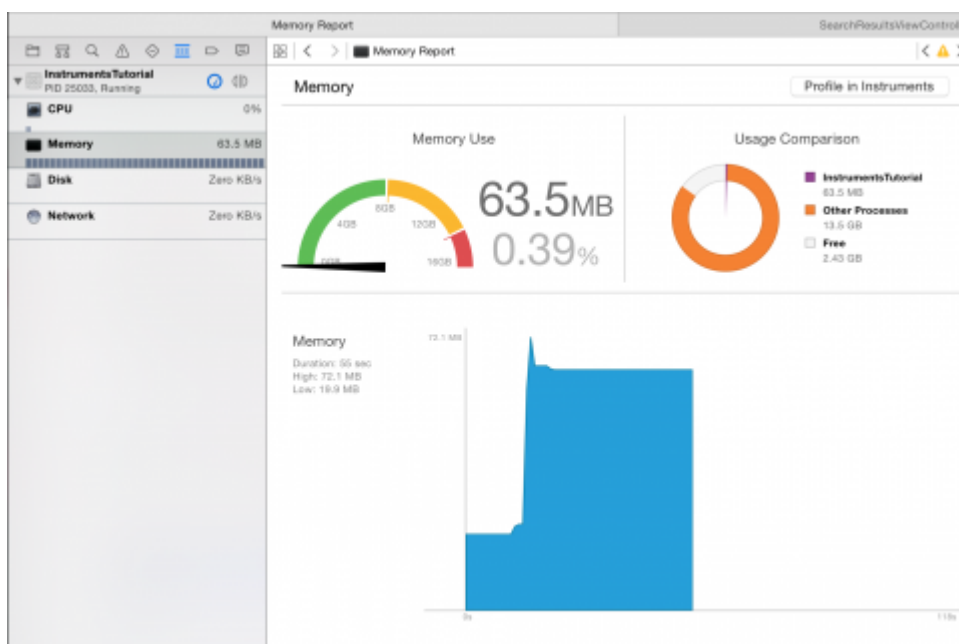


In diesem Fall bezieht sich diese Zeile auf die `cellForItemAtIndexPath` Ansicht der `cellForItemAtIndexPath` . Doppelklicken Sie auf die Zeile, um den zugehörigen Code aus dem Projekt anzuzeigen.

Jetzt können Sie sehen, was das Problem ist. Die Methode zum Anwenden des Tonfilters dauert lange, und er wird direkt von `cellForItemAtIndexPath` aufgerufen, wodurch der `main thread` (und damit die gesamte Benutzeroberfläche) jedes Mal blockiert wird, wenn ein gefiltertes Bild angefordert wird.

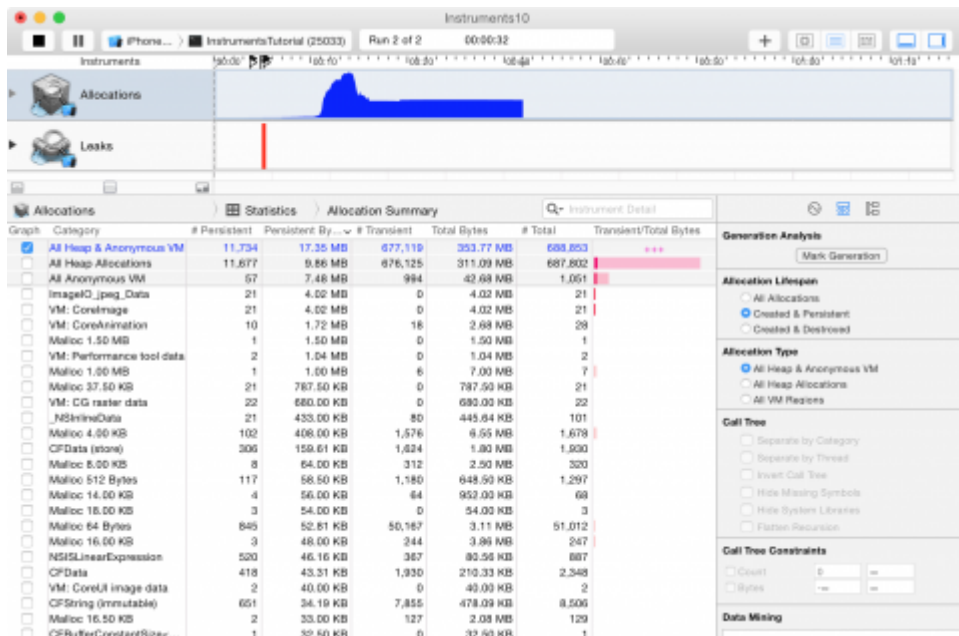
Zuteilungen

Es gibt detaillierte Informationen zu allen **Objekten** , die erstellt werden, und den Speicher, der sie unterstützt. Es zeigt auch, dass Sie `retain counts` für jedes Objekt `retain counts` . Um erneut mit einem neuen `instruments profile` , beenden Sie die Instruments-App. Erstellen und starten Sie die App, und öffnen Sie den Debug Navigator im Bereich Navigators. Klicken Sie dann auf **Speicher** , um im Hauptfenster Grafiken zur Speicherbelegung anzuzeigen:



Diese Diagramme sind hilfreich, um einen schnellen Überblick über die Leistung Ihrer App zu

erhalten. Aber du brauchst ein bisschen mehr Kraft. Klicken Sie auf die Schaltfläche `Profile` in `Instruments` und dann auf Übertragen, um diese Sitzung in **Instruments** zu bringen. Das **Allocations-Instrument** wird automatisch gestartet.



Dieses Mal werden Sie zwei Spuren bemerken. Eines heißt Allokationen und eines heißt Leaks. Der Allocations-Track wird später detailliert beschrieben. Die Leaks-Spur ist im Allgemeinen in Objective-C nützlicher und wird in diesem Tutorial nicht behandelt. Welchen Fehler wirst du als nächstes aufspüren? In dem Projekt ist etwas versteckt, von dem Sie wahrscheinlich nicht wissen, dass es dort ist. Sie haben wahrscheinlich von Speicherlecks gehört. Was Sie vielleicht nicht wissen, ist, dass es tatsächlich zwei Arten von Leaks gibt:

Echte Speicherverluste sind, wenn ein Objekt nicht mehr von irgendetwas referenziert wird, sondern immer noch zugewiesen ist. Dies bedeutet, dass der Speicher nie wieder verwendet werden kann. Selbst wenn Swift und ARC bei der Verwaltung des Speichers helfen, ist die häufigste Art eines Speicherverlusts ein `retain cycle` or `strong reference cycle`. Dies ist der Fall, wenn zwei Objekte starke Verweise auf einander enthalten, sodass jedes Objekt das andere Objekt davon abhält, die Zuordnung aufzuheben. Dies bedeutet, dass ihr Gedächtnis niemals freigegeben wird!

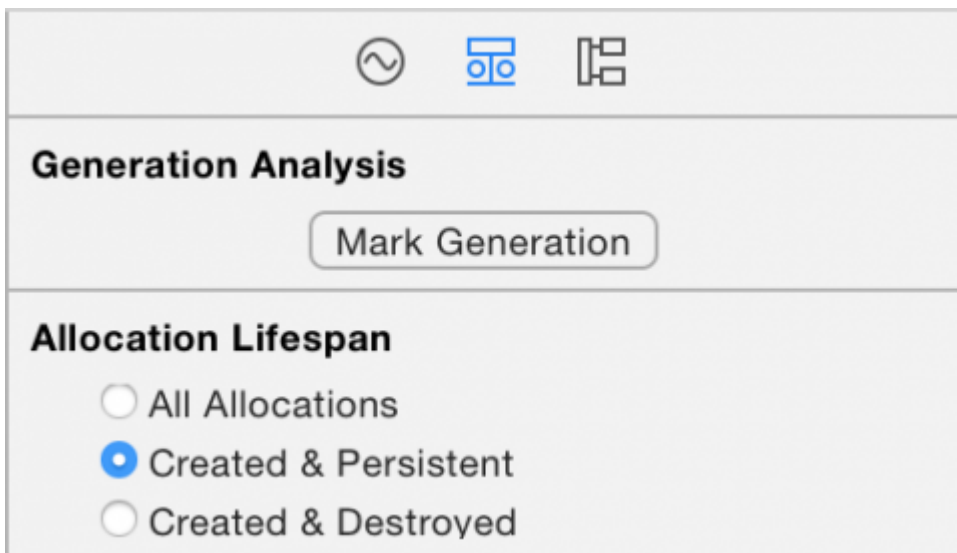
Bei unbegrenztem Speicherwachstum wird weiterhin Speicher zugewiesen, und es besteht nie die Möglichkeit, die **Freigabe aufzuheben**. Wenn dies für immer andauert, `system's memory` des `system's memory` irgendwann gefüllt und Sie haben ein großes Speicherproblem. In iOS bedeutet dies, dass die App vom System beendet wird.

Führen Sie mit dem Allocations- **Instrument**, das in der App ausgeführt wird, fünf verschiedene Suchvorgänge in der App durch, ohne jedoch die Ergebnisse zu untersuchen. Stellen Sie sicher, dass die Suche einige Ergebnisse hat! Nun lassen Sie die App ein wenig warten, indem Sie einige Sekunden warten.

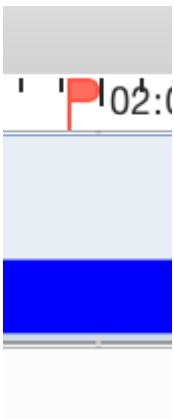
Sie sollten bemerkt haben, dass die **Grafik** in der Zuweisungsspur ansteigt. Dies sagt Ihnen, dass Speicher zugewiesen wird. Mit dieser Funktion können Sie `unbounded memory growth`.

Was Sie ausführen werden, ist eine `generation analysis`. Drücken Sie dazu den Button `Mark`

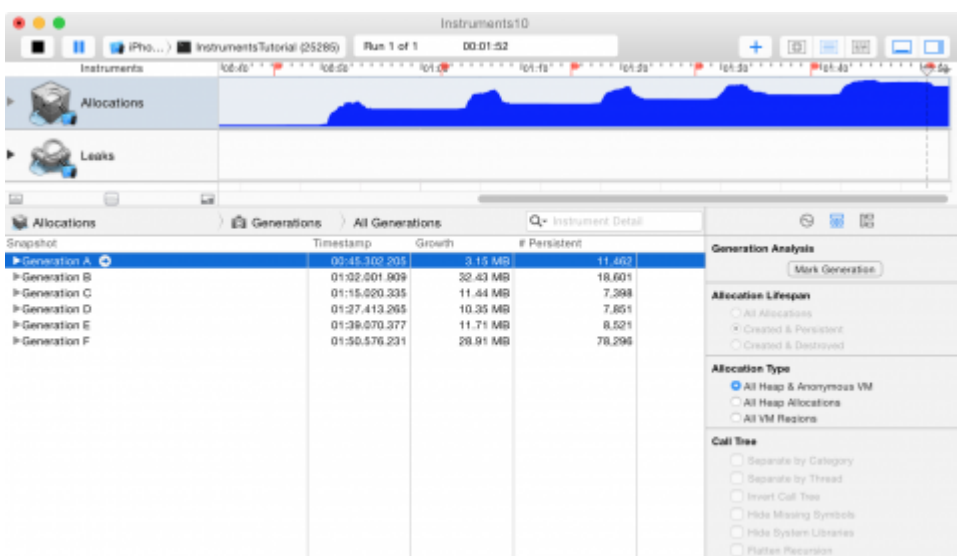
Generation. Die Schaltfläche befindet sich oben im Inspektor für Anzeigeeinstellungen:



Drücken Sie die Taste und Sie sehen eine rote Flagge in der Spur:



Der Zweck der `generation analysis` besteht darin, eine Aktion mehrmals auszuführen und zu sehen, ob der Speicher `unbounded fashion` wächst. **Drill** in eine Suche, warten Sie einige Sekunden, bis die Bilder zu laden, und dann gehen Sie zurück zur Startseite. Dann markieren Sie die Generation erneut. Wiederholen Sie dies für verschiedene Suchvorgänge. Nach einer in ein paar Recherchen **Bohren Instruments** wird wie folgt aussehen:



An diesem Punkt sollten Sie misstrauisch werden. Beachten Sie, wie der blaue Graph bei jeder Suche, in die Sie **bohren**, steigt. Das ist sicherlich nicht gut. Aber warten Sie, was ist mit `memory warnings`? Sie wissen davon, richtig? `Memory warnings` sind eine Möglichkeit von iOS, einer App mitzuteilen, dass es in der Speicherabteilung eng wird und Sie etwas Speicher löschen müssen.

Es ist möglich, dass dieses Wachstum nicht nur auf Ihre App zurückzuführen ist. Es könnte etwas in den Tiefen von `UIKit`, das sich an die Erinnerung hält. Geben Sie den System-Frameworks und Ihrer App die Möglichkeit, zuerst den **Speicher** zu löschen, bevor Sie mit dem Finger auf einen der beiden zeigen.

Simulieren Sie eine `memory warning` indem Sie in der Menüleiste von `Instrument\Simulate Memory Warning` in der Menüleiste des `simulator's Hardware\Simulate Memory Warning` auswählen. Sie werden feststellen, dass der Speicherbedarf ein wenig oder gar nicht sinkt. Sicher nicht zurück, wo es sein sollte. Irgendwo passiert also immer noch **unbegrenztes Gedächtniswachstum**.

Der Grund für das Markieren einer Generation nach jeder Wiederholung einer Suche in einer Suche ist, dass Sie sehen können, welcher **Speicher** zwischen den einzelnen Generationen zugewiesen wurde. Werfen Sie einen Blick in das Detailfenster und Sie werden eine Reihe von Generationen sehen.

Profil mit Instrumenten online lesen: <https://riptutorial.com/de/ios/topic/9629/profil-mit-instrumenten>

Kapitel 130: QR Code Scanner

Einführung

QR-Codes (Quick Response-Codes) sind zweidimensionale Barcodes, die auf maschinenlesbaren optischen Etiketten häufig verwendet werden. iOS bietet die Möglichkeit, QR-Codes mithilfe des `AVFoundation` Frameworks ab iOS 7 zu lesen. Dieses Framework bietet eine Reihe von APIs zum Einrichten / Öffnen der Kamera und zum Lesen von QR-Codes aus dem Kamera-Feed.

Examples

UIViewController sucht nach QR und zeigt den Videoeingang an

```
import AVFoundation
class QRScannerViewController: UIViewController,
    AVCaptureMetadataOutputObjectsDelegate {

    func viewDidLoad() {
        self.initCaptureSession()
    }

    private func initCaptureSession() {
        let captureDevice = AVCaptureDevice
            .defaultDevice(withMediaType: AVMediaTypeVideo)
        do {
            let input = try AVCaptureDeviceInput(device: captureDevice)
            let captureMetadataOutput = AVCaptureMetadataOutput()
            self.captureSession?.addOutput(captureMetadataOutput)
            captureMetadataOutput.setMetadataObjectsDelegate(self,
                queue: DispatchQueue.main)
            captureMetadataOutput
                .metadataObjectTypes = [AVMetadataObjectTypeQRCode]

            self.videoPreviewLayer =
                AVCaptureVideoPreviewLayer(session: self.captureSession)
            self.videoPreviewLayer?
                .videoGravity = AVLayerVideoGravityResizeAspectFill
            self.videoPreviewLayer?.frame =
                self.view.layer.bounds

            self._viewController?.view.layer
                .addSublayer(videoPreviewLayer!)
            self.captureSession?.startRunning()
        } catch {
            //TODO: handle input open error
        }
    }

    private func dismissCaptureSession() {
        if let running = self.captureSession?.isRunning, running {
            self.captureSession?.stopRunning()
        }
        self.captureSession = nil
        self.videoPreviewLayer?.removeFromSuperLayer()
        self.videoPreviewLayer = nil
    }
}
```

```

}

func captureOutput(_ captureOutput: AVCaptureOutput,
  didOutputMetadataObjects metadataObjects: [Any]!,
  from connection: AVCaptureConnection) {
  guard metadataObjects != nil && metadataObjects.count != 0 else {
    //Nothing captured
    return
  }

  if let metadataObj =
    metadataObjects[0] as? AVMetadataMachineReadableCodeObject {
    guard metadataObj.type == AVMetadataObjectTypeQRCode else {
      return
    }

    let barCodeObject = videoPreviewLayer?
      .transformedMetadataObject(for:
        metadataObj as AVMetadataMachineReadableCodeObject)
      as! AVMetadataMachineReadableCodeObject

    if let qrValue = metadataObj.stringValue {
      self.handleQRRead(value: qrValue)
    }
  }
}

private handleQRRead(value: String) {
  //TODO: Handle the read qr
}
private captureSession: AVCaptureSession?
private videoPreviewLayer: AVCaptureVideo
}

```

handleQRRead - Wird bei einem erfolgreichen Scan
 initCaptureSession - Scan für
 dismissCaptureSession und Kameraeingabe initialisieren
 dismissCaptureSession -
 dismissCaptureSession ausblenden und Scanvorgang stoppen

QR-Code mit dem AVFoudation-Framework scannen

Vor iOS 7, wenn Sie einen QR-Code scannen möchten, müssen wir möglicherweise auf Frameworks oder Bibliotheken von [Dritten](#) wie [zBar](#) oder [zXing](#) zurückgreifen . Apple hat jedoch AVCaptureMetaDataOutput von iOS 7 zum Lesen von Barcodes eingeführt.

Um QR-Code mit AVFoundation lesen zu können, müssen Sie AVFoundation AVCaptureSession und erstellen und die captureOutput:didOutputMetadataObjects:fromConnection: delegate-Methode verwenden.

Schritt 1

Importieren Sie das AVFoundation Framework und bestätigen

AVCaptureMetadataOutputObjectsDelegate Protokoll AVCaptureMetadataOutputObjectsDelegate

```
import AVFoundation
class ViewController: UIViewController, AVCaptureMetadataOutputObjectsDelegate
```

Schritt 2

Das Lesen von QR-Codes basiert vollständig auf der Videoaufnahme. `AVCaptureSession` fortlaufende Videos aufzunehmen, erstellen Sie eine `AVCaptureSession` und richten Sie die Geräteeingabe und -ausgabe ein. Fügen Sie den folgenden Code in der `viewDidLoad` Methode des View-Controllers `viewDidLoad`

```
// Create an instance of the AVCaptureDevice and provide the video as the media type
parameter.
let captureDevice = AVCaptureDevice.defaultDevice(withMediaType: AVMediaTypeVideo)

do {
    // Create an instance of the AVCaptureDeviceInput class using the device object and
    initialise capture session
    let input = try AVCaptureDeviceInput(device: captureDevice)
    captureSession = AVCaptureSession()
    captureSession?.addInput(input)

    // Create a instance of AVCaptureMetadataOutput object and set it as the output device the
    capture session.
    let captureMetadataOutput = AVCaptureMetadataOutput()
    captureSession?.addOutput(captureMetadataOutput)
    // Set delegate with a default dispatch queue
    captureMetadataOutput.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)
    //set meta data object type as QR code, here we can add more then one type as well
    captureMetadataOutput.metadataObjectTypes = [AVMetadataObjectTypeQRCode]

    // Initialize the video preview layer and add it as a sublayer to the viewcontroller
    view's layer.
    videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
    videoPreviewLayer?.videoGravity = AVLayerVideoGravityResizeAspectFill
    videoPreviewLayer?.frame = view.layer.bounds
    view.layer.addSublayer(videoPreviewLayer!)

    // Start capture session.
    captureSession?.startRunning()
} catch {
    // If any error occurs, let the user know. For the example purpose just print out the
    error
    print(error)
    return
}
```

Schritt 3

Implementieren `AVCaptureMetadataOutputObjectsDelegate` Delegatenmethode

`AVCaptureMetadataOutputObjectsDelegate` , um den QR-Code zu lesen

```

func captureOutput(_ captureOutput: AVCaptureOutput!, didOutputMetadataObjects
metadataObjects: [Any]!, from connection: AVCaptureConnection!) {

    // Check if the metadataObjects array contains at least one object. If not no QR code is
in our video capture
    if metadataObjects == nil || metadataObjects.count == 0 {
        // NO QR code is being detected.
        return
    }

    // Get the metadata object and cast it to `AVMetadataMachineReadableCodeObject`
    let metadataObj = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

    if metadataObj.type == AVMetadataObjectTypeQRCode {
        // If the found metadata is equal to the QR code metadata then get the string value
from meta data
        let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)

        if metadataObj.stringValue != nil {
            // metadataObj.stringValue is our QR code
        }
    }
}

```

Hier können Metadatenobjekte auch die Grenzen des im Kamera-Feed gelesenen QR-Codes angeben. Um die Grenzen zu ermitteln, übergeben Sie das Metadatenobjekt einfach an `videoPreviewLayer transformedMetadataObject videoPreviewLayer` -Methode (`videoPreviewLayer` unten).

```

let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)
qrCodeFrameView?.frame = barCodeObject!.bounds

```

QR Code Scanner online lesen: <https://riptutorial.com/de/ios/topic/7963/qr-code-scanner>

Kapitel 131: Reich

Bemerkungen

Hinzufügen eines neuen RLMO-Objekts zu einem vorhandenen Realm - Schema und Migrationen

Das Hinzufügen neuer Modellklassen zu einem Realm erfordert keine Migration oder keine Schemaversion. Nur Änderungen an einem vorhandenen Realm vornehmen.

Examples

RLMObject-Basismodellklasse mit Primärschlüssel - Objective-C

Ein Beispiel für eine RLMObject-Basismodellklasse, die einen Primärschlüssel und einige allgemeine Standardeigenschaften verwendet. Unterklassen können dann Metadaten festlegen, die ihren Bedürfnissen entsprechen.

```
@interface BaseModel : RLMObject

@property NSString *uuid;
@property NSString *metadata;

@end

@implementation BaseModel

+ (NSString *)primaryKey
{
    return @"uuid";
}

+ (NSDictionary *)defaultPropertyValues
{
    NSMutableDictionary *defaultPropertyValues = [NSMutableDictionary
dictionaryWithDictionary:[super defaultPropertyValues]];
    NSString *uuid = [[NSUUID UUID] UUIDString];
    [defaultPropertyValues setValue:@"" forKey:@"metadata"];
    [defaultPropertyValues setValue:uuid forKey:@"uuid"];
    return defaultPropertyValues;
}

+ (NSArray *)ignoredProperties
{
    return @[];
}

@end
```

Reich online lesen: <https://riptutorial.com/de/ios/topic/4084/reich>

Kapitel 132: Rich Benachrichtigungen

Einführung

Mit den Rich Notifications können Sie das Erscheinungsbild von lokalen und Remote-Benachrichtigungen anpassen, wenn sie auf dem Gerät des Benutzers angezeigt werden. Zu dieser Benachrichtigung gehören hauptsächlich `UNNotificationServiceExtension` und `UNNotificationContentExtension`, dh die normale Benachrichtigung wird auf erweiterte Weise angezeigt

Examples


Erstellen einer einfachen `UNNotificationContentExtension`

Schritt 1


Die Umgebung für die Benachrichtigung geeignet machen. Stellen Sie sicher, dass Sie **Hintergrundmodi** und **Push-Benachrichtigung** aktiviert haben




PROJECT

 TestApplication

TARGETS

 **TestApplication**

 TestAppNotifConten...



Filter



Background Modes



Inter-App Audio



Keychain Sharing



Associated Domains




App Groups




General


Capa

PROJECT

 TestApplication


TARGETS

 **TestApplication**

 TestAppNotifConten...

▶  **iCloud**

▼  **Push Notifications**

▶  **Game Center**

▶  **Wallet**

▶  **Siri**

▶  **Apple Pay**

▶  **In-App Purchase**

▶  **Maps**



Filter

Schritt 2: Erstellen einer UNNotificationContentExtension

Klicken Sie unten auf das + -Symbol, um eine Zielvorlage zu erstellen, und wählen Sie Benachrichtigungsinhaltserweiterung -> Weiter -> Erstellen Sie einen Namen für die Inhaltserweiterung -> Fertig stellen

PROJECT

TARGETS

- Test
- Test
- Test

Choose a template for your new target:

ios watchOS tvOS macOS Cr

Application Extension



Action Extension



Audio Unit Extension



Content Blocker Extension



Custom Keyboard Extension



Intents UI Extension



Message Filter Extension



Click this + icon

- `UNNotificationDefaultContentHidden`: Dieser Boolean-Wert bestimmt, ob der Standardtext der Benachrichtigung ausgeblendet werden soll oder nicht
- `UNNotificationCategory`: Die Kategorie wird im `UNUserNotificationCenter` in Ihrer Anwendung erstellt. Hierbei kann es sich entweder um einen String oder ein Array von Strings handeln, sodass jede Kategorie unterschiedliche Datentypen angeben kann, aus denen verschiedene Benutzeroberflächen erstellt werden können. Die von uns gesendete Nutzlast muss den Kategorienamen enthalten, um diese bestimmte Erweiterung anzuzeigen in Ihrer Anwendung erstellt. Hierbei kann es sich entweder um einen String oder ein Array von Strings handeln, sodass jede Kategorie unterschiedliche Datentypen angeben kann, aus denen verschiedene Benutzeroberflächen erstellt werden können. Die von uns gesendete Nutzlast muss den Kategorienamen enthalten, um diese bestimmte Erweiterung anzuzeigen
- `UNNotificationExtensionInitialContentSizeRatio`: Die Größe des ursprünglichen Inhalts, dh bei der ersten Anzeige der ContentExtension wird die ursprüngliche Größe in Bezug auf die Breite des Geräts angezeigt. Hier bedeutet 1, dass die Höhe der Breite entspricht

Schritt 4: Erstellen von `UNNotificationAction` und `UNNotificationCategory` in unserer Anwendung

In `AppDelegate.swift` Ihrer App fügen `didFinishLaunchingWithOptions` Funktion

`didFinishLaunchingWithOptions` hinzu

```

let userNotificationAction:UNNotificationAction = UNNotificationAction.init(identifier:
"ID1", title: "வணக்கம்",options: .destructive)
let userNotificationAction2:UNNotificationAction = UNNotificationAction.init(identifier:
"ID2", title: "Success", options: .destructive)

let notifCategory:UNNotificationCategory = UNNotificationCategory.init(identifier:
"CATID1", actions: [userNotificationAction,userNotificationAction2], intentIdentifiers:
["ID1","ID2"] , options:.customDismissAction)

UNUserNotificationCenter.current().delegate = self
UNUserNotificationCenter.current().setNotificationCategories([notifCategory])
UIApplication.shared.registerForRemoteNotifications()

```

Wir haben zwei `UNNotificationAction` mit den Bezeichnern `ID1` und `ID2` und diese Aktionen zu einer `UNNotificationCategory` mit dem Bezeichner `CATID1` hinzugefügt. Wir setzen die Kategorie auf `UNUserNotificationCenter` unserer Anwendung. In der nächsten Zeile registrieren wir uns für die Benachrichtigung, in der die Funktion `didRegisterForRemoteNotificationsWithDeviceToken` wird, in der das `didRegisterForRemoteNotificationsWithDeviceToken`

Hinweis: Vergessen Sie nicht, `import UserNotifications` in `AppDelegate.swift` zu `import UserNotifications` und `UNUserNotificationCenterDelegate` hinzuzufügen

Schritt 5: Beispielnutzdaten für den `NotificationContent`

```

'aps': {
  'badge': 0,
  'alert': {
    'title': "Rich Notification",
    'body': "Body of RICH NOTIFICATION",
  },
  'sound' : "default",
  'category': "CATID1",

```

```
'mutable-content':"1",
},
'attachment': "2"
```

Schritt 6: Konfigurieren der ContentExtension

Die entsprechenden Aktionen für die Kategorie werden automatisch angezeigt, während die Benachrichtigungsaktion ausgeführt wird. Lässt uns den Code sehen, wie er ausgeführt wird

```
import UIKit
import UserNotifications
import UserNotificationsUI

class NotificationViewController: UIViewController, UNNotificationContentExtension {

@IBOutlet var imageView: UIImageView?
override func viewDidLoad() {
    super.viewDidLoad()
}

func didReceive(_ notification: UNNotification) {
    self.title = "Koushik"
    imageView?.backgroundColor = UIColor.clear
    imageView?.image = #imageLiteral(resourceName: "welcome.jpeg")
}

func didReceive(_ response: UNNotificationResponse, completionHandler completion: @escaping
(UNNotificationContentExtensionResponseOption) -> Void) {

    self.title = "Koushik"
    imageView?.image = UIImage.init(named: "Success.jpeg")

    if(response.actionIdentifier == "ID1")
    {
        imageView?.image = UIImage.init(named: "Success.jpeg")
    }
    else
    {
        imageView?.image = UIImage.init(named: "welcome.jpeg")
    }
}
}
```

Schritt 7: Ergebnis

Nach dem Empfang und langem Drücken von / Klicken auf Benachrichtigung anzeigen sieht die Benachrichtigung folgendermaßen aus



Koushik



Rich Notification

Body of RICH NOTIFICATION

வணக்கம்

Success



Koushik



Rich Notification

Body of RICH NOTIFICATION

வணக்கம்

Success



Koushik



Koushik



`UNNotificationExtensionOverrideDefaultTitle` als YES angegeben haben. In Schritt 3 haben wir `UNNotificationExtensionDefaultContentHidden` als NEIN angegeben. Wenn JA, dann sieht die Benachrichtigung wie in Bild 3 und 4 aus.

Rich Benachrichtigungen online lesen: <https://riptutorial.com/de/ios/topic/10769/rich-benachrichtigungen>

Kapitel 133: Safari-Dienstleistungen

Examples

Implementieren Sie SFSafariViewControllerDelegate

Sie sollten `SFSafariViewControllerDelegate` so implementieren, dass Ihre Klasse benachrichtigt wird, wenn der Benutzer auf die Schaltfläche "Fertig" des `SafariViewController`s klickt.

Erklären Sie zuerst Ihre Klasse, um das Protokoll zu implementieren.

```
class MyClass: SFSafariViewControllerDelegate {  
  
}
```

Implementieren Sie die Delegierungsmethode, die bei der Kündigung zu benachrichtigen ist.

```
func safariViewControllerDidFinish(controller: SFSafariViewController) {  
    // Dismiss the SafariViewController when done  
    controller.dismissViewControllerAnimated(true, completion: nil)  
}
```

Vergessen Sie nicht, Ihre Klasse als Delegat des `SafariViewController`s festzulegen.

```
let safariVC = SFSafariViewController(URL: yourURL)  
safariVC.delegate = self
```

Zusätzliche Delegierungsmethoden, die Sie implementieren können, sind:

```
// Called when the initial URL load is complete.  
safariViewController(_ controller: SFSafariViewController, didCompleteInitialLoad  
didLoadSuccessfully: Bool) { }  
  
// Called when the user taps an Action button.  
safariViewController(_ controller: SFSafariViewController, activityItemsFor URL: URL, title:  
String?) -> [UIActivity] { }
```

Elemente zur Safari-Leseliste hinzufügen

Sie können der `addItem` eines Benutzers in Safari Elemente hinzufügen, indem Sie die `addItem` Methode für das `SSReadingList` Singleton `SSReadingList` .

```
let readingList = SSReadingList.default()  
readingList?.addItem(with: yourURL, title: "optional title", previewText: "optional preview  
text")
```

Die Standard-Leseliste kann `nil` wenn der Zugriff auf die Leseliste nicht zulässig ist.

Zusätzlich können Sie prüfen, ob die Leseliste eine URL `supportsURL` indem Sie `supportURL` aufrufen.

```
SSReadingList.default().supportsURL(URL(string: "https://example.com")!)
```

Dies gibt entweder `true` oder `false` zeigt an, ob die angegebene URL von Safari Reading List unterstützt wird. Verwenden Sie dies beispielsweise, um zu bestimmen, ob eine Schaltfläche angezeigt werden soll, um der Leseliste eine URL hinzuzufügen.

Öffnen Sie eine URL mit `SafariViewController`

Vergessen Sie nicht, zuerst das notwendige Framework zu importieren.

```
import SafariServices
//Objective-C
@import SafariServices;
```

`SafariViewController` eine `SafariViewController` Instanz.

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!)
//Objective-C
@import SafariServices;
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL];
```

Optional können Sie `SafariViewController` auch anweisen, nach dem Laden den Lesemodus zu aktivieren.

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!, entersReaderIfAvailable:
true)
//Objective-C
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL
entersReaderIfAvailable:YES];
```

Präsentieren Sie den View Controller.

```
present(safariVC, animated: true, completion: nil)
//Objective-C
[self presentViewController:sfvc animated:YES completion:nil];
```

Safari-Dienstleistungen online lesen: <https://riptutorial.com/de/ios/topic/1371/safari-dienstleistungen>

Kapitel 134: Schlüsselbund

Syntax

- `kSecClassGenericPassword` // Ein Wertschlüssel, der ein Nicht-Internet-Kennwort darstellt
- `kSecClassInternetPassword` // Ein Wertschlüssel, der ein Internetkennwort darstellt
- `kSecClassCertificate` // Ein Wertschlüssel, der ein Zertifikat darstellt
- `kSecClassCertificate` // Ein Wertschlüssel, der einen Schlüssel darstellt
- `kSecClassIdentity` // Ein Wertschlüssel, der eine Identität darstellt, dh ein Zertifikat plus einen Schlüssel

Bemerkungen

iOS speichert private Informationen wie Kennwörter, Verschlüsselungsschlüssel, Zertifikate und Identitäten in einem sicheren Speicherbereich, der als Schlüsselbund bezeichnet wird. Dieser Speicherbereich wird vollständig von einem Co-Prozessor namens Secure Enclave verwaltet, der in den Anwendungsprozessor eingebettet ist. Da der Schlüsselbund unter iOS auf Sandbox gesetzt wird, können Schlüsselbundelemente nur von der Anwendung abgerufen werden, die sie zuerst dort ablegt.

In einigen Fällen müssen Sie Keychain Sharing in Xcode aktivieren, um Fehler zu vermeiden.

Um mit dem Schlüsselbund zu interagieren, verwenden wir ein als Keychain Services bezeichnetes Rahmenwerk. Weitere Informationen finden Sie [im Programmierhandbuch für Apple Keychain Services](#).

Da sich Keychain Services unterhalb der `Foundation` Ebene befindet, können nur `CoreFoundation` Typen verwendet werden. Daher werden die meisten Objekte intern als `CFDictionary` dargestellt, das `CFDictionary CFString` als Schlüssel und eine Vielzahl von `CoreFoundation` Typen als Werte enthält.

Keychain Services sind zwar Bestandteil des `Security Frameworks`, der Import von `Foundation` ist jedoch normalerweise eine gute Option, da er einige Hilfsfunktionen im Backend enthält.

Wenn Sie nicht direkt mit Keychain Services arbeiten möchten, stellt Apple das Swift-Beispielprojekt "[Generic Keychain](#)" bereit, das Swift-Typen bereitstellt, die Keychain Services im Hintergrund verwenden.

Examples

Passwort zum Schlüsselbund hinzufügen

Jedes Schlüsselbundelement wird meistens als `CFDictionary`. Sie können `NSDictionary` jedoch einfach in Objective-C verwenden und Bridging nutzen. In Swift können Sie `Dictionary` und explizit in `CFDictionary`.

Sie können ein Kennwort mit dem folgenden Wörterbuch erstellen:

Schnell

```
var dict = [String : AnyObject]()
```

Zunächst benötigen Sie ein Schlüssel / Wert-Paar, das den Schlüsselbund darüber informiert, dass dies ein Kennwort ist. Da unser `CFString` ein `String`, müssen wir jeden `CFString` in Swift 3 explizit in einen `String` umwandeln. `CFString` kann nicht als Schlüssel für ein Swift-Dictionary verwendet werden, da es nicht Hash-fähig ist.

Schnell

```
dict[kSecClass as String] = kSecClassGenericPassword
```

Als nächstes kann unser Passwort eine Reihe von Attributen haben, die es beschreiben und uns helfen, es später zu finden. [Hier ist eine Liste von Attributen für generische Kennwörter](#).

Schnell

```
// The password will only be accessible when the device is unlocked
dict[kSecAttrAccessible as String] = kSecAttrAccessibleWhenUnlocked
// Label may help you find it later
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
// Service name
dict[kSecAttrService as String] = "MyService" as CFString
```

Schließlich benötigen wir unsere tatsächlichen privaten Daten. Denken Sie daran, dies nicht zu lange im Gedächtnis zu behalten. Dies muss `CFData`.

Schnell

```
dict[kSecValueData as String] = "my_password!".data(using: .utf8) as! CFData
```

Schließlich möchte die Add-Funktion der Schlüsselbunddienste wissen, wie das neu erstellte Schlüsselbundelement zurückgegeben werden soll. Da Sie die Daten nicht lange im Speicher behalten sollten, können Sie nur die Attribute zurückgeben:

Schnell

```
dict[kSecReturnAttributes as String] = kCFBooleanTrue
```

Jetzt haben wir unseren Artikel konstruiert. Lassen Sie uns es hinzufügen:

Schnell

```
var result: AnyObject?  
let status = withUnsafeMutablePointer(to: &result) {  
    SecItemAdd(dict as CFDictionary, UnsafeMutablePointer($0))  
}  
let newAttributes = result as! Dictionary<String, AnyObject>
```

Dadurch werden die neuen Attribute in das `result` eingefügt. `SecItemAdd` das von uns erstellte Wörterbuch sowie einen Hinweis darauf, wo wir unser Ergebnis haben möchten. Die Funktion gibt dann einen `OSStatus`, der den Erfolg oder einen Fehlercode angibt. Ergebniscodes werden [hier](#) beschrieben.

Ein Passwort im Schlüsselbund finden

Um eine Abfrage zu `CFDictionary`, müssen wir sie als `CFDictionary`. Sie können `NSDictionary` in Objective-C oder `Dictionary` in Swift verwenden und in `CFDictionary`.

Wir brauchen einen Klassenschlüssel:

Schnell

```
var dict = [String : AnyObject]()  
dict[kSecClass as String] = kSecClassGenericPassword
```

Als Nächstes können wir Attribute angeben, um unsere Suche einzugrenzen:

Schnell

```
// Label  
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString  
// Username  
dict[kSecAttrAccount as String] = "My Name" as CFString  
// Service name  
dict[kSecAttrService as String] = "MyService" as CFString
```

Wir können auch spezielle Suchmodifizierungsschlüssel angeben, die [hier](#) beschrieben werden.

Zum Schluss müssen wir noch sagen, wie wir möchten, dass unsere Daten zurückgegeben werden. Nachfolgend fordern wir an, dass nur das private Kennwort selbst als `CFData` Objekt zurückgegeben wird:

Schnell

```
dict[kSecReturnData as String] = kCFBooleanTrue
```

Nun suchen wir:

Schnell

```
var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(dict as CFDictionary, UnsafeMutablePointer($0))
}
// Don't keep this in memory for long!!
let password = String(data: queryResult as! Data, encoding: .utf8)!
```

Hier führt `SecItemCopyMatching` ein Abfragewörterbuch und einen Zeiger auf, wohin das Ergebnis gehen soll. Es gibt einen `OSStatus` mit Ergebniscodes zurück. [Hier](#) sind die Möglichkeiten.

Aktualisieren eines Passworts im Schlüsselbund

Wie üblich benötigen wir zuerst ein `CFDictionary`, um das Element `CFDictionary`, das wir aktualisieren möchten. Dieser muss alle alten Werte für das Element enthalten, einschließlich der alten privaten Daten. Dann benötigt es ein `CFDictionary` aller Attribute oder der Daten selbst, die Sie ändern möchten.

Zuerst erstellen wir einen Klassenschlüssel und eine Liste von Attributen. Diese Attribute können unsere Suche einschränken, Sie müssen jedoch alle Attribute und alte Werte angeben, wenn Sie sie ändern.

Schnell

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

Nun müssen wir die alten Daten hinzufügen:

Schnell

```
dict[kSecValueData as String] = "my_password!".data(using: .utf8) as! CFData
```

Jetzt erstellen wir dieselben Attribute, aber ein anderes Passwort:

Schnell

```
var newDict = [String : AnyObject]()
newDict[kSecClass as String] = kSecClassGenericPassword
// Label
newDict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
newDict[kSecAttrAccount as String] = "My Name" as CFString
// New password
newDict[kSecValueData as String] = "new_password!".data(using: .utf8) as! CFData
```

Jetzt geben wir es einfach an Keychain Services weiter:

Schnell

```
let status = SecItemUpdate(dict as CFDictionary, newDict as CFDictionary)
```

`SecItemUpdate` gibt einen Statuscode zurück. Ergebnisse werden [hier](#) beschrieben.

Passwort aus dem Schlüsselbund entfernen

Wir benötigen nur eine Sache, um ein Element aus dem Schlüsselbund zu löschen: ein `CFDictionary` mit Attributen, die die zu löschenden Elemente beschreiben. Alle Elemente, die mit dem Abfragewörterbuch übereinstimmen, werden dauerhaft gelöscht. Wenn Sie also nur ein einzelnes Element löschen möchten, stellen Sie sicher, dass Sie Ihre Abfrage genau angeben. Wie immer können wir ein `NSDictionary` in Objective-C oder in Swift verwenden, wir können ein `Dictionary` und dann in `CFDictionary`.

Ein Abfragewörterbuch enthält in diesem Zusammenhang ausschließlich einen Klassenschlüssel zum Beschreiben des Elements und Attribute zum Beschreiben von Informationen zum Element. Das `kSecMatchCaseInsensitive` von Sucheinschränkungen wie `kSecMatchCaseInsensitive` ist nicht zulässig.

Schnell

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

Und jetzt können wir es einfach entfernen:

Schnell

```
let status = SecItemDelete(dict as CFDictionary)
```

`SecItemDelete` gibt einen `OSStatus`. Ergebniscode werden [hier](#) beschrieben.

Keychain Hinzufügen, Aktualisieren, Entfernen und Suchen mit einer Datei.

Keychain.h

```
#import <Foundation/Foundation.h>
typedef void (^KeychainOperationBlock)(BOOL successfulOperation, NSData *data, OSStatus status);

@interface Keychain : NSObject

-(id) initWithService:(NSString *) service_ withGroup:(NSString*)group_;

-(void)insertKey:(NSString *)key withData:(NSData *)data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)updateKey:(NSString*)key withData:(NSData*) data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)removeDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;
-(void)findDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;

@end
```

Keychain.m

```
#import "Keychain.h"
#import <Security/Security.h>

@implementation Keychain

{
    NSString * keychainService;
    NSString * keychainGroup;
}

-(id) initWithService:(NSString *)service withGroup:(NSString*)group
{
    self =[super init];
    if(self) {
        keychainService = [NSString stringWithString:service];
        if(group) {
            keychainGroup = [NSString stringWithString:group];
        }
    }

    return self;
}

-(void)insertKey:(NSString *)key
```



```

        withData:(NSData *)data
withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dict =[self prepareDict:key];
    [dict setObject:data forKey:(__bridge id)kSecValueData];
    [dict setObject:keychainService forKey:(id)kSecAttrService];

    OSStatus status = SecItemAdd((__bridge CFDictionaryRef)dict, NULL);
    if(errSecSuccess != status) {
        DLog(@"Unable add item with key =%@ error:%d",key, (int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    }
    if (status == errSecDuplicateItem) {
        [self updateKey:key withData:data withCompletion:^(BOOL successfulOperation, NSData
*updateData, OSStatus updateStatus) {
            if (completionBlock) {
                completionBlock(successfulOperation, updateData, updateStatus);
            }
            DLog(@"Found duplication item -- updating key with data");
        }];
    }
}

-(void)findDataForKey:(NSString *)key
withCompletionBlock:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary *dict = [self prepareDict:key];
    [dict setObject:(__bridge id)kSecMatchLimitOne forKey:(__bridge id)kSecMatchLimit];
    [dict setObject:keychainService forKey:(id)kSecAttrService];
    [dict setObject:(id)kCFBooleanTrue forKey:(__bridge id)kSecReturnData];
    CFTypeRef result = NULL;
    OSStatus status = SecItemCopyMatching((__bridge CFDictionaryRef)dict,&result);

    if( status != errSecSuccess) {
        DLog(@"Unable to fetch item for key %@ with error:%d",key, (int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    } else {
        if (completionBlock) {
            completionBlock(errSecSuccess == status, (__bridge NSData *)result, status);
        }
    }
}

-(void)updateKey:(NSString *)key
withData:(NSData *)data
withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dictKey =[self prepareDict:key];

    NSMutableDictionary * dictUpdate =[NSMutableDictionary alloc] init];
    [dictUpdate setObject:data forKey:(__bridge id)kSecValueData];
    [dictUpdate setObject:keychainService forKey:(id)kSecAttrService];
    OSStatus status = SecItemUpdate((__bridge CFDictionaryRef)dictKey, (__bridge
CFDictionaryRef)dictUpdate);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key, (int)status);
    }
}

```

```

    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

-(void)removeDataForKey:(NSString *)key
withCompletionBlock:(KeychainOperationBlock)completionBlock {
    NSMutableDictionary *dict = [self prepareDict:key];
    OSStatus status = SecItemDelete((__bridge CFDictionaryRef)dict);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key,(int)status);
    }
    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

#pragma mark Internal methods

-(NSMutableDictionary*) prepareDict:(NSString *) key {

    NSMutableDictionary *dict = [[NSMutableDictionary alloc] init];
    [dict setObject:(__bridge id)kSecClassGenericPassword forKey:(__bridge id)kSecClass];

    NSData *encodedKey = [key dataUsingEncoding:NSUTF8StringEncoding];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrGeneric];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrAccount];
    [dict setObject:keychainService forKey:(__bridge id)kSecAttrService];
    [dict setObject:(__bridge id)kSecAttrAccessibleAlwaysThisDeviceOnly forKey:(__bridge
id)kSecAttrAccessible];

    //This is for sharing data across apps
    if(keychainGroup != nil) {
        [dict setObject:keychainGroup forKey:(__bridge id)kSecAttrAccessGroup];
    }

    return dict;
}

@end

```

Schlüsselbund-Zugriffskontrolle (TouchID mit Passwort-Fallback)

Mit Keychain können Elemente mit dem speziellen SecAccessControl-Attribut gespeichert werden, mit dem Artikel nur dann von Keychain abgerufen werden können, wenn der Benutzer mit der Touch-ID (oder einem Passcode, wenn ein solcher Fallback zulässig ist) authentifiziert wird. Die App wird nur benachrichtigt, ob die Authentifizierung erfolgreich war oder nicht, die gesamte Benutzeroberfläche wird von iOS verwaltet.

Zunächst sollte das SecAccessControl-Objekt erstellt werden:

Schnell

```
let error: Unmanaged<CFError>?
```

```
guard let accessControl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, .userPresence, &error) else {
    fatalError("Something went wrong")
}
```

Fügen Sie es anschließend mit dem Schlüssel `kSecAttrAccessControl` (der sich gegenseitig mit dem in anderen Beispielen verwendeten Schlüssel `kSecAttrAccessible` ausschließt) dem Wörterbuch hinzu:

Schnell

```
var dictionary = [String : Any]()

dictionary[kSecClass as String] = kSecClassGenericPassword
dictionary[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
dictionary[kSecAttrAccount as String] = "My Name" as CFString
dictionary[kSecValueData as String] = "new_password!".data(using: .utf8) as! CFData
dictionary[kSecAttrAccessControl as String] = accessControl
```

Und speichern Sie es wie zuvor:

Schnell

```
let lastResultCode = SecItemAdd(query as CFDictionary, nil)
```

Um auf gespeicherte Daten zuzugreifen, fragen Sie einfach Keychain nach einem Schlüssel ab. Keychain Services zeigt dem Benutzer ein Authentifizierungsfeld an und gibt Daten oder Null zurück, je nachdem, ob ein geeigneter Fingerabdruck bereitgestellt oder ein Passcode festgelegt wurde.

Optional kann eine Eingabeaufforderungszeichenfolge angegeben werden:

Schnell

```
var query = [String: Any]()

query[kSecClass as String] = kSecClassGenericPassword
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecAttrAccount as String] = "My Name" as CFString
query[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
query[kSecUseOperationPrompt as String] = "Please put your fingers on that button" as CFString

var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(query as CFDictionary, UnsafeMutablePointer($0))
}
```

Achten Sie darauf, dass der `status err` wenn der Benutzer die Autorisierung abgelehnt, abgebrochen oder nicht bestanden hat.

Schnell

```
if status == noErr {
    let password = String(data: queryResult as! Data, encoding: .utf8)!
    print("Password: \(password)")
} else {
    print("Authorization not passed")
}
```

Schlüsselbund online lesen: <https://riptutorial.com/de/ios/topic/6839/schlüsselbund>

Kapitel 135: Schlüsselwert-Schlüsselwertbeobachtung

Bemerkungen

KVC : - Schlüsselwertcodierung

Normalerweise wird auf Instanzvariablen über Eigenschaften oder Zugriffsmethoden zugegriffen. KVC bietet jedoch eine andere Möglichkeit, auf Variablen in Form von Zeichenfolgen zuzugreifen. Auf diese Weise verhält sich Ihre Klasse wie ein Wörterbuch, und der Name Ihrer Eigenschaft, zum Beispiel "age", wird zum Schlüssel und Wert, den die Eigenschaft enthält, wird zum Wert für diesen Schlüssel.

```
For example, you have employee class with "age" property. Normally we access like this.  
emp.age = @"20";  
NSString age = emp.age;  
  
But KVC works like this:  
[emp valueForKey:@"age"];  
[emp setValue:@"25" forKey:@"age"];
```

KVO : - Schlüsselwertbeobachter

Der Mechanismus, durch den Objekte benachrichtigt werden, wenn eine Eigenschaft geändert wird, wird als KVO bezeichnet. Beispiel: Tastaturbenachrichtigung

Beispielsweise ist ein Personenobjekt daran interessiert, eine Benachrichtigung zu erhalten, wenn die accountBalance-Eigenschaft im BankAccount-Objekt geändert wird. Um dies zu erreichen, muss sich Person Object als Beobachter der Eigenschaft accountBalance der BankAccount registrieren, indem ein addObserver: forKeyPath: options: context: message gesendet wird.

Examples

Verwendung des Kontextes für die KVO-Beobachtung

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object  
change:(NSDictionary<NSString *,id> *)change context:(void *)context
```

Der Kontext ist wichtig, wenn Sie Ihre Klasse für andere Benutzer versenden. Mit dem Kontext kann Ihr Klassenbeobachter überprüfen, ob er von Ihnen aufgerufen wird.

Das Problem, einen Beobachter nicht zu übergeben, besteht darin, dass, wenn jemand Ihre Klasse unterordnet und einen Beobachter für dasselbe Objekt, denselben Schlüssel registriert und keinen Kontext übergibt, der Superklassenbeobachter mehrfach aufgerufen werden kann.

Eine für Ihre Verwendung eindeutige und interne Variable ist ein guter Kontext.

Für mehr Informationen.

Wichtigkeit und guter Kontext

Beobachten einer Eigenschaft einer NSObject-Unterklasse

Die meisten KVO- und KVC-Funktionen sind bereits standardmäßig in allen `NSObject` Unterklassen `NSObject` .

Um eine Eigenschaft namens `firstName` eines Objekts namens `personObject` führen Sie dies in der beobachtenden Klasse aus:

```
[personObject addObserver:self
                forKeyPath:@"firstName"
                options:NSKeyValueObservingOptionNew
                context:nil];
```

Das Objekt, auf das sich `self` im obigen Code bezieht, erhält dann eine `observeValueForKeyPath:ofObject:change:context:` -Nachricht, wenn sich der beobachtete Schlüsselpfad ändert.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
    ofObject:(id)object
    change:(NSDictionary<NSString *,id> *)change
    context:(void *)context
{
    NSLog(@"new value of %@ is: %@", keyPath, change[NSKeyValueChangeNewKey]);
}
```

"Schlüsselpfad" ist ein KVC-Begriff. `NSObject` Unterklassen implementieren standardmäßig die KVC-Funktionalität.

Auf eine Instanzvariable mit dem Namen `_firstName` kann über den Schlüsselpfad `@"firstName"` .

Eine Getter-Methode mit dem Namen `firstName` wird aufgerufen, wenn auf den Schlüsselpfad `@"firstName"` wird, unabhängig davon, `_firstName` es eine `_firstName` oder `setFirstName` Setter-Methode `setFirstName` .

Schlüsselwert-Schlüsselwertbeobachtung online lesen:

<https://riptutorial.com/de/ios/topic/3493/schlüsselwert-schlüsselwertbeobachtung>

Kapitel 136: Schneiden Sie einen UIImage in einen Kreis

Examples

Schneiden Sie ein Bild in einen Kreis - Ziel C

importiere `#include <math.h>`

Der Code in `viewDidLoad` oder `loadView` sollte in etwa wie folgt aussehen

```
- (void)loadView
{
    [super loadView];
    UIImageView *imageView=[[UIImageView alloc]initWithFrame:CGRectMake(0, 50, 320, 320)];
    [self.view addSubview:imageView];
    UIImage *image=[UIImage imageNamed:@"Dubai-Photos-Images-Travel-Tourist-Images-Pictures-800x600.jpg"];
    imageView.image=[self circularScaleAndCropImage:[UIImage imageNamed:@"Dubai-Photos-Images-Travel-Tourist-Images-Pictures-800x600.jpg"] frame:CGRectMake(0, 0, 320, 320)];
}
```

Schließlich ist die Funktion, die das schwere Anheben von `circularScaleAndCropImage` ausführt, wie nachstehend definiert

```
- (UIImage*)circularScaleAndCropImage:(UIImage*)image frame:(CGRect)frame {
    // This function returns a newImage, based on image, that has been:
    // - scaled to fit in (CGRect) rect
    // - and cropped within a circle of radius: rectWidth/2

    //Create the bitmap graphics context
    UIGraphicsBeginImageContextWithOptions(CGSizeMake(frame.size.width, frame.size.height),
    NO, 0.0);
    CGContextRef context = UIGraphicsGetCurrentContext();

    //Get the width and heights
    CGFloat imageWidth = image.size.width;
    CGFloat imageHeight = image.size.height;
    CGFloat rectWidth = frame.size.width;
    CGFloat rectHeight = frame.size.height;

    //Calculate the scale factor
    CGFloat scaleFactorX = rectWidth/imageWidth;
    CGFloat scaleFactorY = rectHeight/imageHeight;

    //Calculate the centre of the circle
    CGFloat imageCentreX = rectWidth/2;
    CGFloat imageCentreY = rectHeight/2;

    // Create and CLIP to a CIRCULAR Path
    // (This could be replaced with any closed path if you want a different shaped clip)
    CGFloat radius = rectWidth/2;
    CGContextBeginPath (context);
```

```

CGContextAddArc (context, imageCentreX, imageCentreY, radius, 0, 2*M_PI, 0);
CGContextClosePath (context);
CGContextClip (context);

//Set the SCALE factor for the graphics context
//All future draw calls will be scaled by this factor
CGContextScaleCTM (context, scaleFactorX, scaleFactorY);

// Draw the IMAGE
CGRect myRect = CGRectMake(0, 0, imageWidth, imageHeight);
[image drawInRect:myRect];

UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();

return newImage;
}

```

SWIFT 3 Beispiel

```

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    let imageView = UIImageView(frame: CGRect(x: CGFloat(0), y: CGFloat(50), width:
CGFloat(320), height: CGFloat(320)))
    view.addSubview(imageView)
    let image = UIImage(named: "Dubai-Photos-Images-Travel-Tourist-Images-Pictures-
800x600.jpg")
    imageView.image = circularScaleAndCropImage(UIImage(named: "Dubai-Photos-Images-
Travel-Tourist-Images-Pictures-800x600.jpg")!, frame: CGRect(x: CGFloat(0), y: CGFloat(0),
width: CGFloat(100), height: CGFloat(100)))
}

```

Schließlich ist die Funktion, die das schwere Anheben von `circleScaleAndCropImage` ausführt, wie nachstehend definiert

```

func circularScaleAndCropImage(_ image: UIImage, frame: CGRect) -> UIImage{
    // This function returns a newImage, based on image, that has been:
    // - scaled to fit in (CGRect) rect
    // - and cropped within a circle of radius: rectWidth/2
    //Create the bitmap graphics context
    UIGraphicsBeginImageContextWithOptions(CGSize(width: CGFloat(frame.size.width),
height: CGFloat(frame.size.height)), false, 0.0)
    let context: CGContext? = UIGraphicsGetCurrentContext()
    //Get the width and heights
    let imageWidth: CGFloat = image.size.width
    let imageHeight: CGFloat = image.size.height
    let rectWidth: CGFloat = frame.size.width
    let rectHeight: CGFloat = frame.size.height
    //Calculate the scale factor
    let scaleFactorX: CGFloat = rectWidth / imageWidth
    let scaleFactorY: CGFloat = rectHeight / imageHeight
    //Calculate the centre of the circle
    let imageCentreX: CGFloat = rectWidth / 2
    let imageCentreY: CGFloat = rectHeight / 2
    // Create and CLIP to a CIRCULAR Path
    // (This could be replaced with any closed path if you want a different shaped clip)
    let radius: CGFloat = rectWidth / 2

```



```
context?.beginPath()
context?.addArc(center: CGPoint(x: imageCentreX, y: imageCentreY), radius: radius,
startAngle: CGFloat(0), endAngle: CGFloat(2 * Float.pi), clockwise: false)
context?.closePath()
context?.clip()
//Set the SCALE factor for the graphics context
//All future draw calls will be scaled by this factor
context?.scaleBy(x: scaleFactorX, y: scaleFactorY)
// Draw the IMAGE
let myRect = CGRect(x: CGFloat(0), y: CGFloat(0), width: imageWidth, height:
imageHeight)
image.draw(in: myRect)
let newImage: UIImage? = UIGraphicsGetImageFromCurrentImageContext()
UIGraphicsEndImageContext()
return newImage!
}
```

Schneiden Sie einen UIImage in einen Kreis online lesen:

<https://riptutorial.com/de/ios/topic/7222/schneiden-sie-einen-uiimage-in-einen-kreis>

Kapitel 137: Schnelle und Objective-C-Interoperabilität

Examples

Verwenden von Objective-C-Klassen in Swift

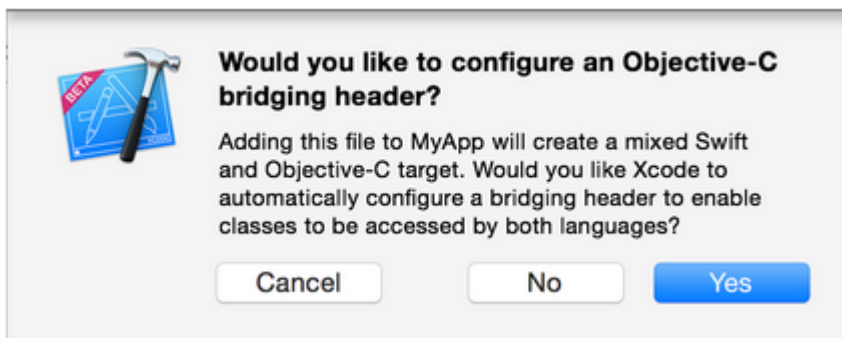
Wenn Sie über eine vorhandene Klasse verfügen, die Sie verwenden möchten, führen Sie **Schritt 2 aus** und fahren Sie dann mit **Schritt 5 fort** . (In einigen Fällen musste ich einer älteren ObjC-Datei explizit `#import <Foundation/Foundation.h` hinzufügen.)

Schritt 1: Fügen Sie die Objective-C-Implementierung hinzu

Fügen `.m` Ihrer Klasse eine `.m` Datei hinzu, und nennen Sie sie `CustomObject.m`

Schritt 2: Fügen Sie den Bridging-Header hinzu

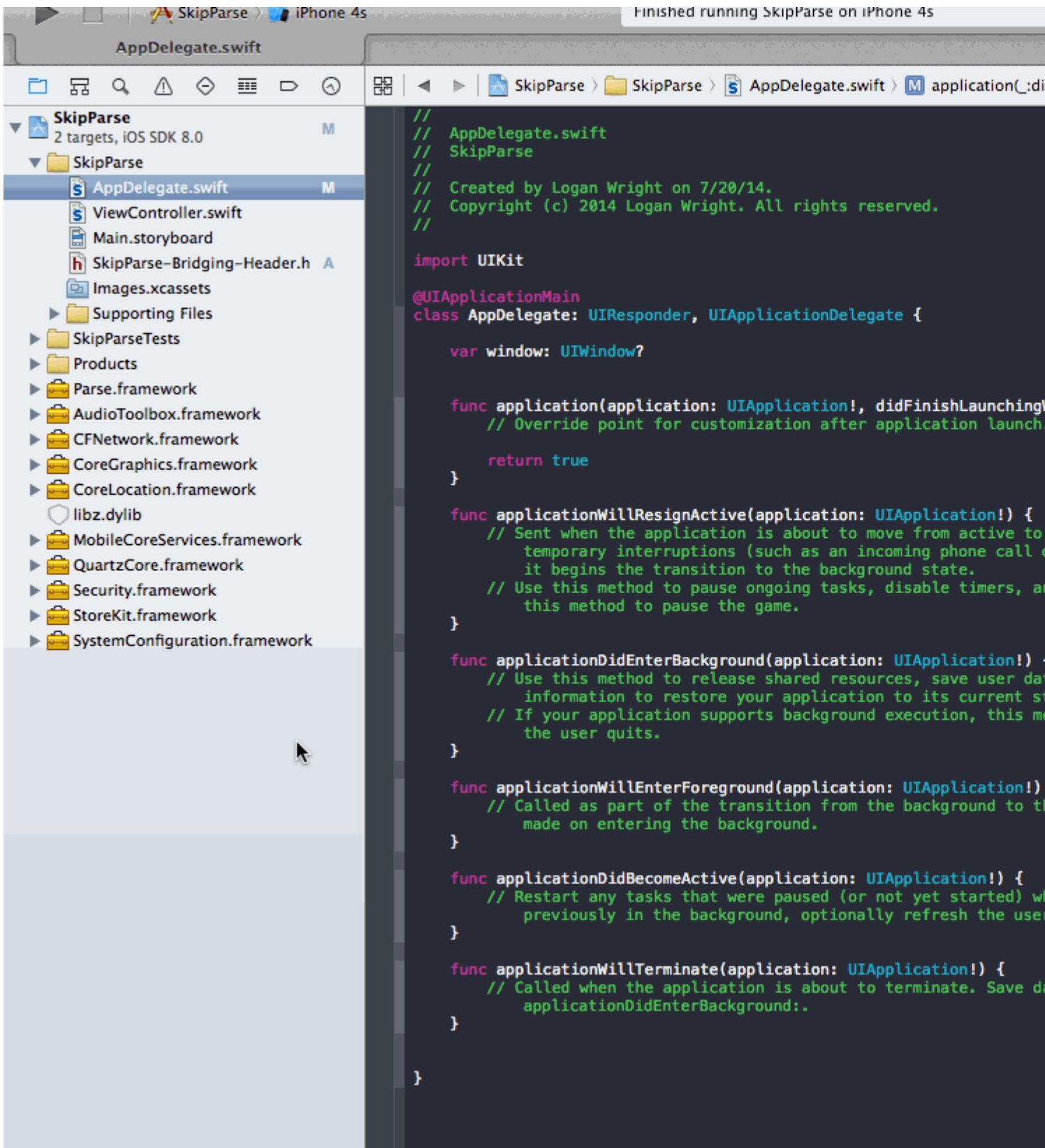
Wenn Sie Ihre `.m` Datei hinzufügen, werden Sie wahrscheinlich mit einer Eingabeaufforderung getroffen, die wie `.m` aussieht:



Klicken Sie auf **JA** !

Wenn die Aufforderung nicht angezeigt oder der Bridging-Header versehentlich gelöscht wurde, fügen `.h` Ihrem Projekt eine neue `.h` Datei hinzu, und nennen Sie sie `<#YourProjectName#>-Bridging-Header.h`

In einigen Situationen, insbesondere bei der Arbeit mit ObjC-Frameworks, fügen Sie eine Objective-C-Klasse nicht explizit hinzu, und Xcode kann den Linker nicht finden. In diesem Fall erstellen Sie Ihre `.h` Datei mit dem oben genannten Namen und stellen Sie sicher, dass Sie den Pfad in den Projekteinstellungen Ihres Ziels wie folgt verknüpfen:



Hinweis

Es ist `$(SRCROOT)` Ihr Projekt mit dem `$(SRCROOT)` zu verknüpfen. Wenn Sie Ihr Projekt verschieben oder mit anderen über ein Remote-Repo daran arbeiten, funktioniert es trotzdem. `$(SRCROOT)` kann als das Verzeichnis betrachtet werden, das Ihre `.xcodproj`-Datei enthält. Es könnte so aussehen:

```
$(SRCROOT)/Folder/Folder/<#YourProjectName#>-Bridging-Header.h
```

Schritt 3: Objective-C Header hinzufügen - .h

Fügen Sie eine weitere .h Datei hinzu und nennen Sie sie `CustomObject.h`

Schritt 4: Erstellen Sie Ihre Objective-C-Klasse

In `CustomObject.h`

```
#import <Foundation/Foundation.h>

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

- (void) someMethod;

@end
```

In `CustomObject.m`

```
#import "CustomObject.h"

@implementation CustomObject

- (void) someMethod {
    NSLog(@"SomeMethod Ran");
}

@end
```

Schritt 5: Klasse zum Bridging-Header hinzufügen

In `YourProject-Bridging-Header.h`:

```
#import "CustomObject.h"
```

Schritt 6: Verwenden Sie Ihr Objekt

In `SomeSwiftFile.swift`:

```
var instanceOfCustomObject: CustomObject = CustomObject()
instanceOfCustomObject.someProperty = "Hello World"
println(instanceOfCustomObject.someProperty)
instanceOfCustomObject.someMethod()
```

Es ist nicht notwendig, explizit zu importieren, dafür ist der Bridging-Header gedacht.

Verwendung schneller Klassen in Objective-C

Schritt 1: Neue Swift-Klasse erstellen

Fügen `.swift` Ihrem Projekt eine `.swift` Datei hinzu und nennen Sie sie `MySwiftObject.swift`

In `MySwiftObject.swift` :

```
import Foundation

class MySwiftObject : NSObject {

    var someProperty: AnyObject = "Some Initializer Val"

    init() {}

    func someFunction(someArg:AnyObject) -> String {
        var returnVal = "You sent me \(someArg)"
        return returnVal
    }

}
```

Schritt 2: Importieren Sie Swift-Dateien in die ObjC-Klasse

In `SomeRandomClass.m` :

```
#import "<#YourProjectName#>-Swift.h"
```

Die Datei: `<#YourProjectName#>-Swift.h` sollte bereits in Ihrem Projekt automatisch erstellt werden, auch wenn Sie sie nicht sehen können.

Schritt 3: Verwenden Sie Ihre Klasse

```
MySwiftObject * myOb = [MySwiftObject new];
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
myOb.someProperty = @"Hello World";
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
NSString * retString = [myOb someFunction:@"Arg"];
NSLog(@"RetString: %@", retString);
```

Hinweis:

1. CodeCompletion verhielt sich nicht so genau, wie ich es gerne hätte. Auf meinem System schien ein schneller Build mit "cmd + r" dem Swift dabei zu helfen, einen Teil des Objc-Codes zu finden und umgekehrt.
2. Wenn Sie einem älteren Projekt eine `.swift` Datei hinzufügen und eine Fehlermeldung erhalten: `dyld: Library not loaded: @rpath/libswift_stdlib_core.dylib` , [starten Sie Xcode](#) vollständig neu.
3. Während ursprünglich reine Swift-Klassen in Objective-C mit dem `@objc` Präfix verwendet werden konnten, ist dies nach Swift 2.0 nicht mehr möglich. Die ursprüngliche Erklärung finden Sie unter Bearbeitungshistorie. Wenn diese Funktion in zukünftigen Swift-Versionen wieder aktiviert wird, wird die Antwort entsprechend aktualisiert.

Schnelle und Objective-C-Interoperabilität online lesen:

<https://riptutorial.com/de/ios/topic/1497/schnelle-und-objective-c-interoperabilitat>

Kapitel 138: Segues

Examples

Ein Überblick

Aus der Apple-Dokumentation:

Ein UIStoryboardSegue-Objekt ist dafür verantwortlich , **den visuellen Übergang zwischen zwei Ansichtsccontrollern** auszuführen. Darüber hinaus werden Segue-Objekte verwendet, um den Übergang von einem View-Controller zu einem anderen vorzubereiten. **Segue-Objekte enthalten Informationen zu den View-Controllern, die an einem Übergang beteiligt sind** . Wenn ein Segment ausgelöst wird, der visuelle Übergang jedoch nicht erfolgt, ruft die Storyboard-Laufzeitumgebung die Methode `prepareForSegue(sender:)` des aktuellen View-Controllers auf, damit er alle erforderlichen Daten an den View-Controller übergeben kann, der gerade angezeigt wird.

Attribute

Schnell

```
sourceViewController: UIViewController {get}
destinationViewController: UIViewController {get}
identifier: String? {get}
```

Verweise:

- [UIViewController-Klassenreferenz](#)
- [UIStoryboardSegue-Klassenreferenz](#)

Vorbereiten des View Controllers vor dem Auslösen eines Segue

PrepareForSegue :

```
func prepareForSegue(_ segue:UIStoryboardSegue, sender sender:AnyObject?)
```

Benachrichtigt den View-Controller, dass ein Segment gerade ausgeführt wird

Parameter

segue : Das segue-Objekt.

Sender: Das Objekt, das das Segment initialisiert hat.

Beispiel in Swift

Führen Sie eine Aufgabe aus, wenn der Bezeichner des Segues "SomeSpecificIdentifier" lautet.

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "SomeSpecificIdentifier" {
        //- Do specific task
    }
}
```

Entscheidung, ob eine aufgerufene Segue ausgeführt werden soll.

ShouldPerformSegueWithIdentifier :

```
func shouldPerformSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?) -> Bool
```

Legt fest, ob das Segment mit der angegebenen Kennung ausgeführt werden soll.

Parameter

Bezeichner: Zeichenfolge, die den ausgelösten Abschnitt angibt

Sender: Das Objekt, das das Segment initialisiert hat.

Beispiel in Swift

Führen Sie Segue nur durch, wenn der Bezeichner "SomeSpecificIdentifier" lautet.

```
override func shouldPerformSegueWithIdentifier(identifier:String, sender:AnyObject?) -> Bool {
    if identifier == "SomeSpecificIdentifier" {
        return true
    }
    return false
}
```

Verwenden von Segues, um im Navigationsstapel rückwärts zu navigieren

Wickeln Sie Segues auf

Abwicklungssegmente geben Ihnen die Möglichkeit, den Navigationsstapel "abzuwickeln" und ein Ziel anzugeben, zu dem Sie zurückkehren möchten. Die Signatur dieser Funktion ist der Schlüssel zum Erkennen von Interface Builder. **Es muss einen Rückgabewert von IBAction haben und einen Parameter von**

UIStoryboardSegue haben . Der Name der Funktion spielt keine Rolle. Tatsächlich muss die Funktion gar nichts tun. Es ist nur als Marker da, an dem UIViewController das Ziel des Unwind Segue ist. [Quelle] [1]

Erforderliche Signatur eines Abwicklungsabschnitts

Ziel c:

```
-(IBAction)prepareForUnwind:(UIStoryboardSegue *) segue {  
}
```

Schnell:

```
@IBAction func prepareForUnwind(segue: UIStoryboardSegue) {  
}
```

Programmgesteuert auslösen

PerformSegueWithIdentifier:

```
func performSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?)
```

Initiiert das Segment mit der angegebenen Kennung aus der Storyboard-Datei des aktuellen View-Controllers

Parameter

Bezeichner : Zeichenfolge, die den ausgelösten Abschnitt angibt

Sender : Das Objekt, das das Segue initiiert.

Beispiel in Swift

Ausführen eines Segues mit der Kennung "SomeSpecificIdentifier" aus einer Tabellenansicht-Zeilenauswahl

```
func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    performSegueWithIdentifier("SomeSpecificIdentifier", sender: indexPath.item)  
}
```

Segues online lesen: <https://riptutorial.com/de/ios/topic/5575/segues>

Kapitel 139: Selektive UIView-Ecken abrunden

Examples

Ziel-C-Code, um die ausgewählte Ecke eines UIView abzurunden

Importieren Sie zunächst `#import <QuartzCore/QuartzCore.h>` in Ihre ViewController-Klasse. So setze ich meine Ansicht in Code

```
UIView *view1=[[UIView alloc]init];
view1.backgroundColor=[UIColor colorWithRed:255/255.0 green:193/255.0 blue:72/255.0
alpha:1.0];
CGRect view1Frame = view1.frame;
view1Frame.size.width = SCREEN_WIDTH*0.97;
view1Frame.size.height = SCREEN_HEIGHT*0.2158;
view1Frame.origin.x = 0;
view1Frame.origin.y = 0.1422*SCREEN_HEIGHT-10;
view1.frame = view1Frame;
[self setMaskTo:view1 byRoundingCorners:UIRectCornerBottomRight|UIRectCornerTopRight];
[self.view addSubview:view1];
```

Hier ist die Funktion, die das schwere Heben übernimmt und die ausgewählten Kanten abrundet. Dies ist in unserem Fall die rechte untere und die rechte obere Kante

```
- (void) setMaskTo:(UIView*)view byRoundingCorners:(UIRectCorner) corners
{
    UIBezierPath *rounded = [UIBezierPath bezierPathWithRoundedRect:view.bounds
                                                                    byRoundingCorners:corners
                                                                    cornerRadii:CGSizeMake(20.0, 20.0)];

    CAShapeLayer *shape = [[CAShapeLayer alloc] init];
    [shape setPath:rounded.CGPath];
    view.layer.mask = shape;
}
```

Selektive UIView-Ecken abrunden online lesen: <https://riptutorial.com/de/ios/topic/7224/selektive-UIView-ecken-abrunden>

Kapitel 140: Sicherheit

Einführung

Die Sicherheit von iOS bezieht sich auf Datensicherheit, Transportsicherheit, Codesicherheit usw

Examples

Transportsicherheit mit SSL

iOS-Apps müssen so geschrieben werden, dass die Daten, die über das Netzwerk transportiert werden, geschützt werden.

SSL ist die übliche Methode, dies zu tun.

Immer, wenn die App versucht, Web-Services aufzurufen, um Daten auf Server zu laden oder zu übertragen, sollte **sie SSL über HTTP, dh HTTPS, verwenden** .

Dazu muss die **App** `https://server.com/part` solcher Webdienste und nicht `http://server.com/part` **aufrufen** .

In diesem Fall muss die App dem Server `server.com` mithilfe des SSL-Zertifikats vertrauen.

Hier ist das Beispiel für die Überprüfung der Server-Vertrauenswürdigkeit.

Implementieren Sie `URLSessionDelegate` als:

```
func urlSession(_ session: URLSession, didReceive challenge: URLAuthenticationChallenge,
completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    if challenge.protectionSpace.authenticationMethod == NSURLAuthenticationMethodServerTrust
    {
        let serverTrust:SecTrust = challenge.protectionSpace.serverTrust!

        func acceptServerTrust() {
            let credential:URLCredential = URLCredential(trust: serverTrust)
            challenge.sender?.use(credential, for: challenge)
            completionHandler(.useCredential, URLCredential(trust:
challenge.protectionSpace.serverTrust!))
        }

        let success = SSLTrustManager.shouldTrustServerTrust(serverTrust, forCert:
"Server_Public_SSL_Cert")
        if success {
            acceptServerTrust()
            return
        }
    }
    else if challenge.protectionSpace.authenticationMethod ==
NSURLAuthenticationMethodClientCertificate {
        completionHandler(.rejectProtectionSpace, nil);
        return
    }
    completionHandler(.cancelAuthenticationChallenge, nil)
}
```

```
}
```

Hier ist der Vertrauensmanager: (Swift-Code konnte nicht gefunden werden)

```
@implementation SSLTrustManager
+ (BOOL)shouldTrustServerTrust:(SecTrustRef)serverTrust forCert:(NSString*)certName {
    // Load up the bundled certificate.
    NSString *certPath = [[NSBundle mainBundle] pathForResource:certName ofType:@"der"];
    NSData *certData = [[NSData alloc] initWithContentsOfFile:certPath];
    CFDataRef certDataRef = (__bridge_retained CFDataRef)certData;
    SecCertificateRef cert = SecCertificateCreateWithData(NULL, certDataRef);

    // Establish a chain of trust anchored on our bundled certificate.
    CFArrayRef certArrayRef = CFArrayCreate(NULL, (void *)&cert, 1, NULL);
    SecTrustSetAnchorCertificates(serverTrust, certArrayRef);

    // Verify that trust.
    SecTrustResultType trustResult;
    SecTrustEvaluate(serverTrust, &trustResult);

    // Clean up.
    CFRelease(certArrayRef);
    CFRelease(cert);
    CFRelease(certDataRef);

    // Did our custom trust chain evaluate successfully?
    return trustResult == kSecTrustResultUnspecified;
}
@end
```

Server_Public_SSL_Cert.der ist der öffentliche SSL-Schlüssel der Server.

Mit diesem Ansatz kann unsere App sicherstellen, dass sie mit dem beabsichtigten Server kommuniziert und dass niemand die App-Server-Kommunikation abfängt.

Sichern von Daten in iTunes-Backups

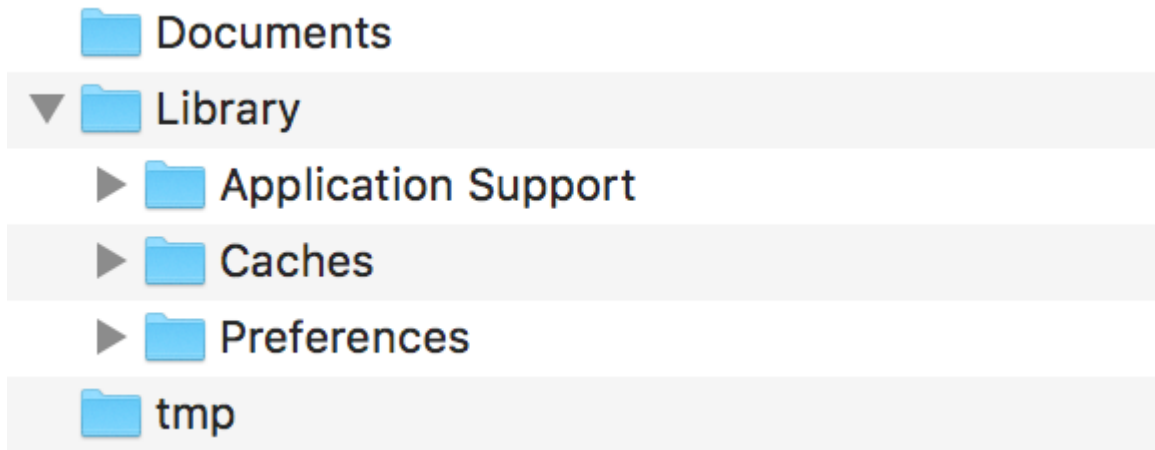
Wenn unsere App-Daten vor iTunes-Sicherungen geschützt werden sollen, müssen wir die App-Daten von der Sicherung in iTunes auslassen.

Wenn ein iOS-Gerät mit iTunes auf macOS gesichert wird, werden alle von allen Apps gespeicherten Daten in dieses Backup kopiert und auf dem Sicherungscomputer gespeichert.

Wir können jedoch unsere App-Daten mit Hilfe `URLResourceKey.isExcludedFromBackupKey` Schlüssels `URLResourceKey.isExcludedFromBackupKey` von dieser Sicherung

`URLResourceKey.isExcludedFromBackupKey` .

Hier ist die Verzeichnisstruktur unserer App:



Hinweis: Im Allgemeinen werden sensible Daten im Verzeichnis 'Application Support' gespeichert.

Wenn Sie beispielsweise alle im **Application Support**- Verzeichnis gespeicherten Daten ausschließen möchten, können Sie den oben genannten Schlüssel wie folgt verwenden:

```
let urls = FileManager.default.urls(for: .applicationSupportDirectory, in:
.userDomainMask)
let baseURL = urls[urls.count-1];

let bundleIdentifier = Bundle.main.object(forKey: "CFBundleIdentifier") as!
String
let pathURL = baseURL.appendingPathComponent(bundleIdentifier)
let persistentStoreDirectoryPath = pathURL.path
if !FileManager.default.fileExists(atPath: persistentStoreDirectoryPath) {
    do {
        try FileManager.default.createDirectory(atPath: path, withIntermediateDirectories:
true, attributes: nil)
    }catch {
        //handle error
    }
}
let dirURL = URL.init(fileURLWithPath: persistentStoreDirectoryPath, isDirectory: true)
do {
    try (dirURL as NSURL).setResourceValue((true), forKey: .isExcludedFromBackupKey)
} catch {
    //handle error
}
```

Es gibt zahlreiche Tools zum Anzeigen von iTunes-Backups für alle gesicherten Daten, um zu überprüfen, ob der oben beschriebene Ansatz funktioniert oder nicht.

[iExplorer](#) eignet sich gut zum [Durchsuchen von iTunes-Backups](#).

Sicherheit online lesen: <https://riptutorial.com/de/ios/topic/9999/sicherheit>

Kapitel 141: Simulator

Einführung

iOS-, watchOS- und tvOS-Simulatoren sind eine hervorragende Möglichkeit, Ihre Apps ohne Verwendung eines Geräts zu testen. Hier werden wir über die Arbeit mit Simulatoren sprechen.

Bemerkungen

Verschiedene Arten von Simulatoren

- iOS Simulator
- watchOS Simulator
- tvOS Simulator
- Touch Bar Simulator

Es gibt keinen Simulator für macOS, da Xcode unter macOS ausgeführt wird und die Apps nativ ausgeführt werden.

Hilfe bekommen

Sie können die Simulator-Hilfe immer unter Hilfe -> Simulator-Hilfe besuchen:



Xcode

File

Edit

View

Find

Navigate



- ▶ Swift
- ▶ Objective-C
- ▶ JavaScript

Abc

Impo
chang

Simulat
Simulat
of the s

Simulat
simulat
These s

Kapitel 142: Simulator-Builds

Einführung

Wo finde ich Simulator Build?

Gehen Sie zu ~ / Library / Developer / CoreSimulator / Devices /

Sie finden Verzeichnisse mit alphanumerischen Namen

Klicken Sie dann auf eines der Verzeichnisse und nehmen Sie folgende Auswahl vor

Daten / Container / Bündel / Anwendung /

Wiederum werden Sie Verzeichnisse mit alphanumerischen Namen finden, wenn Sie darauf klicken

Simulator dort bauen

Hinweis:

Die Installation des auf dem Simulator erstellten iOS-Geräts wird nicht funktionieren.

iPhone-Simulator verwendet i386-Architektur. iPad-Simulator-Builds verwenden x8

Examples

Build manuell auf dem Simulator installieren

```
xcrun simctl install booted *.app
```

Simulator-Builds online lesen: <https://riptutorial.com/de/ios/topic/9813/simulator-builds>

Kapitel 143: SiriKit

Bemerkungen

Verschiedene Arten von Siri-Anfragen

- Ride Booking (z. B. bringen Sie mich über MyApp nach New York)
- Messaging (z. B. Senden eines Textes an John mit MyApp)
- Fotosuche (z. B. Suche nach Strandfotos, die im letzten Sommer in MyApp aufgenommen wurden)
- Zahlungen (z. B. Schicken Sie mit MyApp letzte Nacht \$ 20 für John zum Abendessen)
- VoIP-Calling (zB Mike auf meiner MyApp anrufen)
- Workouts (zB mein tägliches Lauftraining über MyApp starten)
- Klima und Radio (speziell für CarPlay entwickelt, z. B. Heizung auf 72 Grad einstellen)

Examples

Hinzufügen der Siri-Erweiterung zur App

Um Siri-Funktionen in Ihre App zu integrieren, sollten Sie eine Erweiterung hinzufügen, wie Sie dies beim Erstellen eines iOS 10-Widget (alte Today View Extension) oder einer benutzerdefinierten Tastatur tun würden.

Fähigkeit hinzufügen

1- Wählen Sie in den Projekteinstellungen Ihr iOS-App-Ziel aus, und wechseln Sie zur Registerkarte Funktionen

2- Aktivieren Sie die Siri-Funktion

Erweiterung hinzufügen

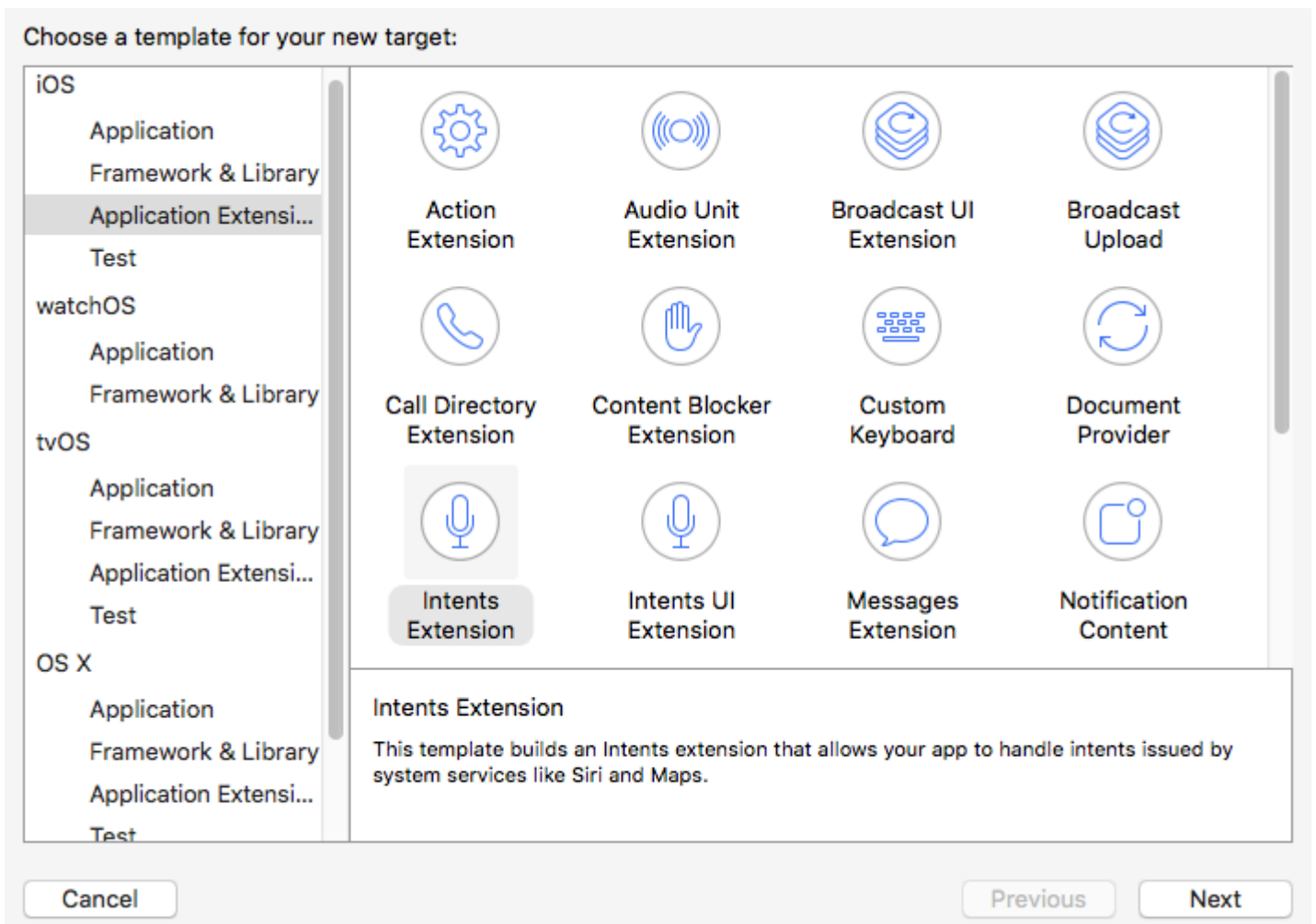
1- Gehe zu Datei -> Neu -> Ziel ...

2- Wählen Sie im linken Fensterbereich iOS -> Application Extension aus

3- Doppelklicken Sie von rechts auf Intents Extension

Laut Apple:

Intents-Erweiterungsvorlage erstellt eine Intents-Erweiterung, mit der Ihre App die von Systemdiensten wie Siri und Maps ausgegebenen Absichten verarbeiten kann.



4- Wählen Sie einen Namen und vergewissern Sie sich, dass "UI-Erweiterung einbeziehen" aktiviert ist.

Language:

Include UI Extension

Durch diese Schritte werden zwei neue Ziele (Intents Extension und UI Extension) erstellt, die standardmäßig den Code für die Trainingsabsicht enthalten. Informationen zu verschiedenen Arten von Siri-Anforderungen finden Sie unter Anmerkungen.

Hinweis

Wann immer Sie Ihre Erweiterung debuggen möchten, wählen Sie einfach das Intent-Schema aus den verfügbaren Schemata aus.

Hinweis

Sie können SiriKit-Apps nicht im Simulator testen. Stattdessen brauchen Sie ein echtes Gerät.

SiriKit online lesen: <https://riptutorial.com/de/ios/topic/5869/sirikit>

Kapitel 144: SLComposeViewController

Examples

SLComposeViewController für Twitter, Facebook, SinaWeibo und TencentWeibo

Ziel c

Fügen Sie zunächst das `Social Framework` zum XCode-Projekt hinzu.

Importieren Sie die `#import "Social/Social.h"` `Import #import "Social/Social.h"` in den erforderlichen ViewController

Twitter mit Text, Bild und Link

```
//- - To Share text on twitter - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
{
    //Tweet
    SLComposeViewController *twitterVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
    //To send link together with text
    [twitterVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
    //To add a photo to a link
    [twitterVC addImage:[UIImage imageNamed:@"image"]];
    //Sending link and Image with the tweet
    [twitterVC setInitialText:text];
    /* While adding link and images in a tweet the effective length of a tweet i.e.
the number of characters which can be entered by the user decreases.
The default maximum length of a tweet is 140 characters*/
    [self presentViewController:twitterVC animated:YES completion:nil];
}
else
{
    //Shows alert if twitter is not signed in
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
    [alertCont addAction:okay];
}
}
```

Facebook mit Text, Bild und Link

```
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeFacebook])
{
    SLComposeViewController *fbVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeFacebook];
    [fbVC setInitialText:text];
    //To send link together with text
    [fbVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
}
```

```

//To add a photo to a link
[fbVC addImage:[UIImage imageNamed:@"image"]];
[self presentViewController:fbVC animated:YES completion:nil];
}
else
{//Shows alert if twitter is not signed in
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
    [alertCont addAction:okay];
}
}

```

SinaWeibo

```

// - - SinaWeibo - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeSinaWeibo]){

    SLComposeViewController *SinaWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeSinaWeibo];
    [SinaWeiboVC setInitialText:text];

    [self presentViewController:SinaWeiboVC animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
}

```

TencentWeibo

```

// - -TencentWeibo text share
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTencentWeibo])
{
    SLComposeViewController *tencentWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTencentWeibo];
    [tencentWeibo setInitialText:text];
    [self presentViewController:tencentWeibo animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
}

```

SLComposeViewController online lesen:

<https://riptutorial.com/de/ios/topic/7366/slcomposeviewController>

Kapitel 145: Standort mit GPX-Dateien simulieren iOS

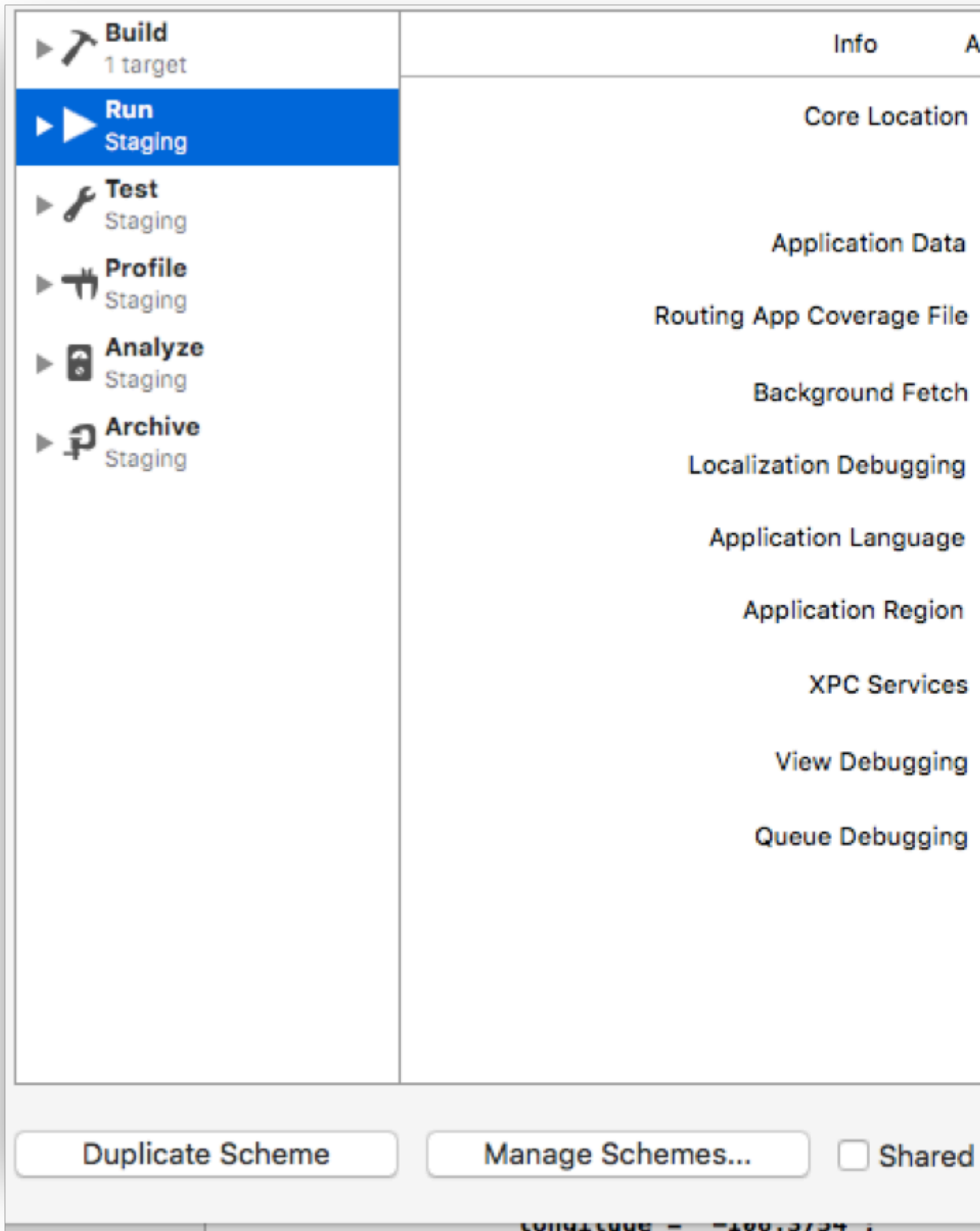
Examples

Ihre .gpx-Datei: MPS_HQ.gpx

```
<gpx xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:gpxx = "http://www.garmin.com/xmlschemas/GpxExtensions/v3"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd
http://www.garmin.com/xmlschemas/GpxExtensions/v3
http://www8.garmin.com/xmlschemas/GpxExtensions/v3/GpxExtensionsv3.xsd"
  version="1.1"
  creator="gpx-poi.com">
<wpt lat="38.9072" lon="77.0369">38.9072/-77.0369
<time>2015-04-16T22:20:29Z</time>
  <name>Washington, DC</name>
  <extensions>
    <gpxx:WaypointExtension>
      <gpxx:Proximity>10</gpxx:Proximity>
      <gpxx:Address>
        <gpxx:StreetAddress>Washington DC</gpxx:StreetAddress>
        <gpxx:City>Washington</gpxx:City>
        <gpxx:State>DC</gpxx:State>
        <gpxx:Country>United States</gpxx:Country>
        <gpxx:PostalCode> 20005 </gpxx:PostalCode>
      </gpxx:Address>
    </gpxx:WaypointExtension>
  </extensions>
```

Um diesen Ort festzulegen:

1. Gehen Sie zum Bearbeitungsschema.
2. Wählen Sie Ausführen -> Optionen.
3. Aktivieren Sie "Standortsimulation zulassen".
4. Wählen Sie den * .GPX-Dateinamen aus der Dropdown-Liste "Standardspeicherort" aus.



Standort mit GPX-Dateien simulieren iOS online lesen:

<https://riptutorial.com/de/ios/topic/9883/standort-mit-gpx-dateien-simulieren-ios>

Kapitel 146: StoreKit

Examples

Erhalten Sie lokalisierte Produktinformationen aus dem App Store

Mit `SKProductsRequest` Sie lokalisierte Produktinformationen aus einer Reihe von Produktbezeichner-Strings `SKProductsRequest` :

```
import StoreKit

let productIdentifierSet = Set(["yellowSubmarine", "pennyLane"])
let productsRequest = SKProductsRequest(productIdentifiers: productIdentifierSet)
```

Um die Produkte von `productsRequest` bearbeiten zu können, müssen wir der Anfrage, die die Antwort bearbeitet, einen Delegierten zuweisen. Der Delegat muss dem `SKProductsRequestDelegate` Protokoll entsprechen. Das bedeutet, dass er von `NSObject` (dh jedem `Foundation` Objekt) erben und die `productsRequest` Methode implementieren muss:

```
class PaymentManager: NSObject, SKProductsRequestDelegate {

    var products: [SKProduct] = []

    func productsRequest(request: SKProductsRequest,
                        didReceiveResponse response: SKProductsResponse) {

        products = response.products

    }

}
```

Um die `productsRequest` zu initiieren `productsRequest` weisen wir `PaymentManager` als `PaymentManager` der `PaymentManager` `-request` zu und rufen die `start()` -Methode für die Anfrage auf:

```
let paymentManager = PaymentManager()
productsRequest.delegate = paymentManager
productsRequest.start()
```

Wenn die Anforderungen erfolgreich sind, werden die Produkte in `paymentManager.products` .

StoreKit online lesen: <https://riptutorial.com/de/ios/topic/6025/storekit>

Kapitel 147: Storyboard

Einführung

Normalerweise werden Ansichts-Controller in einem Storyboard instanziiert und automatisch als Reaktion auf Aktionen erstellt, die im Storyboard selbst definiert sind. Sie können jedoch ein Storyboard-Objekt verwenden, um den anfänglichen View-Controller in einer Storyboard-Datei oder andere View-Controller zu instanziiieren, die Sie programmgesteuert darstellen möchten. Im Folgenden finden Sie Beispiele für beide Anwendungsfälle.

Examples

Initialisieren

```
//Swift
let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

//Objective-c
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"Main" bundle:[NSBundle mainBundle]];
```

Initial ViewController abrufen

```
//Swift
let mainScreen = storyboard.instantiateInitialViewController()

//Objective-c
UIViewController *initialScreen = [storyboard instantiateInitialViewController];
```

ViewController abrufen

```
//Swift
let viewController = storyboard.instantiateViewControllerWithIdentifier("identifier")

//Objective-c
UIViewController *viewController = [storyboard
instantiateViewControllerWithIdentifier:@"identifier"];
```

Storyboard online lesen: <https://riptutorial.com/de/ios/topic/3514/storyboard>

Kapitel 148: Swift: Ändern des rootViewControllers in AppDelegate, um den Haupt- oder Login- / Onboarding-Ablauf anzuzeigen

Einführung

Es ist oft hilfreich, neuen Benutzern Ihrer App eine erste Erfahrung zu bieten. Dies kann verschiedene Ursachen haben, z. B. die Aufforderung zur Anmeldung (falls dies für Ihre Situation erforderlich ist), die Verwendung der App erklären oder einfach über neue Funktionen in einem Update informieren (z iOS11).

Bemerkungen

Erstens können Sie Storyboards effektiv verwenden, wenn Sie mit mehreren Flows umgehen. Standardmäßig verwendet Ihre Anwendung `Main.storyboard` für Ihren primären Ablauf. Ihr Onboarding- / Alternativfluss kann in einem sekundären Storyboard enthalten sein, z. `Onboarding.storyboard`

Das hat eine Reihe von Vorteilen:

- In einem Team von Entwicklern kann die Arbeit für jeden Benutzerfluss getrennt werden
- klarere Quellcodeverwaltung (git)
- Trennung von Bedenken

Wenn Ihre App gestartet wird, können Sie festlegen, welcher Flow angezeigt werden soll. Die Logik dazu kann in Ihrem AppDelegate enthalten sein:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let isFirstRun = true // logic to determine goes here
    if isFirstRun {
        showOnboarding()
    }
    return true
}
```

Um den Onboarding-Fluss zu zeigen, ist es sinnvoll, darüber nachzudenken, wie Sie mit der Entlassung umgehen möchten, wenn die Person, die sie verwendet, die Reise abgeschlossen hat, und welche semantisch korrekt für das ist, was Sie erstellen möchten.

Ansätze:

Die zwei Hauptansätze sind:

1. Tauschen Sie den Root-View-Controller des Hauptfensters der App aus
2. Präsentieren Sie den Onboarding-Flow als modale Reise, wobei Sie den Main-Flow überlappen.

Die Implementierung davon sollte in einer Erweiterung von AppDelegate enthalten sein.

Examples

Option 1: Den Root-View-Controller austauschen (gut)

Das Wechseln des Root-View-Controllers bietet Vorteile, auch wenn die Übergangsoptionen auf die von `UIViewAnimationOptions` unterstützten Optionen beschränkt sind. Abhängig davon, wie Sie den Übergang zwischen Flows `UIViewAnimationOptions` möchten, müssen Sie möglicherweise einen benutzerdefinierten Übergang implementieren, was mühsam sein kann.

Sie können den Onboarding-Fluss anzeigen, indem Sie einfach den

```
UIApplication.shared.keyWindow.rootViewController
```

Die Entlassung erfolgt durch Verwendung von `UIView.transition(with:)` und Übergeben des Übergangsstils als `UIViewAnimationOptions`, in diesem Fall `Cross Dissolve`. (Flips und Curls werden ebenfalls unterstützt).

Sie müssen auch den Rahmen der Hauptansicht festlegen, bevor Sie wieder in diese Ansicht wechseln, da Sie sie zum ersten Mal instanziiieren.

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = UIApplication.shared.keyWindow, let onboardingViewController =
        UIStoryboard(name: "Onboarding", bundle: nil).instantiateInitialViewController() as?
        OnboardingViewController {
            onboardingViewController.delegate = self
            window.rootViewController = onboardingViewController
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow, let mainViewController =
        UIStoryboard(name: "Main", bundle: nil).instantiateInitialViewController() {
            mainViewController.view.frame = window.bounds
            UIView.transition(with: window, duration: 0.5, options: .transitionCrossDissolve,
            animations: {
                window.rootViewController = mainViewController
            }, completion: nil)
        }
    }
}
```

Option 2: Alternativen Fluss modal (besser) präsentieren

Bei der einfachsten Implementierung kann der Onboarding-Fluss einfach in einem modalen Kontext dargestellt werden, da sich der Benutzer semantisch auf einer einzigen Reise befindet.

[Apple Human Interface Guidelines - Modalität] [1]:

Erwägen Sie die Erstellung eines modalen Kontextes nur dann, wenn es entscheidend ist, die Aufmerksamkeit einer Person zu erlangen, wenn eine Aufgabe abgeschlossen oder aufgegeben werden muss, um die App weiterzuverwenden oder wichtige Daten zu speichern.

Die modale Darstellung erlaubt die einfache Möglichkeit der Kündigung am Ende der Reise, wobei nur wenig der Controller ausgetauscht werden muss.

Benutzerdefinierte Übergänge werden ebenfalls standardmäßig unterstützt, da hierfür die `ViewController.present()` API verwendet wird:

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = window, let onboardingViewController = UIStoryboard(name:
"Onboarding", bundle: nil).instantiateInitialViewController() as? OnboardingViewController {
            onboardingViewController.delegate = self
            window.makeKeyAndVisible()
            window.rootViewController?.present(onboardingViewController, animated: false,
completion: nil)
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow {
            window.rootViewController?.dismiss(animated: true, completion: nil)
        }
    }
}
```

Swift: Ändern des `rootViewControllers` in `AppDelegate`, um den Haupt- oder Login- / Onboarding-Ablauf anzuzeigen online lesen: <https://riptutorial.com/de/ios/topic/10880/swift--andern-des-rootviewcontrollers-in-appdelegate--um-den-haupt--oder-login---onboarding-ablauf-anzuzeigen>

Kapitel 149: SWRevealViewController

Bemerkungen

Die Verwendung der SWRevealViewController-Klasse als Hauptnavigation führt möglicherweise nicht immer zu einer optimalen Benutzererfahrung. Wenn die Seitenleiste nur 5 oder weniger Einträge enthält (oder der Inhalt in 5 oder weniger Einträge komprimiert werden kann), sollten Sie die Standard-Registerkartenleiste in Betracht ziehen.

Die Registerkartenleiste ist intuitiv und ermöglicht dem Benutzer einen schnellen Wechsel zwischen Ansichten und Kontexten. Auf der anderen Seite kann die Sidebar-Navigation mehr Aktionen ausführen als das Umschalten der Ansicht / des Kontextes und verbraucht weniger Platz.

Weitere Informationen finden Sie in den [iOS-Richtlinien zur](#) Benutzeroberfläche von Apple.

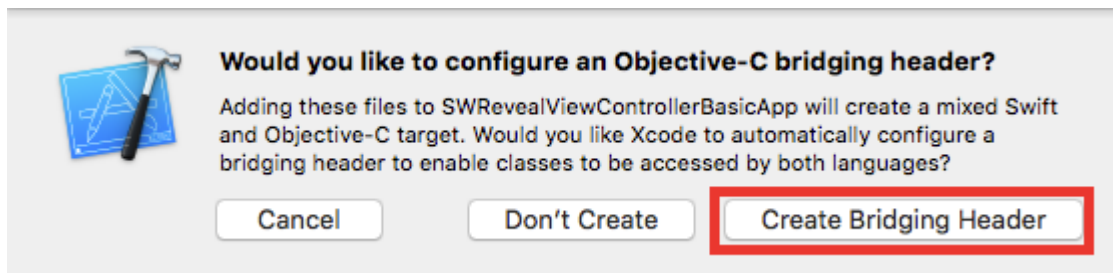
Examples

Einrichten einer Basis-App mit SWRevealViewController

Erstellen Sie eine Basisanwendung mit Einzelansichtsanwendungsvorlage mit Swift als Sprache

Fügen Sie `SWRevealViewController.h` und `SWRevealViewController.m`

Klicken Sie dann auf die Schaltfläche Bridging-Header erstellen

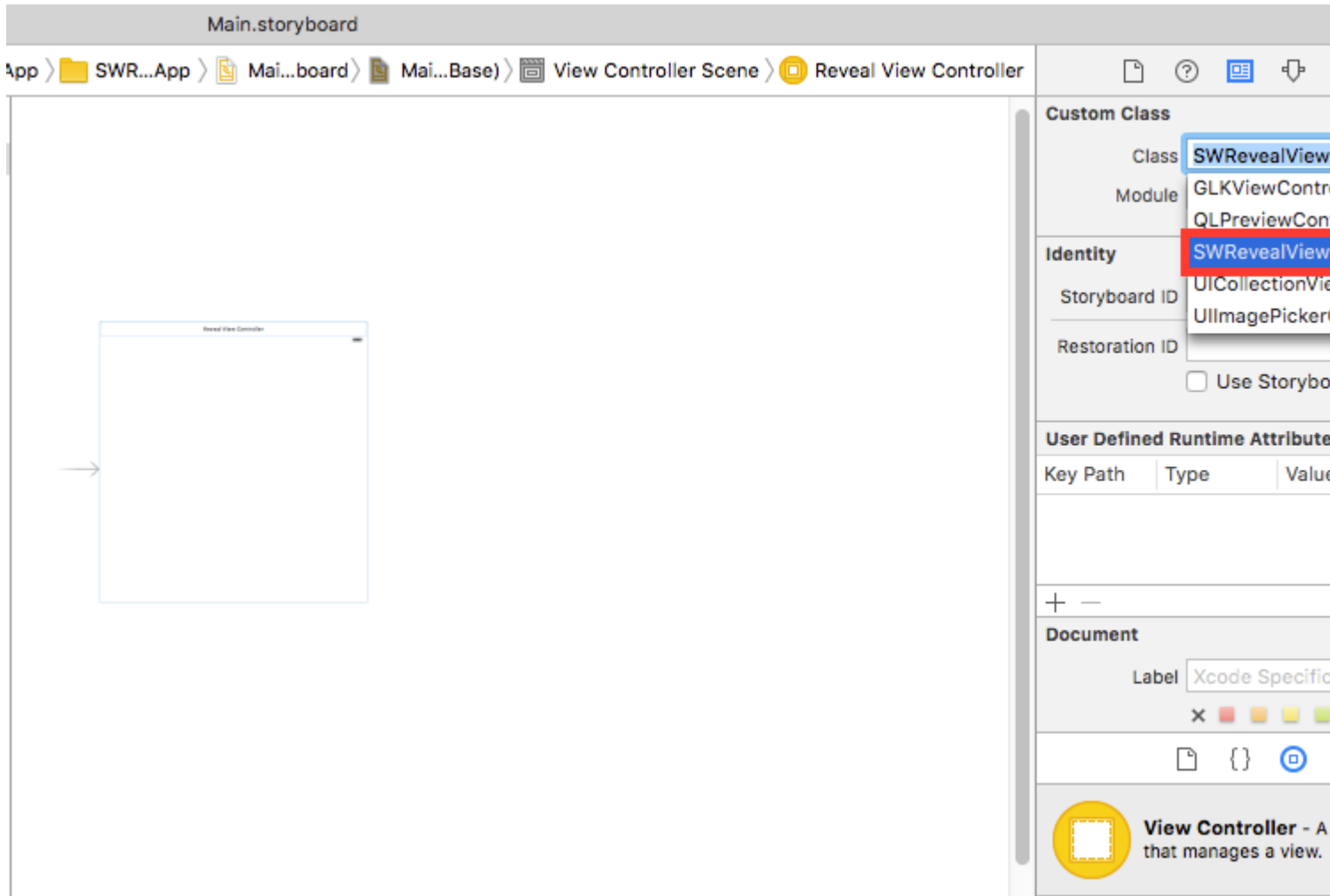


und fügen Sie hinzu

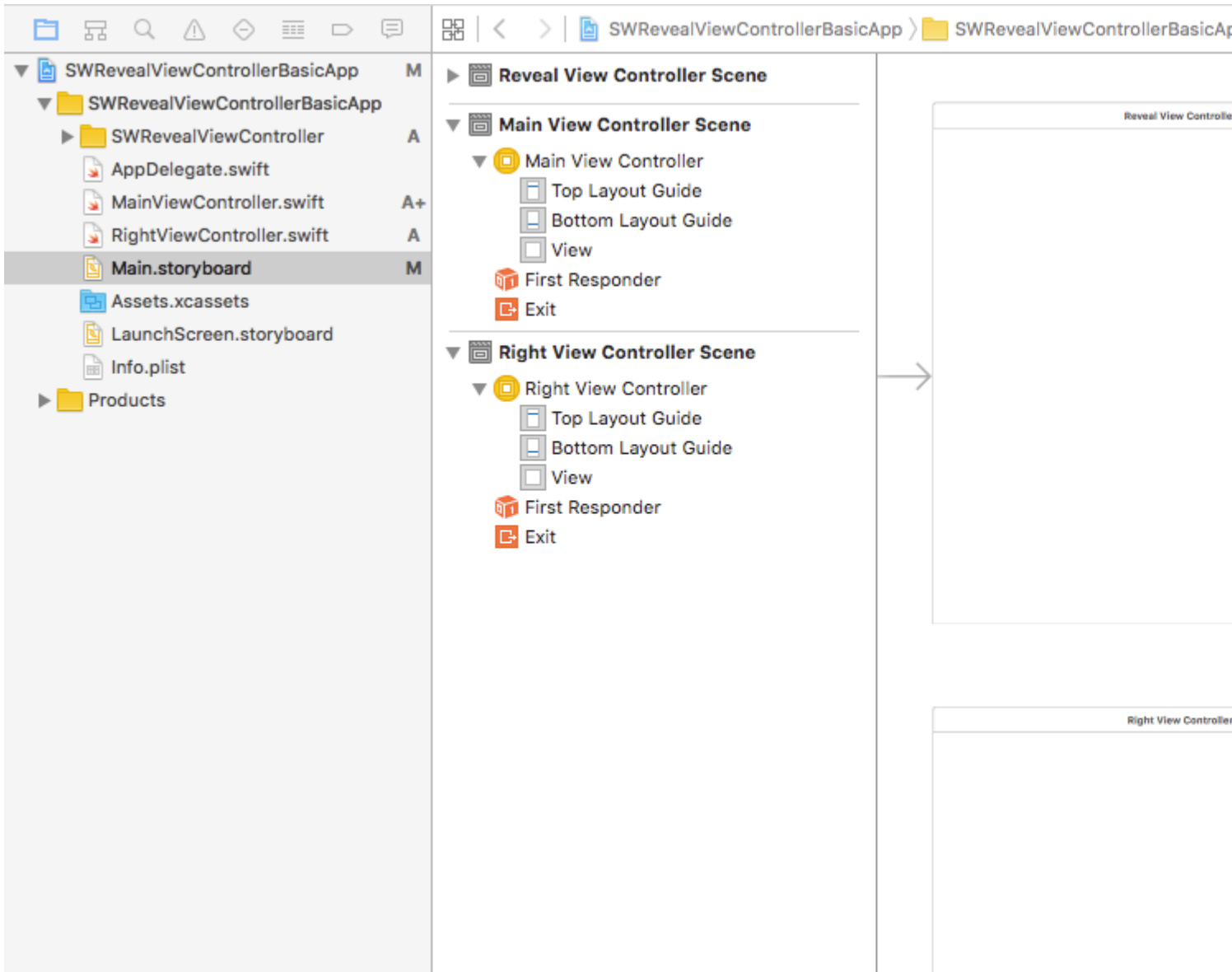
```
#import "SWRevealViewController.h"
```

auf dem Bridging-Header

Wählen Sie dann viewController auf dem Storyboard und ändern Sie die Klasse in `SWRevealViewController`

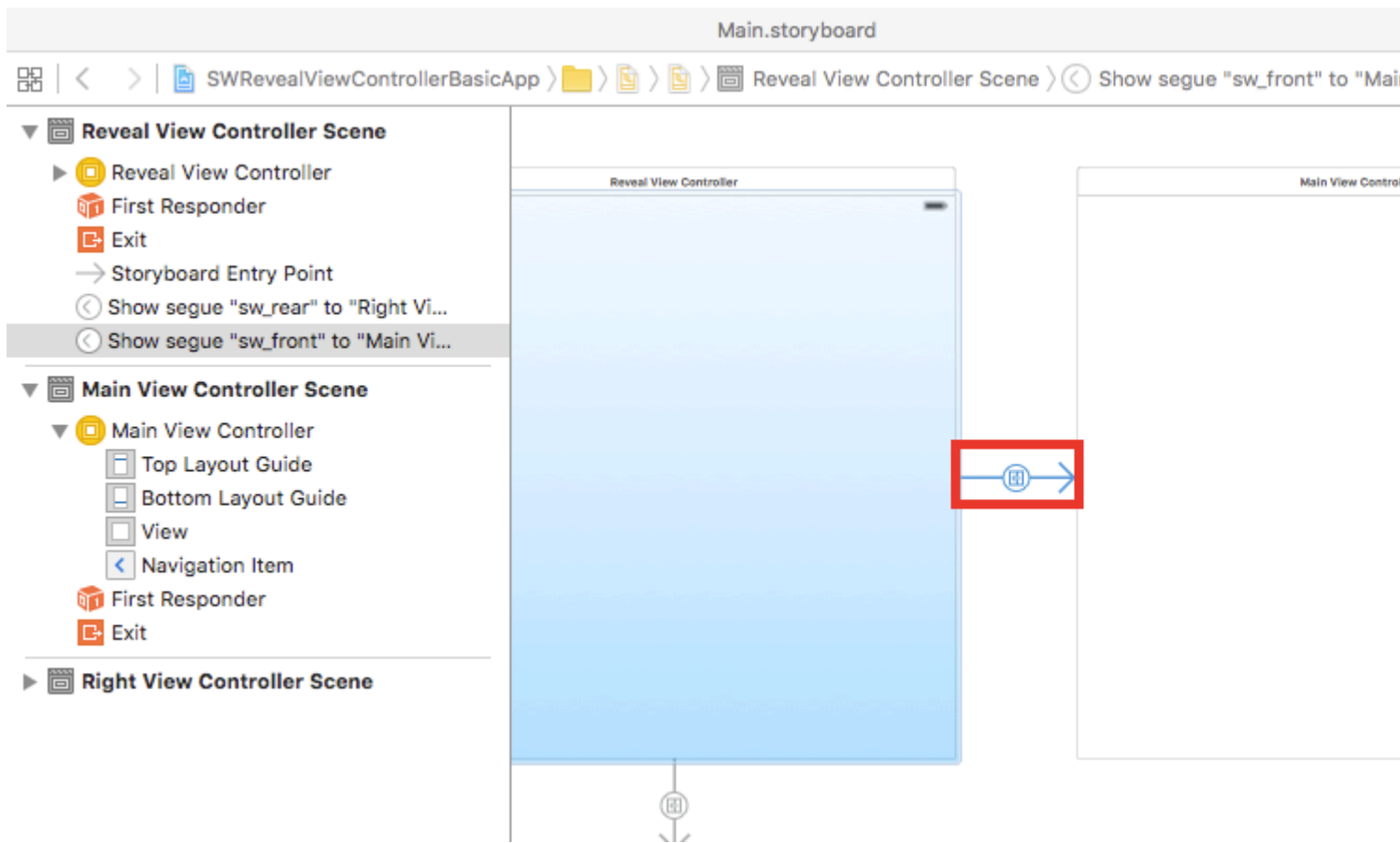


Benennen Sie dann den viewController für Dateien in MainViewController um und fügen Sie einen neuen ViewController mit dem Namen RightViewController hinzu



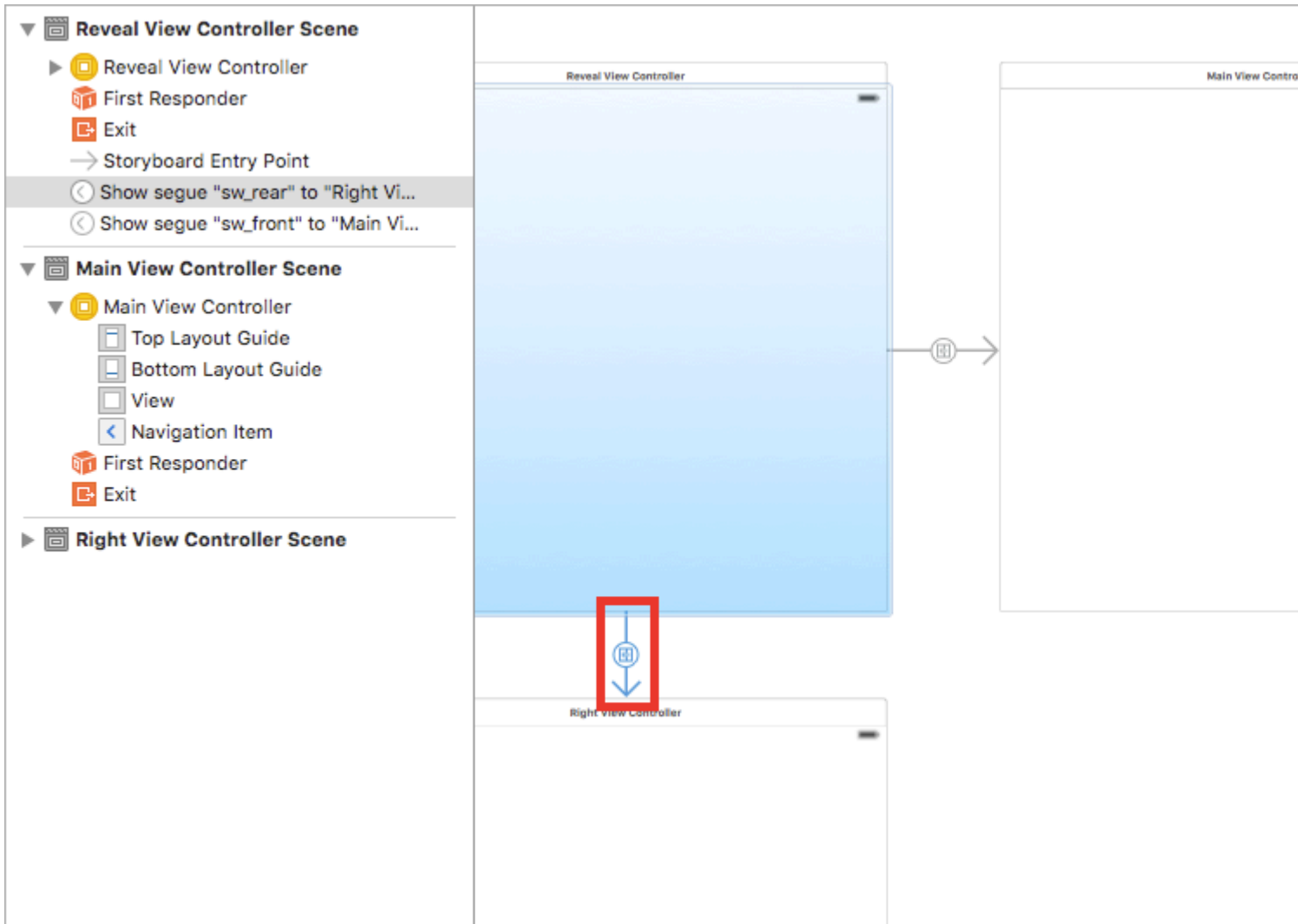
Dann fügen wir zwei Segmente von SWRevealViewController zu MainViewController und von SWRevealViewController zu RightViewController hinzu. Dann müssen wir die erste (von SWRevealViewController bis MainViewController) auswählen und die Eigenschaften bearbeiten

bei Bezeichner setze `sw_front` auf Class set `SWRevealViewControllerSegueSetController`



danach müssen wir dasselbe mit dem segue tun (von SWRevealViewController zu RightViewController)

bei Bezeichner set `sw_rear` bei Klassensatz `SWRevealViewControllerSegueSetController`



Fügen Sie dann am MainViewController diese Zeile zur `viewDidLoad` Methode hinzu

```
self.view.addGestureRecognizer(self.revealViewController().panGestureRecognizer());
```

Und das ist alles, Sie haben eine Basis-App mit integriertem `SWRevealViewController`. Sie können nach rechts `RightViewController`, um `RightViewController` als seitliches Menü `RightViewController`

SWRevealViewController online lesen:

<https://riptutorial.com/de/ios/topic/4614/swrevealviewController>

Kapitel 150: Tastatur verwalten

Examples

Scrollen einer UIScrollView / UITableView bei Anzeige der Tastatur

Dort gibt es nur wenige Ansätze:

1. Sie können Benachrichtigungen zu Tastaturerscheinungsereignissen abonnieren und den Versatz manuell ändern:

```
//Swift 2.0+
override func viewDidLoad() {
    super.viewDidLoad()

    NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillShow(_:)), name: UIKeyboardWillShowNotification, object:
nil)
    NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
}

func keyboardWillShow(notification: NSNotification) {
    if let userInfo = notification.userInfo {
        if let keyboardHeight =
userInfo[UIKeyboardFrameEndUserInfoKey]?.CGRectValue.size.height {
            tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardHeight, 0)
        }
    }
}

func keyboardWillHide(notification: NSNotification) {
    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0)
}

//Objective-C
- (void)viewDidLoad {

    [super viewDidLoad];

    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];
}

- (void)keyboardWillShow:(NSNotification *)notification {

    NSDictionary *userInfo = [notification userInfo];

    if (userInfo) {

        CGRect keyboardEndFrame;
```

```

        [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardEndFrame];
        tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardEndFrame.size.height, 0);
    }
}
- (void)keyboardWillHide:(NSNotification *)notification {
    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0);
}

```

2. Oder nutzen Sie fertige Lösungen wie TPKeyboardAvoidingTableView oder TPKeyboardAvoidingScrollView <https://github.com/michaeltyson/TPKeyboardAvoiding>

Schließen Sie eine Tastatur mit Tipp auf Anzeige ab

Wenn Sie eine Tastatur durch Antippen außerhalb der Tastatur ausblenden möchten, können Sie diesen Hacky-Trick verwenden (funktioniert nur mit Objective-C):

```

- (void)viewDidLoad {
    [super viewDidLoad];

    // dismiss keyboard when tap outside a text field
    UITapGestureRecognizer *tapGestureRecognizer = [[UITapGestureRecognizer alloc]
initWithTarget:self.view action:@selector(endEditing:)];
    [tapGestureRecognizer setCancelsTouchesInView:NO];
    [self.view addGestureRecognizer:tapGestureRecognizer];
}

```

Für Swift wird es etwas mehr Code geben:

```

override func viewDidLoad() {
    super.viewDidLoad()

    // dismiss keyboard when tap outside a text field
    let tapGestureRecognizer: UITapGestureRecognizer = UITapGestureRecognizer(target: self,
action: #selector(YourVCName.dismissKeyboard))
    view.addGestureRecognizer(tapGestureRecognizer)
}

//Calls this function when the tap is recognized.
func dismissKeyboard() {
    //Causes the view (or one of its embedded text fields) to resign the first responder
status.
    view.endEditing(true)
}

```

Ein weiteres Swift 3 / iOS 10-Beispiel

```

class vc: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }
}

```

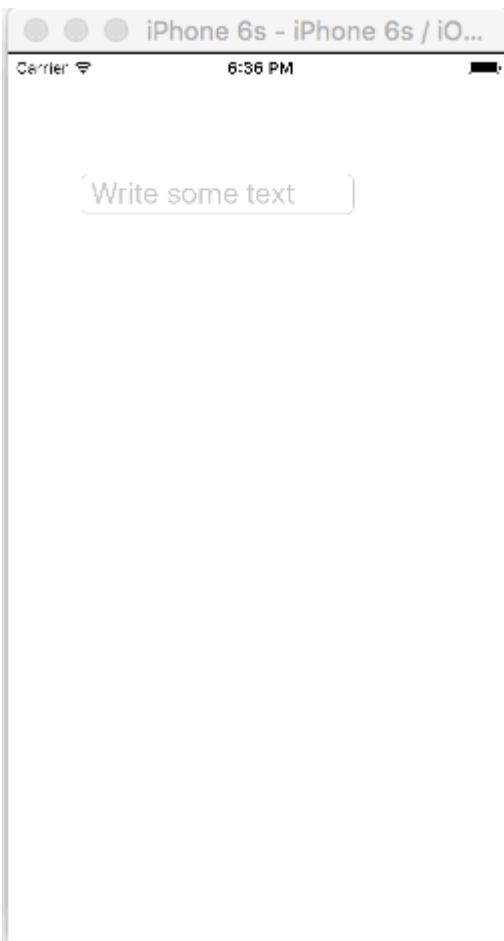
```

        txtSomeField.delegate = self
    }
}

extension vc: UITextFieldDelegate {
    //Hide the keyboard for any text field when the UI is touched outside of the keyboard.
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
    {
        self.view.endEditing(true) //Hide the keyboard
    }
}
}

```

Erstellen Sie eine benutzerdefinierte In-App-Tastatur



Dies ist eine grundlegende In-App-Tastatur. Die gleiche Methode kann verwendet werden, um nahezu jedes Tastaturlayout zu erstellen. Hier sind die wichtigsten Dinge, die getan werden müssen:

- Erstellen Sie das Tastaturlayout in einer .xib-Datei, deren Besitzer eine Swift- oder Objective-C-Klasse ist, bei der es sich um eine `UIView` Unterklasse handelt.
- `UITextField` Sie dem `UITextField` mit, die benutzerdefinierte Tastatur zu verwenden.
- Verwenden Sie einen Delegaten, um zwischen der Tastatur und dem Hauptansicht-Controller zu kommunizieren.

Erstellen Sie die .xib-Tastaturlayoutdatei

- In Xcode gehen Sie zu **Datei > Neu > Datei ... > iOS > Benutzeroberfläche > Ansicht** , um die .xib-Datei zu erstellen.
- Ich habe meine Keyboard.xib angerufen
- Fügen Sie die Schaltflächen hinzu, die Sie benötigen.
- Verwenden Sie Auto-Layout-Einschränkungen, damit die Größe der Tasten unabhängig von der Größe der Tastatur entsprechend angepasst wird.
- Legen Sie den Eigentümer der Datei (nicht die Stammansicht) als `Keyboard` Klasse fest. Dies ist eine häufige Fehlerquelle. Sie erstellen diese Klasse im nächsten Schritt. Siehe die Notiz am Ende.

Erstellen Sie die .swift UIView-Unterklassen-Tastaturdatei

- In Xcode gehen Sie zu **Datei > Neu > Datei ... > iOS > Quelle > Cocoa Touch-Klasse** , um die Swift- oder Objective-C- **Klasse** zu erstellen. Wählen Sie `UIView` als Superklasse für neu erstellte Klassen
- Ich habe meine `Keyboard.swift` (`Keyboard` Klasse in Objective-C) angerufen.
- Fügen Sie den folgenden Code für Swift hinzu:

```
import UIKit

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
protocol KeyboardDelegate: class {
    func keyWasTapped(character: String)
}

class Keyboard: UIView {

    // This variable will be set as the view controller so that
    // the keyboard can send messages to the view controller.
    weak var delegate: KeyboardDelegate?

    // MARK:- keyboard initialization

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        initializeSubviews()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        initializeSubviews()
    }

    func initializeSubviews() {
```

```

        let xibName = "Keyboard" // xib extension not included
        let view = NSBundle.mainBundle().loadNibNamed(xibName, owner: self,
options: nil)[0] as! UIView
        self.addSubview(view)
        view.frame = self.bounds
    }

    // MARK:- Button actions from .xib file

    @IBAction func keyTapped(sender: UIButton) {
        // When a button is tapped, send that information to the
        // delegate (ie, the view controller)
        self.delegate?.keyWasTapped(sender.titleLabel!.text!) // could alternatively
send a tag value
    }
}

```

- Fügen Sie den folgenden Code für Objective-C hinzu:

Keyboard.h-Datei

```

#import <UIKit/UIKit.h>

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
@protocol KeyboardDelegate<NSObject>
- (void)keyWasTapped:(NSString *)character;
@end

@interface Keyboard : UIView
@property (nonatomic, weak) id<KeyboardDelegate> delegate;
@end

```

Keyboard.m-Datei

```

#import "Keyboard.h"

@implementation Keyboard

- (id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    [self initializeSubviews];
    return self;
}

- (id)initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];
    [self initializeSubviews];
    return self;
}

- (void)initializeSubviews {
    NSString *xibName = @"Keyboard"; // xib extension not included
    UIView *view = [[[NSBundle mainBundle] loadNibNamed:xibName owner:self
options:nil] firstObject];
    [self addSubview:view];
    view.frame = self.bounds;
}

```

```

}

// MARK:- Button actions from .xib file

-(IBAction)keyTapped:(UIButton *)sender {
    // When a button is tapped, send that information to the
    // delegate (ie, the view controller)
    [self.delegate keyWasTapped:sender.titleLabel.text]; // could alternatively send a
tag value
}

@end

```

- Steuern Sie die Drag-Aktionen von den Schaltflächen zum Schaltflächenrückruf in der .xib-Datei auf die @IBAction Methode im Besitzer von Swift oder Objective-C, um sie alle miteinander zu verbinden.
- Beachten Sie, dass der Protokoll- und Delegierungscode. In [dieser Antwort finden Sie](#) eine einfache Erklärung zur Funktionsweise von Delegierten.

Richten Sie den View Controller ein

- Fügen UITextField Ihrem Haupt-Storyboard ein UITextField und verbinden Sie es mit einem IBOutlet mit Ihrem View-Controller. Nennen Sie es textField.
- Verwenden Sie den folgenden Code für den View Controller in Swift:

```

import UIKit

class ViewController: UIViewController, KeyboardDelegate {

    @IBOutlet weak var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

        // initialize custom keyboard
        let keyboardView = Keyboard(frame: CGRect(x: 0, y: 0, width: 0, height: 300))
        keyboardView.delegate = self // the view controller will be notified by the
keyboard whenever a key is tapped

        // replace system keyboard with custom keyboard
        textField.inputView = keyboardView
    }

    // required method for keyboard delegate protocol
    func keyWasTapped(character: String) {
        textField.insertText(character)
    }
}

```

- Verwenden Sie den folgenden Code für Objective-C:

.h Datei

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@end
```

.m Datei

```
#import "ViewController.h"
#import "Keyboard.h"

@interface ViewController ()<KeyboardDelegate>

@property (nonatomic, weak) IBOutlet UITextField *textField;

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // initialize custom keyboard
    Keyboard *keyboardView = [[Keyboard alloc] initWithFrame:CGRectMake(0, 0, 0, 300)];
    keyboardView.delegate = self; // the view controller will be notified by the keyboard
    whenever a key is tapped

    // replace system keyboard with custom keyboard
    self.textField.inputView = keyboardView;
}

- (void)keyWasTapped:(NSString *)character {
    [self.textField insertText:character];
}

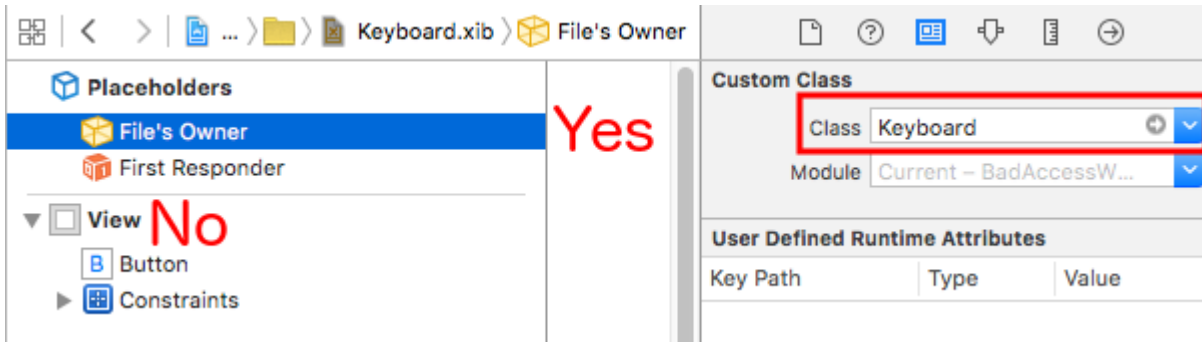
@end
```

- Beachten Sie, dass der View Controller das `KeyboardDelegate` Protokoll verwendet, das wir oben definiert haben.

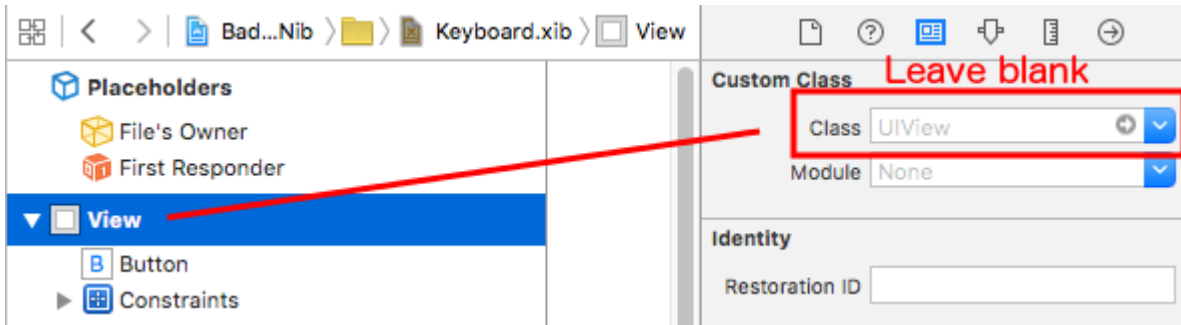
Häufiger Fehler

Wenn Sie einen `EXC_BAD_ACCESS` Fehler erhalten, liegt dies wahrscheinlich daran, dass Sie die benutzerdefinierte Klasse der Ansicht als `Keyboard EXC_BAD_ACCESS`, anstatt dies für den Besitzer der Nib-Datei zu tun.

Wählen Sie `Keyboard.nib` und wählen Sie dann den Eigentümer der Datei.



Stellen Sie sicher, dass die benutzerdefinierte Klasse für die Stammansicht leer ist.



Anmerkungen

Dieses Beispiel stammt ursprünglich aus [dieser Stack Overflow-Antwort](#) .

Verwalten der Tastatur mit einem Singleton + -Delegierten

Als ich anfing, die Tastatur zu verwalten, verwendete ich in jedem ViewController separate Benachrichtigungen.

Benachrichtigungsmethode (mit NSNotification):

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(ViewController.keyboardNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    func keyboardNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)
```

```

    if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
        lowerViewBottomConstraint.constant = 0
    } else {
        lowerViewBottomConstraint.constant = endFrame?.size.height ?? 0.0
    }
    view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
}
}

```

Mein Problem war, dass ich diesen Code immer wieder für jeden einzelnen ViewController schrieb. Nachdem ich ein wenig experimentiert hatte, fand ich mit einem Singleton + Delegate-Muster die Möglichkeit, eine Reihe von Code wiederzuverwenden und das gesamte Keyboard Management an einem einzigen Ort zu organisieren!

Singleton + Delegate-Methode:

```

protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
}

class KeyboardManager {

    weak var delegate: KeyboardManagerDelegate?

    class var sharedInstance: KeyboardManager {
        struct Singleton {
            static let instance = KeyboardManager()
        }
        return Singleton.instance
    }

    init() {
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    @objc func keyboardWillChangeFrameNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)

        delegate?.keyboardWillChangeFrame(endFrame, duration: duration, animationCurve:
animationCurve)
    }
}

```

Wenn ich nun die Tastatur von einem ViewController aus verwalten möchte, muss ich nur den Delegierten auf diesen ViewController setzen und beliebige Delegatemethoden implementieren.

```

class ViewController: UIViewController {
    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)
        KeyboardManager.sharedInstance.delegate = self
    }
}

// MARK: - Keyboard Manager

extension ViewController: KeyboardManagerDelegate {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions) {
        if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
            lowerViewBottomConstraint.constant = 0
        } else {
            lowerViewBottomConstraint.constant = (endFrame?.size.height ?? 0.0)
        }
        view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
    }
}

```

Diese Methode ist auch sehr anpassbar! `UIKeyboardWillHideNotification` wir möchten die Funktionalität für `UIKeyboardWillHideNotification` hinzufügen. Dies ist so einfach wie das Hinzufügen einer Methode zu unserem `KeyboardManagerDelegate` .

`KeyboardManagerDelegate` mit `UIKeyboardWillHideNotification` :

```

protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
    func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])
}

class KeyboardManager {
    init() {
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
    }

    func keyboardWillHide(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }
        delegate?.keyboardWillHide(userInfo)
    }
}

```

`func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` wir möchten nur `func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` in einem `ViewController` implementieren. Wir können diese Methode auch optional machen.

```

typealias KeyboardManagerDelegate = protocol<KeyboardManagerModel,
KeyboardManagerConfigureable>

```

```
protocol KeyboardManagerModel: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
    UIViewAnimationOptions)
}

@objc protocol KeyboardManagerConfigurable {
    optional func keyboardWillHide(userInfo: [NSObject: AnyObject])
}
```

* Beachten Sie, dass dieses Muster den @objc von @objc . Siehe <http://www.jessesquires.com/avoiding-objc-in-swift/> für weitere Details!

Zusammenfassend habe ich festgestellt, dass die Verwendung eines Singleton + -Delegierten zur Verwaltung der Tastatur sowohl effizienter als auch einfacher zu verwenden ist als die Verwendung von Benachrichtigungen

Ansicht nach oben oder unten verschieben, wenn Tastatur vorhanden ist

Hinweis: Dies funktioniert nur für die von iOS bereitgestellte integrierte Tastatur

SCHNELL:

Damit die Ansicht eines **UIViewController** den Ursprung des Frames bei der **Darstellung** erhöht und beim **Ausblenden** verringert, fügen Sie Ihrer Klasse folgende Funktionen hinzu:

```
func keyboardWillShow(notification: NSNotification) {

    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
    NSValue)?.cgRectValue {
        if self.view.frame.origin.y == 0{
            self.view.frame.origin.y -= keyboardSize.height
        }
    }
}

func keyboardWillHide(notification: NSNotification) {
    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
    NSValue)?.cgRectValue {
        if self.view.frame.origin.y != 0{
            self.view.frame.origin.y += keyboardSize.height
        }
    }
}
```

viewDidLoad() in der viewDidLoad() Methode Ihrer Klasse die folgenden Beobachter hinzu:

```
NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillShow),
name: NSNotification.Name.UIKeyboardWillShow, object: nil)
NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillHide),
name: NSNotification.Name.UIKeyboardWillHide, object: nil)
```

Dies funktioniert bei jeder Bildschirmgröße und verwendet die Height-Eigenschaft der Tastatur.

ZIEL C:

Um das gleiche in Objective-C zu tun, kann dieser Code verwendet werden:

```
- (void) viewWillAppear:(BOOL) animated {
    [super viewWillAppear:animated];
    [[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];
}

- (void) viewWillDisappear:(BOOL) animated {
    [super viewWillDisappear:animated];
    [[NSNotificationCenter defaultCenter] removeObserver:self
 name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] removeObserver:self
 name:UIKeyboardWillHideNotification object:nil];
}

- (void) keyboardWillShow:(NSNotification *) notification
{
    CGSize keyboardSize = [[[notification userInfo]
 objectForKey:UIKeyboardFrameBeginUserInfoKey] CGRectValue].size;

    [UIView animateWithDuration:0.3 animations:^(
        CGRect f = self.view.frame;
        f.origin.y = -keyboardSize.height;
        self.view.frame = f;
    )];
}

- (void) keyboardWillHide:(NSNotification *) notification
{
    [UIView animateWithDuration:0.3 animations:^(
        CGRect f = self.view.frame;
        f.origin.y = 0.0f;
        self.view.frame = f;
    )];
}
```

Tastatur verwalten online lesen: <https://riptutorial.com/de/ios/topic/436/tastatur-verwalten>

Kapitel 151: Überholspur

Examples

Fastlane-Werkzeuge

[fastlane](#) ist ein Open Source Build-Automatisierungstool für Android und iOS für Entwickler. Es reduziert die Build-Generierungszeit. Es ist ein Befehlszeilentool, das [Ruby verwendet](#). Sie benötigen also Ruby auf Ihrem Computer. Bei den meisten Macs ist Ruby bereits standardmäßig installiert.

Installieren Sie Fastlane

1. Öffnen Sie ein Terminal.
2. Führen Sie `sudo gem install fastlane --verbose`
3. Wenn Sie die Xcode-Befehlszeilentools noch nicht installiert haben, führen Sie `xcode-select --install`, um sie zu installieren
4. Nun `cd` in Ihren Projektordner (geben Sie `cd` [mit dem Leerzeichen am Ende] ein und ziehen Sie Ihren Projektordner in das Terminal).
5. Führen Sie `fastlane init`, um die Einrichtung von `fastlane init` zu erhalten.
6. Jetzt können Sie alle Fastlane-Tools verwenden:

iOS-Tools

- [Lieferrn](#) : Laden Sie Screenshots, Metadaten und Ihre App in den App Store
- [Schnappschuss](#) : Automatisieren Sie die Erstellung lokalisierter Screenshots Ihrer iOS-App auf jedem Gerät
- [Frameit](#) : [Platzieren Sie](#) Ihre Screenshots schnell in die richtigen [Gerätebilder](#)
- [pem](#) : Generieren und erneuern Sie Ihre Push-Benachrichtigungsprofile automatisch
- [Seufzer](#) : Weil Sie Ihre Zeit lieber mit dem Aufbau von Sachen verbringen, als mit der Bereitstellung von Mitteln zu kämpfen
- [Produzieren](#) : Erstellen Sie neue iOS-Apps in iTunes Connect und Dev Portal über die Befehlszeile
- [cert](#) : Automatisches Erstellen und [Verwalten von iOS- Codesignaturzertifikaten](#)
- [Fitnessstudio](#) : Das Erstellen von iOS-Apps war noch nie einfacher
- [Spiel](#) : Synchronisieren Sie Ihre Zertifikate und Profile mit Git in Ihrem Team
- [Scan](#) : Die einfachste Möglichkeit, Tests für Ihre iOS- und Mac-Apps auszuführen
- [spaceship](#) : Ruby-Bibliothek für den Zugriff auf das Apple Dev Center und iTunes Connect

iOS TestFlight-Tools

- [pilot](#) : Der beste Weg, um Ihre TestFlight-Tester und Builds von Ihrem Terminal aus zu verwalten

- [Einsteigen](#) : Der einfachste Weg, Ihre TestFlight-Betatester einzuladen

Android-Tools

- [Angebot](#) : Laden Sie Ihre Android-App und ihre Metadaten in Google Play hoch
- [screengrab](#) : Automatisieren Sie auf jedem Gerät lokalisierte Screenshots Ihrer Android-App

Überholspur online lesen: <https://riptutorial.com/de/ios/topic/3574/uberholspur>

Kapitel 152: UIA Auftritt

Examples

Legt das Aussehen aller Instanzen der Klasse fest

Um das Erscheinungsbild aller Instanzen einer Klasse anzupassen, greifen Sie auf das Erscheinungsbild der gewünschten Klasse zu. Zum Beispiel:

UIButton-Farbton einstellen

Schnell:

```
UIButton.appearance().tintColor = UIColor.greenColor()
```

Ziel c:

```
[UIButton appearance].tintColor = [UIColor greenColor];
```

UIButton-Hintergrundfarbe einstellen

Schnell:

```
UIButton.appearance().backgroundColor = UIColor.blueColor()
```

Ziel c:

```
[UIButton appearance].backgroundColor = [UIColor blueColor];
```

Legen Sie die Textfarbe für UILabel fest

Schnell:

```
UILabel.appearance().textColor = UIColor.redColor()
```

Ziel c:

```
[UILabel appearance].textColor = [UIColor redColor];
```

UILabel-Hintergrundfarbe einstellen

Schnell:

```
UILabel.appearance().backgroundColor = UIColor.greenColor()
```

Ziel c:


```
[UILabel appearance].backgroundColor = [UIColor greenColor];
```

Legen Sie die Farbtonfarbe für UINavigationController fest

Schnell:

```
UINavigationController.appearance().tintColor = UIColor.cyanColor()
```

Ziel c:

```
[UINavigationController appearance].tintColor = [UIColor cyanColor];
```

Legen Sie die Hintergrundfarbe für UINavigationController fest

Schnell:

```
UINavigationController.appearance().backgroundColor = UIColor.redColor()
```

Ziel c:

```
[UINavigationController appearance].backgroundColor = [UIColor redColor];
```

Aussehen für die Klasse, wenn sie in der Containerklasse enthalten ist

Verwenden Sie `appearanceWhenContainedInInstancesOfClasses`: Zum Anpassen der Darstellung einer Instanz einer Klasse, wenn diese in einer Instanz der Containerklasse enthalten ist. Zum Beispiel der Anpassung von `UILabel`, `textColor` und `backgroundColor` innerhalb `ViewController` Klasse wird wie folgt aussehen:

Legen Sie die Textfarbe für UILabel fest

Schnell:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).textColor =  
UIColor.whiteColor()
```

Ziel c:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[ViewController class]].textColor =  
[UIColor whiteColor];
```

UILabel-Hintergrundfarbe einstellen

Schnell:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).backgroundColor =  
UIColor.blueColor()
```

Ziel c:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController  
class]].backgroundColor = [UIColor blueColor];
```

UIA Auftritt online lesen: <https://riptutorial.com/de/ios/topic/3422/ui-auftritt>

Kapitel 153: UIActivityViewController

Parameter

Parametername	Beschreibung
ActivityItems	Enthält ein Array von Objekten, um die Aktivität auszuführen. Dieses Array darf nicht null sein und muss mindestens ein Objekt enthalten.
applicationActivities	Ein Array von UIActivity-Objekten, das die benutzerdefinierten Dienste darstellt, die Ihre Anwendung unterstützt. Dieser Parameter kann Null sein.

Examples

Activity View Controller initialisieren

Ziel c

```
NSString *textToShare = @"StackOverflow Documentation!! Together, we can do for Documentation what we did for Q&A.";
NSURL *documentationURL = [NSURL
    URLWithString:@"http://stackoverflow.com/tour/documentation"];

NSArray *objectsToShare = @[textToShare, documentationURL];

UIActivityViewController *activityVC = [[UIActivityViewController alloc]
    initWithActivityItems:objectsToShare applicationActivities:nil];

[self presentViewController:activityVC animated:YES completion:nil];
```

Schnell

```
let textToShare = "StackOverflow Documentation!! Together, we can do for Documentation what we did for Q&A."
let documentationURL = NSURL(string:"http://stackoverflow.com/tour/documentation")

let objToShare : [AnyObject] = [textToShare, documentationURL!]

let activityVC = UIActivityViewController(activityItems: objToShare, applicationActivities: nil)
self.presentViewController(activityVC, animated: true, completion: nil)
```

UIActivityViewController online lesen:

<https://riptutorial.com/de/ios/topic/2889/uiactivityviewcontroller>

Kapitel 154: UIAlertController

Bemerkungen

Ein `UIAlertController` Objekt zeigt dem Benutzer eine Warnmeldung an. Diese Klasse ersetzt die `UIActionSheet` und `UIAlertView` Klassen zum Anzeigen von Warnungen. Nachdem Sie den Alert Controller mit den gewünschten Aktionen und dem `presentViewController:animated:completion:` Stil konfiguriert haben, präsentieren Sie ihn mit der Methode `presentViewController:animated:completion:`

Aus der [Apple-Dokumentation](#)

[UIAlertController in Swift](#)

Examples

AlertViews mit UIAlertController

`UIAlertView` und `UIActionSheet` werden in iOS 8 und `UIActionSheet` mehr unterstützt. Daher hat Apple einen neuen Controller für `alertView` und `actionSheet` namens `UIAlertController`, mit dem Sie den `preferredStyle` `alertView` `actionSheet`. Sie können zwischen `alertView` und `actionSheet`. Es gibt keine Delegatenmethode, da alle Schaltflächenereignisse in ihren Blöcken behandelt werden.

Einfache UIAlertView

Schnell:

```
let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Cancel", style: .cancel) { _ in
    print("Cancel")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
    print("OK")
})

present(alert, animated: true)
```

Ziel c:

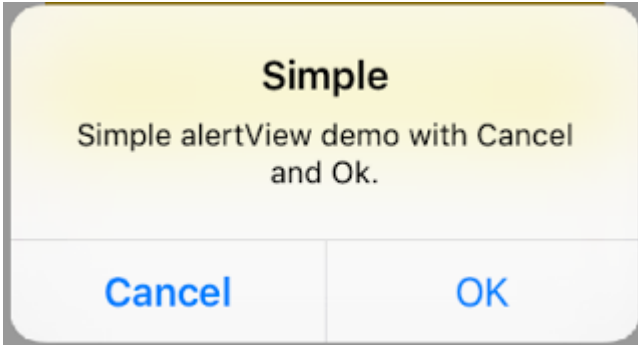
```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Simple"
message:@"Simple alertView demo with Cancel and OK."
preferredStyle:UIAlertControllerStyleAlert];
UIAlertAction *cancelAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    NSLog(@"Cancel");
}];
UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
```

```

style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    NSLog(@"OK");
}];

[alertController addAction:cancelAction];
[alertController addAction:okAction];
[self presentViewController:alertController animated: YES completion: nil];

```



Zerstörerische UIAlertView

Schnell:

```

let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Destructive", style: .destructive) { _ in
    print("Destructive")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
    print("OK")
})

present(alert, animated: true)

```

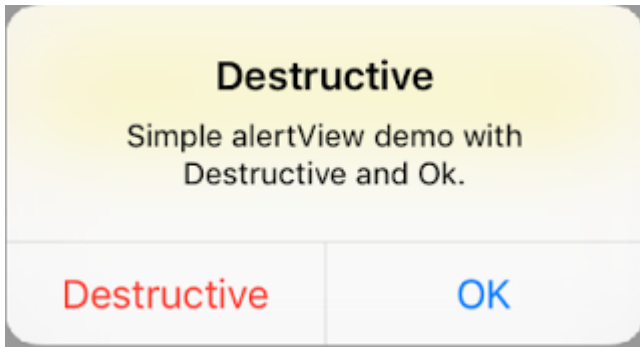
Ziel c:

```

UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"Destructive" message:@"Simple alertView demo with Destructive and OK." preferredStyle:UIAlertControllerStyleAlert];
    UIAlertAction *destructiveAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
    NSLog(@"Destructive");
}];
    UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    NSLog(@"OK");
}];

[alertController addAction:destructiveAction];
[alertController addAction:okAction];
[self presentViewController:alertController animated: YES completion: nil];

```



Vorübergehendes toastähnliches Popup

Gut für schnelle Benachrichtigungen, die keine Interaktion erfordern.

Schnell

```
let alert = UIAlertController(title: "Toast", message: "Hello World", preferredStyle: .Alert)

presentViewController(alert, animated: true) {
    let delay_s:Double = 2
    let delayTime = dispatch_time(DISPATCH_TIME_NOW, Int64(delay_s * Double(NSEC_PER_SEC)))
    dispatch_after(delayTime, dispatch_get_main_queue()) {
        alert.dismissViewControllerAnimated(true, completion: nil)
    }
}
```

Textfeld in UIAlertController wie eine Eingabeaufforderung hinzufügen

Schnell

```
let alert = UIAlertController(title: "Hello",
                             message: "Welcome to the world of iOS",
                             preferredStyle: UIAlertControllerStyle.alert)

let defaultAction = UIAlertAction(title: "OK", style: UIAlertActionStyle.default) { (action)
in
}
defaultAction.isEnabled = false
alert.addAction(defaultAction)

alert.addTextFieldWithConfigurationHandler { (textField) in
    textField.delegate = self
}

present(alert, animated: true, completion: nil)
```

Ziel c

```
UIAlertController* alert = [UIAlertController alertControllerWithTitle:@"Hello"
                                                                    message:@"Welcome to the world
of iOS"
```

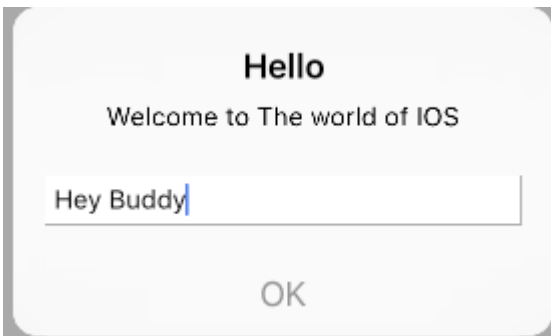
```
preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction* defaultAction = [UIAlertAction actionWithTitle:@"OK"
                                                                    style:UIAlertActionStyleDefault
                                                                    handler:^(UIAlertAction * action) {}];

defaultAction.enabled = NO;
[alert addAction:defaultAction];

[alert addTextFieldWithConfigurationHandler:^(UITextField *textField) {
    textField.delegate = self;
}];

[self presentViewController:alert animated:YES completion:nil];
```



Aktionsblätter mit UIAlertController

Mit `UIAlertController` werden Aktionsblätter wie das veraltete `UIActionSheet` mit derselben API erstellt, die Sie für `AlertViews` verwenden.

Einfaches Aktionsblatt mit zwei Tasten

Schnell

```
let alertController = UIAlertController(title: "Demo", message: "A demo with two buttons",
preferredStyle: UIAlertControllerStyle.actionSheet)
```

Ziel c

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Demo"
message:@"A demo with two buttons" preferredStyle:UIAlertControllerStyleActionSheet];
```

Erstellen Sie die Schaltflächen "Abbrechen" und "OK".

Schnell

```
let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) { (result : UIAlertAction) -
```

```
> Void in
    //action when pressed button
}
let okAction = UIAlertAction(title: "Okay", style: .default) { (result : UIAlertAction) ->
Void in
    //action when pressed button
}
```

Ziel c

```
UIAlertAction *cancelAction = [UIAlertAction initWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    //action when pressed button
}];

UIAlertAction * okAction = [UIAlertAction initWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    //action when pressed button
}];
```

Und füge sie dem Aktionsblatt hinzu:

Schnell

```
alertController.addAction(cancelAction)
alertController.addAction(okAction)
```

Ziel c

```
[alertController addAction:cancelAction];
[alertController addAction:okAction];
```

Jetzt präsentieren Sie den `UIAlertController` :

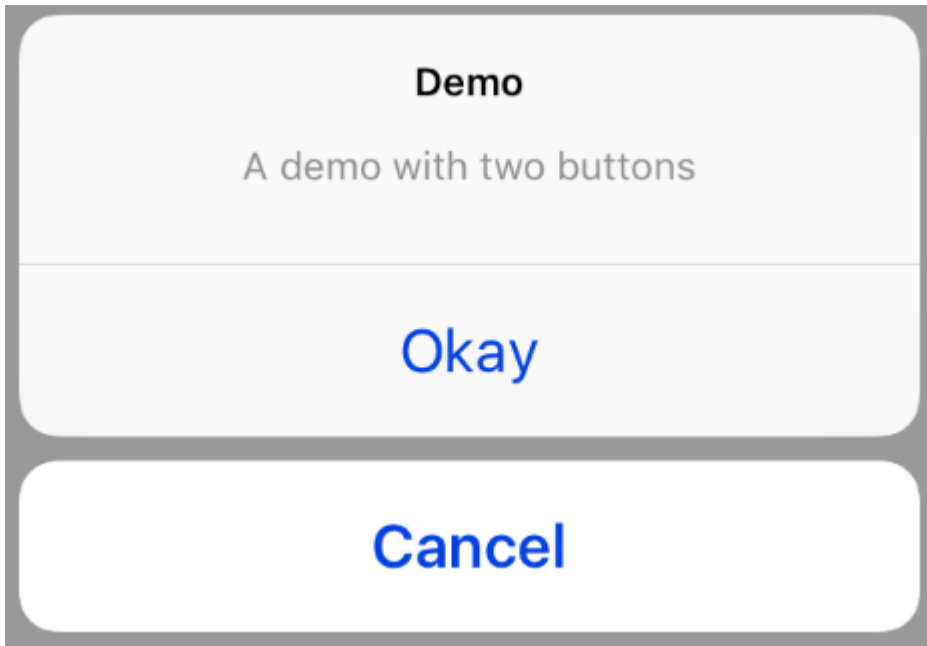
Schnell

```
self.present(alertController, animated: true, completion: nil)
```

Ziel c

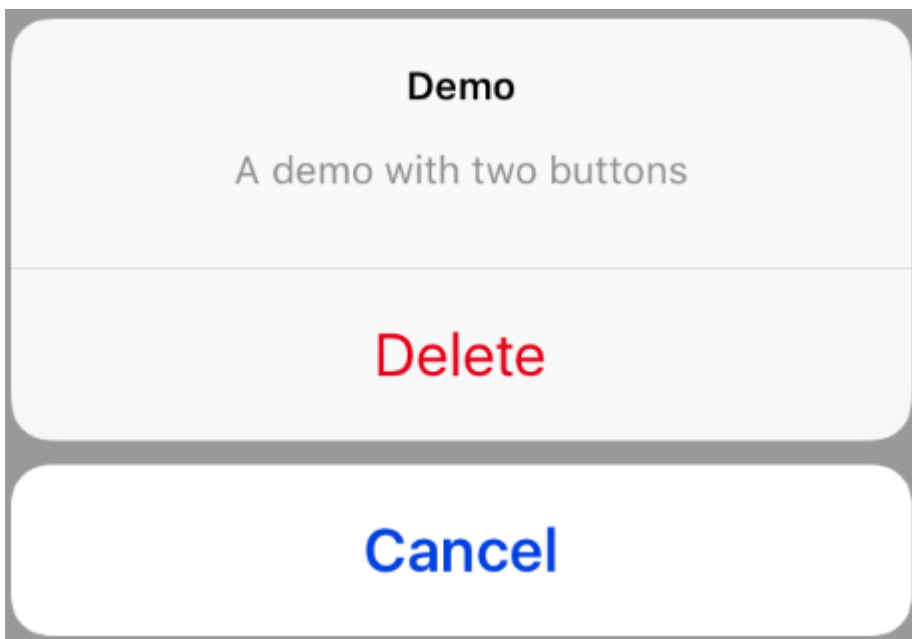
```
[self presentViewController:alertController animated: YES completion: nil];
```

Das sollte das Ergebnis sein:



Aktionsblatt mit zerstörerischem Button

Bei Verwendung des `UIAlertActionStyle.destructive` für eine `UIAlertAction` wird eine Schaltfläche mit roter Tönungsfarbe erstellt.



In diesem Beispiel wurde die `okAction` genannte `UIAlertAction` durch diese `UIAlertAction` :

Schnell

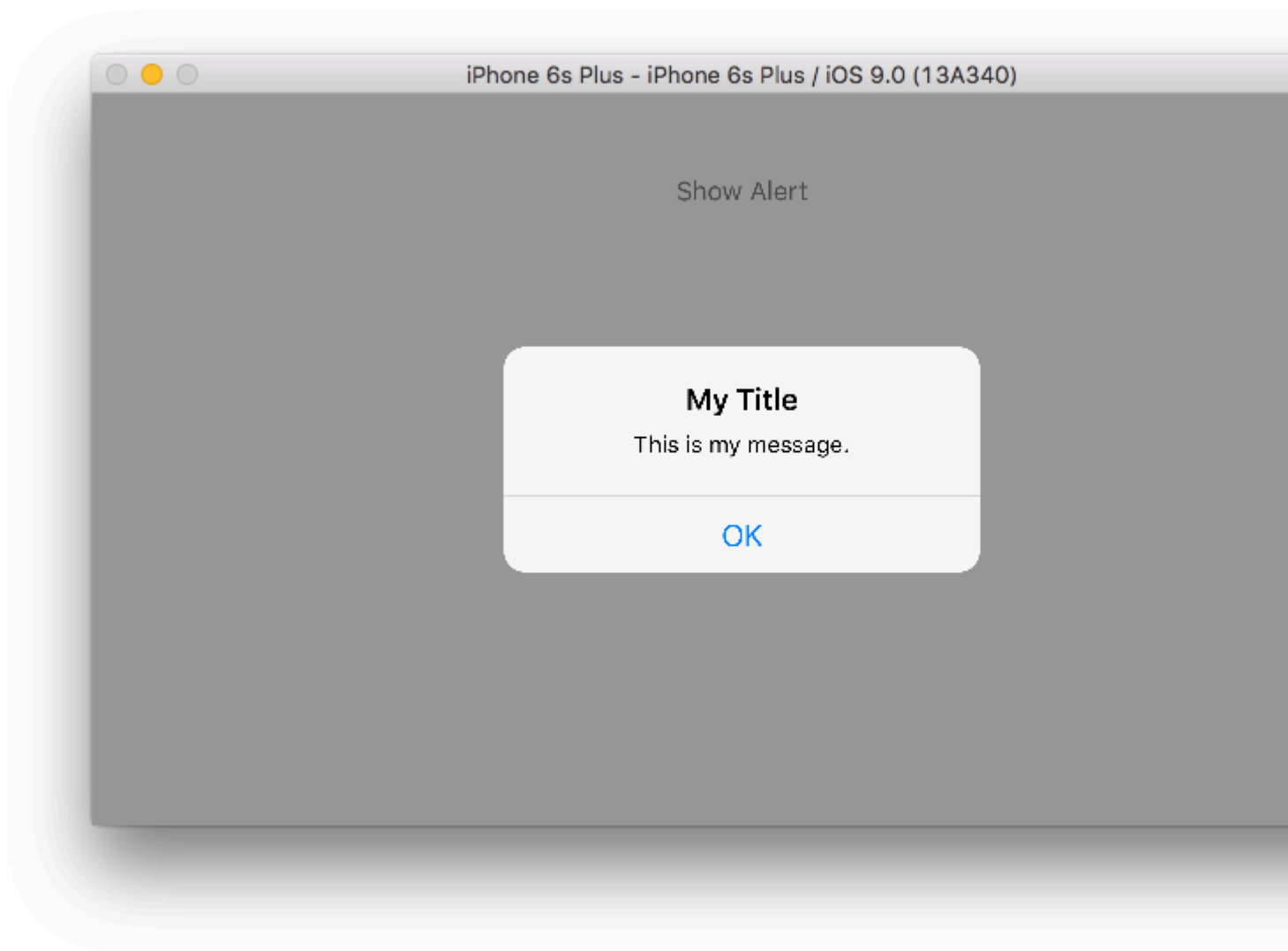
```
let destructiveAction = UIAlertAction(title: "Delete", style: .destructive) { (result :  
UIAlertAction) -> Void in  
    //action when pressed button  
}
```

Ziel c

```
UIAlertAction * destructiveAction = [UIAlertAction actionWithTitle:@"Delete"  
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {  
    //action when pressed button  
}];
```

Anzeigen und Verarbeiten von Warnungen

Ein Knopf



Schnell

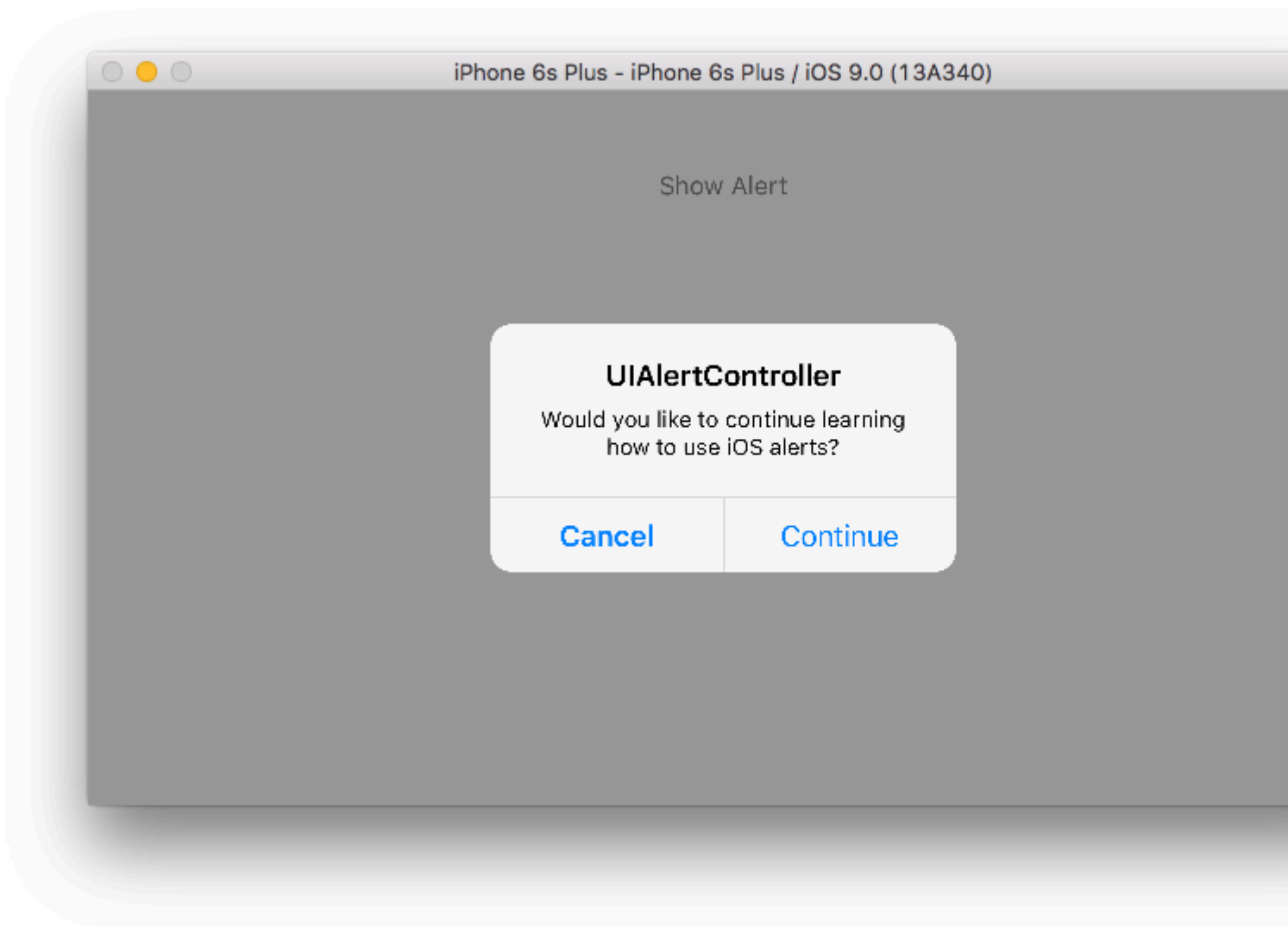
```
class ViewController: UIViewController {  
  
    @IBAction func showAlertButtonTapped(sender: UIButton) {  
  
        // create the alert
```

```
    let alert = UIAlertController(title: "My Title", message: "This is my message.",
preferredStyle: UIAlertControllerStyle.Alert)

    // add an action (button)
    alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default, handler:
nil))

    // show the alert
    self.presentViewController(alert, animated: true, completion: nil)
}
}
```

Zwei Knöpfe



Schnell

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {

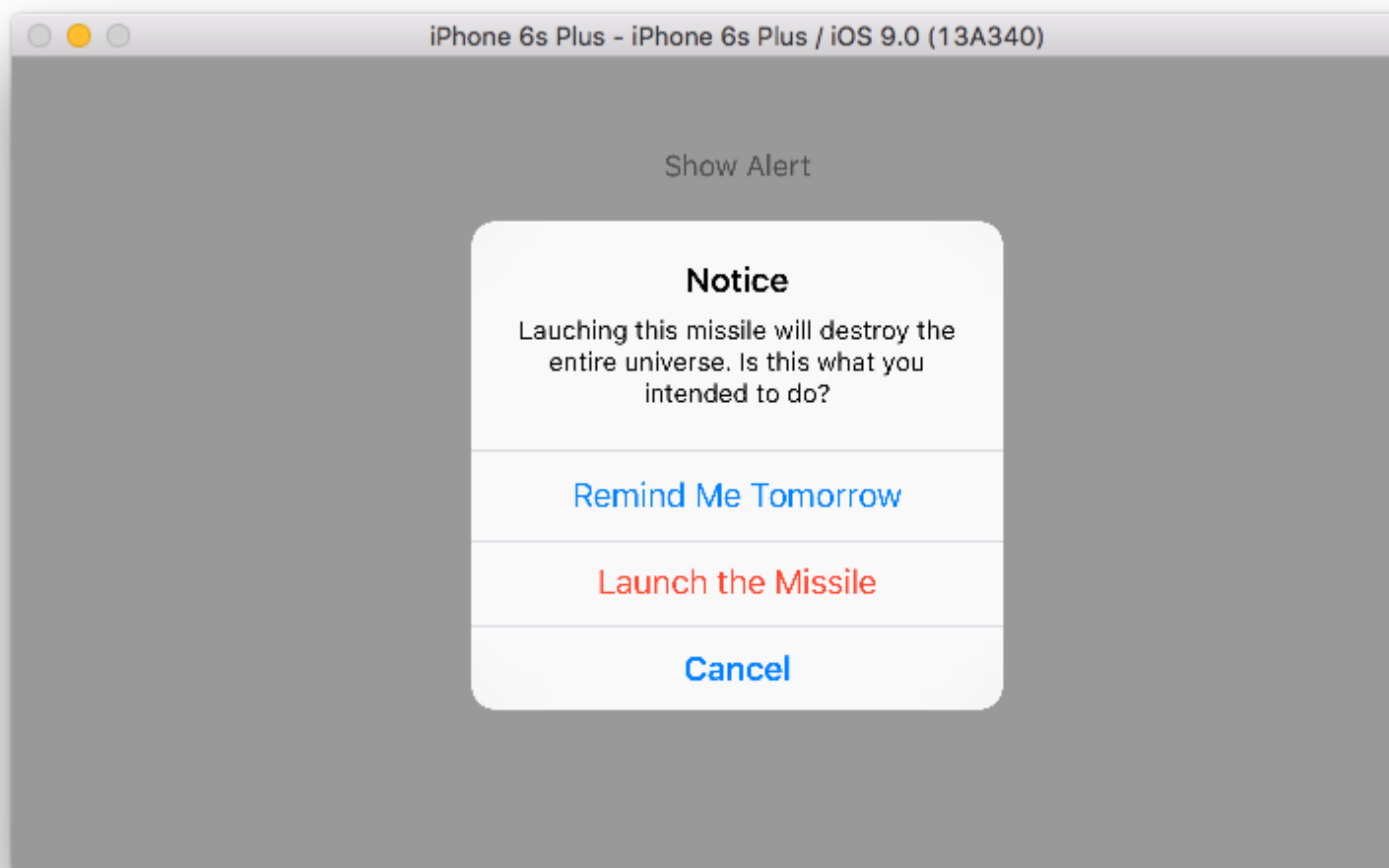
        // create the alert
```

```
let alert = UIAlertController(title: "UIAlertController", message: "Would you like to
continue learning how to use iOS alerts?", preferredStyle: UIAlertControllerStyle.Alert)

// add the actions (buttons)
alert.addAction(UIAlertAction(title: "Continue", style: UIAlertActionStyle.Default,
handler: nil))
alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.Cancel,
handler: nil))

// show the alert
self.presentViewController(alert, animated: true, completion: nil)
}
```

Drei Knöpfe



Schnell

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {
```

```

    // create the alert
    let alert = UIAlertController(title: "Notice", message: "Lauching this missile will
destroy the entire universe. Is this what you intended to do?", preferredStyle:
UIAlertControllerStyle.Alert)

    // add the actions (buttons)
    alert.addAction(UIAlertAction(title: "Remind Me Tomorrow", style:
UIAlertActionStyle.Default, handler: nil))
    alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertController.Cancel,
handler: nil))
    alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: nil))

    // show the alert
    self.presentViewController(alert, animated: true, completion: nil)
}
}

```

Handhabung von Tastenanschlägen

Der `handler` war in den obigen Beispielen gleich `nil`. Sie können `nil` durch einen [Abschluss](#) ersetzen, um etwas zu tun, wenn der Benutzer auf eine Schaltfläche tippt.

Schnell

```

alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: { action in

    // do something like...
    self.launchMissile()

}))

```

Anmerkungen

- Mehrere Schaltflächen müssen nicht unbedingt verschiedene `UIAlertActionStyle` Typen verwenden. Sie könnten alle `.Default`.
- Bei mehr als drei Schaltflächen sollten Sie ein Aktionsblatt verwenden. Das Setup ist sehr ähnlich. [Hier ist ein Beispiel](#).

Eine Aktionsschaltfläche hervorheben

Der Alert Controller verfügt über eine Eigenschaft, mit der eine im Alert Controller hinzugefügte Aktion hervorgehoben wird. Diese Eigenschaft kann verwendet werden, um eine bestimmte Aktion für die Aufmerksamkeit des Benutzers hervorzuheben. Für Ziel C;

```
@property(nonatomic, strong) UIAlertAction *preferredAction
```

Eine **bereits im Alert Controller hinzugefügte** Aktion kann dieser Eigenschaft zugewiesen werden. Der Alert Controller wird diese Aktion hervorheben.

Diese Eigenschaft kann nur mit UIAlertControllerStyleAlert verwendet werden.

Das folgende Beispiel zeigt die Verwendung.

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Cancel edit" message:@"Are you really want to cancel your edit?" preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *cancel = [UIAlertAction actionWithTitle:@"Cancel" style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    NSLog(@"Cancel");
}];

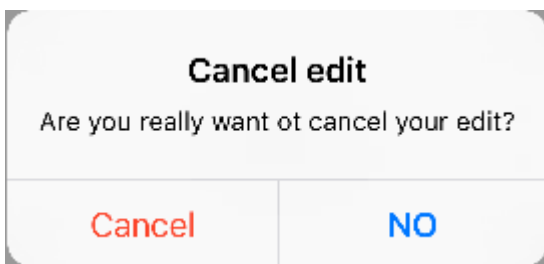
UIAlertAction *no = [UIAlertAction actionWithTitle:@"NO" style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    NSLog(@"Highlighted button is pressed.");
}];

[alertController addAction:cancel];
[alertController addAction:no];

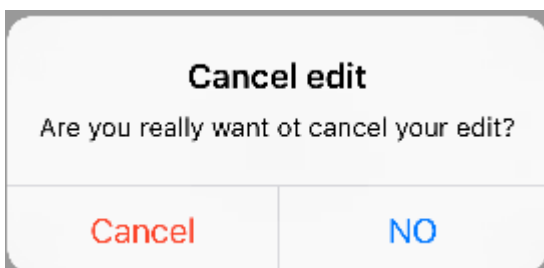
//add no action to preferred action.
//Note
//the action should already be added to alert controller
alertController.preferredAction = no;

[self presentViewController:alertController animated: YES completion: nil];
```

Alarm Controller mit dem **bevorzugten Aktionssatz**. Die Schaltfläche **NO** ist hervorgehoben.



Alarm Controller mit **bevorzugter Aktion nicht eingestellt**. Die Schaltfläche **NO** wird nicht hervorgehoben.



UIAlertController online lesen: <https://riptutorial.com/de/ios/topic/874/uialertcontroller>

Kapitel 155: UIBarButtonItem

Parameter

Parameter	Beschreibung
Titel	Der UIBarButtonItem-Titel
Stil	Der Stil des UIBarButtonItem
Ziel	Das Objekt, das die UIBarButtonItem-Aktion empfangen soll
Aktion	Die Auswahl (Methode), die ausgeführt werden soll, wenn UIBarButtonItem gedrückt wird

Bemerkungen

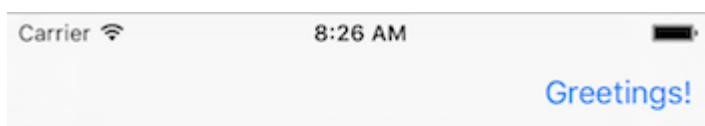
Beim Verweisen auf `self.navigationItem` davon `self.navigationItem`, dass der UIViewController in einen UINavigationController eingebettet ist.

Examples

Erstellen eines UIBarButtonItem

```
//Swift
let barButtonItem = UIBarButtonItem(title: "Greetings!", style: .Plain, target: self, action:
#selector(barButtonTapped))
self.navigationItem.rightBarButtonItem = barButtonItem

//Objective-C
UIBarButtonItem *barButtonItem = [[UIBarButtonItem alloc] initWithTitle:@"Greetings!"
style:UIBarButtonItemStylePlain target:self action:@selector(barButtonTapped)];
self.navigationItem.rightBarButtonItem = barButtonItem;
```

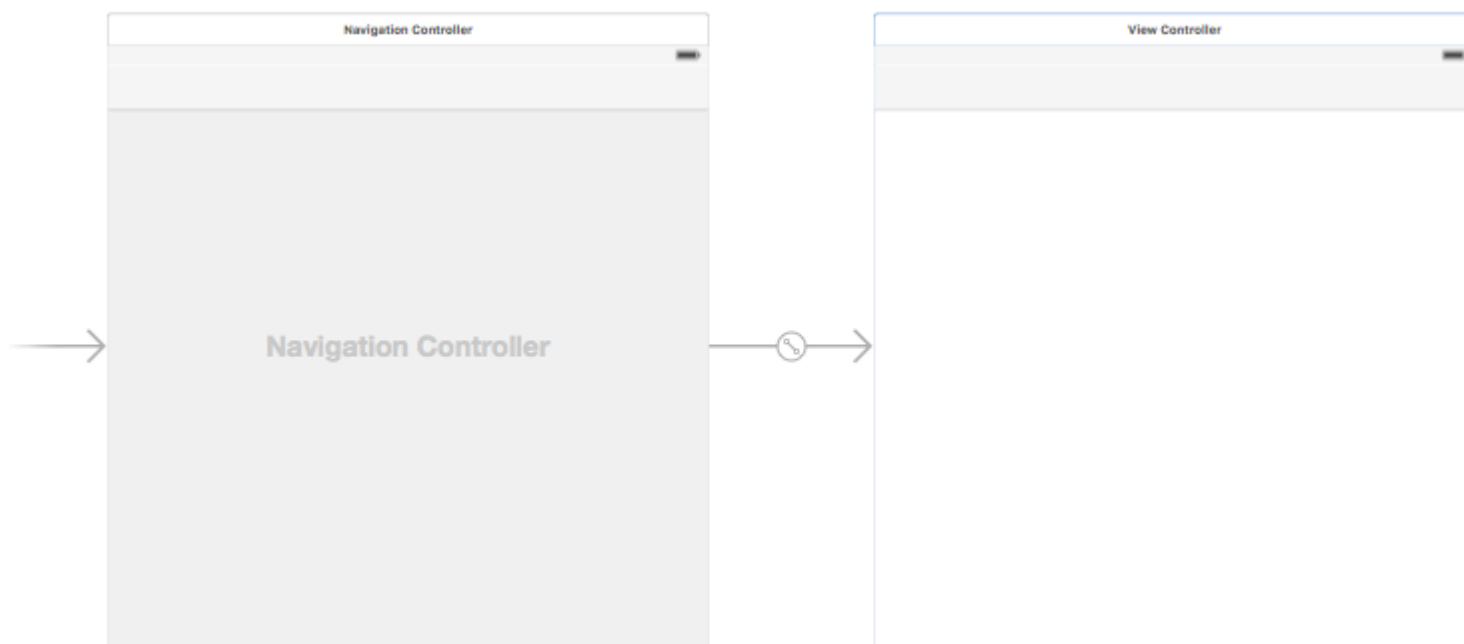


Erstellen eines UIBarButtonItem im Interface Builder

Das folgende Beispiel zeigt, wie UIBarButtonItem im Interface Builder eine UIBarButtonItem für die Navigationsleiste (UIBarButtonItem) hinzufügen.

Fügen Sie Ihrem Storyboard einen Navigations-Controller hinzu

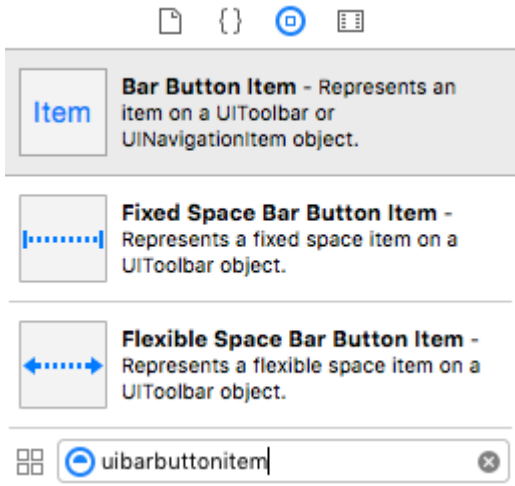
Wählen Sie Ihren View Controller aus und wählen Sie dann im Xcode-Menü **Editor> Einbetten in> Navigationscontroller** .



Alternativ können Sie eine `UINavigationController` aus der `UINavigationController` hinzufügen.

Fügen Sie ein Bar-Schaltflächenelement hinzu

Ziehen Sie ein `UIBarButtonItem` aus der `UIBarButtonItem` in die obere Navigationsleiste.

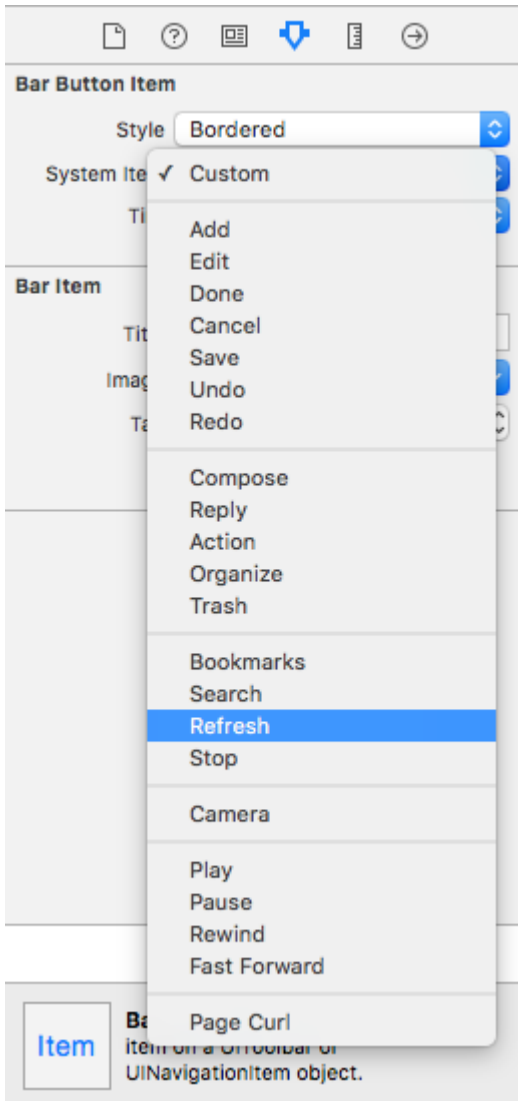


Es sollte so aussehen:



Legen Sie die Attribute fest

Sie können auf "Element" doppelklicken, um den Text in etwas wie "Aktualisieren" zu ändern. Es gibt jedoch ein Symbol für *Aktualisieren*, das Sie verwenden können. `UIBarButtonItem` einfach den Inspektorattribute für `UIBarButtonItem` und für **System Item** die `UIBarButtonItem` **Refresh aus**.



Dadurch erhalten Sie das Standard-Aktualisierungssymbol.



Fügen Sie eine IB-Aktion hinzu

Ziehen Sie das `UIBarButtonItem` per Drag `UIBarButtonItem` in den View Controller, um eine `@IBAction` hinzuzufügen.

```
class ViewController: UIViewController {
```

```
@IBAction func refreshBarButtonItemTap(sender: UIBarButtonItem) {  
    print("How refreshing!")  
}  
}
```

Das ist es.

Anmerkungen

- Dieses Beispiel stammt ursprünglich aus [dieser Stack Overflow-Antwort](#) .

Bar Button Element Originalbild ohne Farbton

Vorausgesetzt, dass `BarButtonItem` über eine Nicht-Null-Image-Eigenschaft verfügt (z. B. im Interface Builder festgelegt).

Ziel c

```
BarButtonItem.image = [BarButtonItem.image  
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
```

UIBarButtonItem online lesen: <https://riptutorial.com/de/ios/topic/1543/uiBarButtonItem>

Kapitel 156: UIBezierPath

Examples

Wie wird ein Eckenradius auf von UIBezierPath gezeichnete Rechtecke angewendet?

Eckenradius für alle 4 Kanten:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) cornerRadius: 11];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für oberen linken Rand:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für oberen rechten Rand:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für linken unteren Rand:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für rechten unteren Rand:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für Unterkanten:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft |
UIRectCornerBottomRight cornerRadii: CGSizeMake(11, 11)];
```

```
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Eckenradius für Oberkanten:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft | UIRectCornerTopRight
cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

So erstellen Sie einfache Formen mit UIBezierPath

Für einen einfachen Kreis:



```
UIBezierPath* ovalPath = [UIBezierPath bezierPathWithOvalInRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[ovalPath fill];
```

Schnell:

```
let ovalPath = UIBezierPath(ovalInRect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
ovalPath.fill()
```

Für ein einfaches Rechteck:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Schnell:

```
let rectanglePath = UIBezierPath(rect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
rectanglePath.fill()
```

Für eine einfache Leitung:



```
UIBezierPath* bezierPath = [UIBezierPath bezierPath];
[bezierPath moveToPoint: CGPointMake(x1,y1)];
[bezierPath addLineToPoint: CGPointMake(x2,y2)];
[UIColor.blackColor setStroke];
bezierPath.lineWidth = 1;
[bezierPath stroke];
```

Schnell:

```
let bezierPath = UIBezierPath()
bezierPath.moveToPoint(CGPoint(x: x1, y: y1))
bezierPath.addLineToPoint(CGPoint(x: x2, y: y2))
UIColor.blackColor().setStroke()
bezierPath.lineWidth = 1
bezierPath.stroke()
```

Für einen halben Kreis:



```
CGRect ovalRect = CGRectMake(x,y,width,height);
UIBezierPath* ovalPath = [UIBezierPath bezierPath];
[ovalPath addArcWithCenter: CGPointMake(0, 0) radius: CGRectGetWidth(ovalRect) / 2 startAngle:
180 * M_PI/180 endAngle: 0 * M_PI/180 clockwise: YES];
[ovalPath addLineToPoint: CGPointMake(0, 0)];
[ovalPath closePath];
```

```
CGAffineTransform ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect));
ovalTransform = CGAffineTransformScale(ovalTransform, 1, CGRectGetHeight(ovalRect) /
CGRectGetWidth(ovalRect));
[ovalPath applyTransform: ovalTransform];

[UIColor.grayColor setFill];
[ovalPath fill];
```

Schnell:

```
let ovalRect = CGRect(x: 0, y: 0, width: 50, height: 50)
let ovalPath = UIBezierPath()
ovalPath.addArcWithCenter(CGPoint.zero, radius: ovalRect.width / 2, startAngle: 180 *
CGFloat(M_PI)/180, endAngle: 0 * CGFloat(M_PI)/180, clockwise: true)
ovalPath.addLineToPoint(CGPoint.zero)
ovalPath.closePath()

var ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect))
ovalTransform = CGAffineTransformScale(ovalTransform, 1, ovalRect.height / ovalRect.width)
ovalPath.applyTransform(ovalTransform)

UIColor.grayColor().setFill()
ovalPath.fill()
```

Für ein einfaches Dreieck:



```
UIBezierPath* polygonPath = [UIBezierPath bezierPath];
[polygonPath moveToPoint: CGPointMake(x1, y1)];
[polygonPath addLineToPoint: CGPointMake(x2, y2)];
[polygonPath addLineToPoint: CGPointMake(x3, y2)];
[polygonPath closePath];
[UIColor.grayColor setFill];
[polygonPath fill];
```

Schnell:

```
let polygonPath = UIBezierPath()
polygonPath.moveToPoint(CGPoint(x: x1, y: y1))
polygonPath.addLineToPoint(CGPoint(x: x2, y: y2))
polygonPath.addLineToPoint(CGPoint(x: x3, y: y3))
polygonPath.closePath()
UIColor.grayColor().setFill()
polygonPath.fill()
```

UIBezierPath + AutoLayout

Um die Größe des Bezier-Pfads basierend auf dem Ansichtsrahmen zu ändern, überschreiben Sie

die drawRect-Ansicht, die Sie den Bezier-Pfad zeichnen:

```
- (void)drawRect:(CGRect) frame
{
    UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
    CGRectMake(CGRectGetMinX(frame), CGRectGetMinY(frame), CGRectGetWidth(frame),
    CGRectGetHeight(frame))];
    [UIColor.grayColor setFill];
    [rectanglePath fill];
}
```

So wenden Sie Schatten auf UIBezierPath an

Betrachten Sie ein einfaches Rechteck, das vom Bezier-Pfad gezeichnet wird.



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(x,y,width,height)];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Grundlegender Außenschatten:



```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(7.1, 5.1)];
[shadow setShadowBlurRadius: 5];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
CGContextSaveGState(context);
CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[shadow.shadowColor CGColor]);
[UIColor.grayColor setFill];
[rectanglePath fill];
CGContextRestoreGState(context);
```

Grundfüllender Innenschatten:



```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(9.1, -7.1)];
[shadow setShadowBlurRadius: 6];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
[UIColor.grayColor setFill];
[rectanglePath fill];

CGContextSaveGState(context);
UIRectClip(rectanglePath.bounds);
CGContextSetShadowWithColor(context, CGSizeZero, 0, NULL);

CGContextSetAlpha(context, CGColorGetAlpha([shadow.shadowColor CGColor]));
CGContextBeginTransparencyLayer(context, NULL);
{
    UIColor* opaqueShadow = [shadow.shadowColor colorWithAlphaComponent: 1];
    CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[opaqueShadow CGColor]);
    CGContextSetBlendMode(context, kCGBlendModeSourceOut);
    CGContextBeginTransparencyLayer(context, NULL);

    [opaqueShadow setFill];
    [rectanglePath fill];

    CGContextEndTransparencyLayer(context);
}
CGContextEndTransparencyLayer(context);
CGContextRestoreGState(context);
```

Entwerfen und Zeichnen eines Bézierpfads

Dieses Beispiel zeigt den Prozess vom Entwerfen der Form, die Sie in einer Ansicht zeichnen möchten. Eine bestimmte Form wird verwendet, aber die Konzepte, die Sie lernen, können auf jede Form angewendet werden.

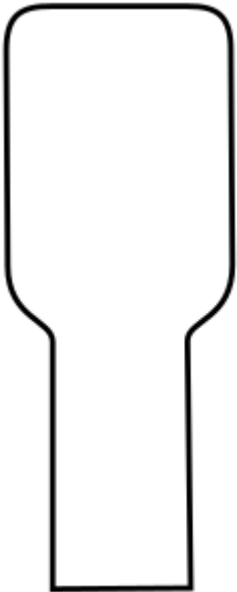
So zeichnen Sie einen Bézier-Pfad in einer benutzerdefinierten Ansicht

Dies sind die Hauptschritte:

1. Gestalten Sie den Umriss der gewünschten Form.
2. Teilen Sie den Umrisspfad in Segmente von Linien, Bögen und Kurven.
3. Bauen Sie diesen Pfad programmgesteuert auf.
4. Zeichnen Sie den Pfad entweder in `drawRect` oder mit einem `CAShapeLayer` .

Umriss der Form gestalten

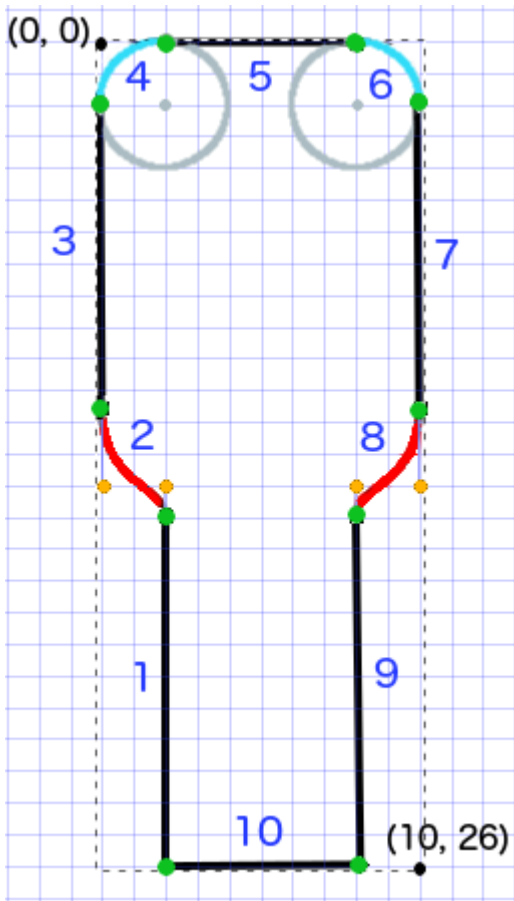
Sie können alles tun, aber als Beispiel habe ich die Form unten ausgewählt. Es könnte sich um eine Popup-Taste auf einer Tastatur handeln.



Teilen Sie den Pfad in Segmente

Sehen Sie sich Ihr Formdesign an und unterteilen Sie es in einfachere Linienelemente (für gerade Linien), Bögen (für Kreise und runde Ecken) und Kurven (für alles andere).

So würde unser Beispieldesign aussehen:



- Schwarz sind Liniensegmente
- Hellblau sind Bogensegmente
- Rot sind Kurven
- Orange Punkte sind die Kontrollpunkte für die Kurven
- Grüne Punkte sind die Punkte zwischen Pfadsegmenten
- Gepunktete Linien zeigen das Begrenzungsrechteck
- Dunkelblaue Zahlen sind die Segmente in der Reihenfolge, in der sie programmgesteuert hinzugefügt werden

Bauen Sie den Pfad programmgesteuert auf

Wir beginnen willkürlich in der unteren linken Ecke und arbeiten im Uhrzeigersinn. Ich verwende das Raster im Bild, um die x- und y-Werte für die Punkte zu erhalten. Ich werde hier alles hartcodieren, aber das würde man in einem echten Projekt natürlich nicht tun.

Der grundlegende Prozess ist:

1. Erstellen Sie einen neuen `UIBezierPath`
2. Wählen Sie mit `moveToPoint` einen Startpunkt auf dem Pfad
3. Fügen Sie dem Pfad Segmente hinzu

- Zeile: `addLineToPoint`
- arc: `addArcWithCenter`
- Kurve: `addCurveToPoint`

4. Schließen Sie den Pfad mit `closePath`

Hier ist der Code für den Pfad im obigen Bild.

```
func createBezierPath() -> UIBezierPath {

    // create a new path
    let path = UIBezierPath()

    // starting point for the path (bottom left)
    path.moveToPoint(CGPoint(x: 2, y: 26))

    // *****
    // ***** Left side *****
    // *****

    // segment 1: line
    path.addLineToPoint(CGPoint(x: 2, y: 15))

    // segment 2: curve
    path.addCurveToPoint(CGPoint(x: 0, y: 12), // ending point
        controlPoint1: CGPoint(x: 2, y: 14),
        controlPoint2: CGPoint(x: 0, y: 14))

    // segment 3: line
    path.addLineToPoint(CGPoint(x: 0, y: 2))

    // *****
    // ***** Top side *****
    // *****

    // segment 4: arc
    path.addArcWithCenter(CGPoint(x: 2, y: 2), // center point of circle
        radius: 2, // this will make it meet our path line
        startAngle: CGFloat(M_PI), // π radians = 180 degrees = straight left
        endAngle: CGFloat(3*M_PI_2), // 3π/2 radians = 270 degrees = straight up
        clockwise: true) // startAngle to endAngle goes in a clockwise direction

    // segment 5: line
    path.addLineToPoint(CGPoint(x: 8, y: 0))

    // segment 6: arc
    path.addArcWithCenter(CGPoint(x: 8, y: 2),
        radius: 2,
        startAngle: CGFloat(3*M_PI_2), // straight up
        endAngle: CGFloat(0), // 0 radians = straight right
        clockwise: true)

    // *****
    // ***** Right side *****
    // *****

    // segment 7: line
    path.addLineToPoint(CGPoint(x: 10, y: 12))

    // segment 8: curve
    path.addCurveToPoint(CGPoint(x: 8, y: 15), // ending point
        controlPoint1: CGPoint(x: 10, y: 14),
        controlPoint2: CGPoint(x: 8, y: 14))
}
```

```

// segment 9: line
path.lineTo(CGPoint(x: 8, y: 26))

// *****
// **** Bottom side ****
// *****

// segment 10: line
path.closePath() // draws the final line to close the path

return path
}

```

Hinweis: Einige der obigen Codes können durch Hinzufügen einer Linie und eines Bogens in einem einzelnen Befehl reduziert werden (da der Bogen einen implizierten Startpunkt hat). Sehen Sie [hier](#) für weitere Details.

Zeichne den Pfad

Wir können den Pfad entweder in einer Ebene oder in `drawRect` .

Methode 1: Zeichnen Sie den Pfad in einer Ebene

Unsere benutzerdefinierte Klasse sieht so aus. Wir fügen unseren Bezier-Pfad zu einem neuen `CAShapeLayer` wenn die Ansicht initialisiert wird.

```

import UIKit
class MyCustomView: UIView {

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    func setup() {

        // Create a CAShapeLayer
        let shapeLayer = CAShapeLayer()

        // The Bezier path that we made needs to be converted to
        // a CGPath before it can be used on a layer.
        shapeLayer.path = createBezierPath().CGPath

        // apply other properties related to the path
        shapeLayer.strokeColor = UIColor.blueColor().CGColor
        shapeLayer.fillColor = UIColor.whiteColor().CGColor
        shapeLayer.lineWidth = 1.0
        shapeLayer.position = CGPoint(x: 10, y: 10)

        // add the new layer to our custom view
        self.layer.addSublayer(shapeLayer)
    }
}

```

```

}

func createBezierPath() -> UIBezierPath {

    // see previous code for creating the Bezier path
}
}

```

Und so erstellen Sie unsere Ansicht im View Controller

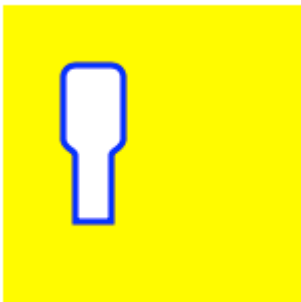
```

override func viewDidLoad() {
    super.viewDidLoad()

    // create a new UIView and add it to the view controller
    let myView = MyCustomView()
    myView.frame = CGRect(x: 100, y: 100, width: 50, height: 50)
    myView.backgroundColor = UIColor.yellowColor()
    view.addSubview(myView)
}

```

Wir bekommen...

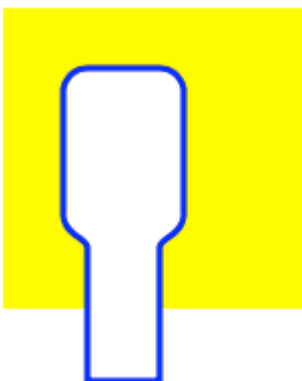


Hmm, das ist ein bisschen klein, weil ich alle Zahlen fest codiert habe. Ich kann die Pfadgröße jedoch wie folgt vergrößern:

```

let path = createBezierPath()
let scale = CGAffineTransformMakeScale(2, 2)
path.applyTransform(scale)
shapeLayer.path = path.CGPath

```



Methode 2: Zeichnen Sie den Pfad in `drawRect`

Die Verwendung von `drawRect` ist langsamer als das Zeichnen auf die Ebene. `drawRect` wird diese Methode nicht empfohlen, wenn Sie sie nicht benötigen.

Hier ist der überarbeitete Code für unsere benutzerdefinierte Ansicht:

```
import UIKit
class MyCustomView: UIView {

    override func drawRect(rect: CGRect) {

        // create path (see previous code)
        let path = createBezierPath()

        // fill
        let fillColor = UIColor.whiteColor()
        fillColor.setFill()

        // stroke
        path.lineWidth = 1.0
        let strokeColor = UIColor.blueColor()
        strokeColor.setStroke()

        // Move the path to a new location
        path.applyTransform(CGAffineTransformMakeTranslation(10, 10))

        // fill and stroke the path (always do these last)
        path.fill()
        path.stroke()

    }

    func createBezierPath() -> UIBezierPath {

        // see previous code for creating the Bezier path
    }

}
```

was uns das gleiche Ergebnis gibt ...



Weitere Studie

Hervorragende Artikel zum Verständnis der Bezierpfade.

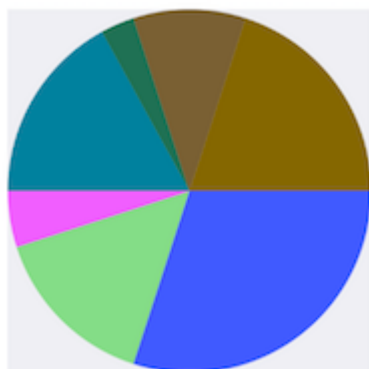
- [Denken wie ein Bézier-Pfad](#) (Alles, was ich je von diesem Autor gelesen habe, ist gut und die Inspiration für mein Beispiel oben stammt von hier.)
- [Kodierung Mathematik: Episode 19 - Bezierkurven](#) (unterhaltsame und gute visuelle Illustrationen)
- [Bezierkurven](#) (wie sie in Grafikanwendungen verwendet werden)
- [Bezierkurven](#) (gute Beschreibung der Herleitung der mathematischen Formeln)

Anmerkungen

- Dieses Beispiel stammt ursprünglich aus [dieser Stack Overflow-Antwort](#) .
- In Ihren aktuellen Projekten sollten Sie wahrscheinlich keine fest codierten Zahlen verwenden, sondern die Größenangaben aus den Grenzen Ihrer Ansicht ziehen.

Kreisansicht und Spaltenansicht mit UIBezierPath

- Kuchenansicht



```
- (void)drawRect:(CGRect)rect {

    NSArray *data = @[30, 15, 5, 17, 3, 10, 20];

    // 1. context
    CGContextRef cxtRef = UIGraphicsGetCurrentContext();

    CGPoint center = CGPointMake(150, 150);
    CGFloat radius = 150;
    __block CGFloat startAngle = 0;
    [data enumerateObjectsUsingBlock:^(NSNumber * _Nonnull obj, NSUInteger idx, BOOL *
    _Nonnull stop) {

        // 2. create path
        CGFloat endAngle = obj.floatValue / 100 * M_PI * 2 + startAngle;
        UIBezierPath *circlePath = [UIBezierPath bezierPathWithArcCenter:center radius:radius
startAngle:startAngle endAngle:endAngle clockwise:YES];
        [circlePath addLineToPoint:center];

        // 3. add path
        CGContextAddPath(cxtRef, circlePath.CGPath);

        // set color
        [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)
```

```

alpha:1.0] setFill];

    // 4. render
    CGContextDrawPath(cxtRef, kCGPathFill);

    // reset angle
    startAngle = endAngle;
  }];
}

```

```

override func draw(_ rect: CGRect) {
    // define data to create pie chart
    let data: [Int] = [30, 15, 5, 17, 3, 10, 20]

    // 1. find center of draw rect
    let center: CGPoint = CGPoint(x: rect.midX, y: rect.midY)

    // 2. calculate radius of pie
    let radius = min(rect.width, rect.height) / 2.0

    var startAngle: CGFloat = 0.0
    for value in data {

        // 3. calculate end angle for slice
        let endAngle = CGFloat(value) / 100.0 * CGFloat.pi * 2.0 + startAngle

        // 4. create UIBezierPath for slice
        let circlePath = UIBezierPath(arcCenter: center, radius: radius, startAngle: startAngle,
endAngle: endAngle, clockwise: true)

        // 5. add line to center to close path
        circlePath.addLine(to: center)

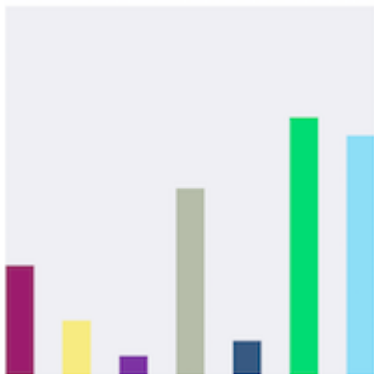
        // 6. set fill color for current slice
        UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

        // 7. fill slice path
        circlePath.fill()

        // 8. set end angle as start angle for next slice
        startAngle = endAngle
    }
}

```

- Spaltenansicht



```

- (void)drawRect:(CGRect)rect {

    NSArray *data = @[300, 150.65, 55.3, 507.7, 95.8, 700, 650.65];

    // 1.
    CGContextRef cxtRef = UIGraphicsGetCurrentContext();

    NSInteger columnCount = 7;
    CGFloat width = self.bounds.size.width / (columnCount + columnCount - 1);
    for (NSInteger i = 0; i < columnCount; i++) {

        // 2.
        CGFloat height = [data[i] floatValue] / 1000 * self.bounds.size.height; // floatValue
        CGFloat x = 0 + width * (2 * i);
        CGFloat y = self.bounds.size.height - height;
        UIBezierPath *rectPath = [UIBezierPath bezierPathWithRect:CGRectMake(x, y, width,
height)];
        CGContextAddPath(cxtRef, rectPath.CGPath);

        // 3.
        [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)
alpha:1.0] setFill];
        CGContextDrawPath(cxtRef, kCGPathFill);
    }
}

```

```

override func draw(_ rect: CGRect) {
    // define data for chart
    let data: [CGFloat] = [300, 150.65, 55.3, 507.7, 95.8, 700, 650.65]

    // 1. calculate number of columns
    let columnCount = data.count

    // 2. calculate column width
    let columnWidth = rect.width / CGFloat(columnCount + columnCount - 1)

    for (columnIndex, value) in data.enumerated() {
        // 3. calculate column height
        let columnHeight = value / 1000.0 * rect.height

        // 4. calculate column origin
        let columnOrigin = CGPoint(x: (columnWidth * 2.0 * CGFloat(columnIndex)), y:
(rect.height - columnHeight))

        // 5. create path for column
        let columnPath = UIBezierPath(rect: CGRect(origin: columnOrigin, size: CGSize(width:
columnWidth, height: columnHeight)))

        // 6. set fill color for current column
        UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

        // 7. fill column path
        columnPath.fill()
    }
}

```

UIBezierPath online lesen: <https://riptutorial.com/de/ios/topic/3186/uiBezierPath>

Kapitel 157: UIButton

Einführung

UIButton : **UIControl** fängt Berührungseignisse ab und sendet **beim Tippen** eine Aktionsnachricht an ein Zielobjekt. Sie können den Titel, das Bild und andere Darstellungseigenschaften einer Schaltfläche festlegen. Darüber hinaus können Sie für jeden Schaltflächenstatus ein anderes Erscheinungsbild angeben.

Bemerkungen

Schaltflächentypen

Der Typ einer Schaltfläche bestimmt das grundlegende Erscheinungsbild und Verhalten. Nachdem Sie eine Schaltfläche erstellt haben, können Sie ihren Typ nicht ändern. Die am häufigsten verwendeten Schaltflächentypen sind "Benutzerdefiniert" und "Systemtyp", verwenden Sie jedoch gegebenenfalls die anderen Typen

- UIButtonTypeCustom

```
No button style.
```

- UIButtonTypeSystem

```
A system style button, such as those shown in navigation bars and toolbars.
```

- UIButtonTypeDetailDisclosure

```
A detail disclosure button.
```

- UIButtonTypeInfoLight

```
An information button that has a light background.
```

- UIButtonTypeInfoDark

```
An information button that has a dark background.
```

- UIButtonTypeContactAdd

```
A contact add button.
```

Beim Erstellen einer benutzerdefinierten Schaltfläche - das ist eine Schaltfläche mit dem

benutzerdefinierten Typ - wird der Rahmen der Schaltfläche anfänglich auf (0, 0, 0, 0) gesetzt. Bevor Sie die Schaltfläche zu Ihrer Benutzeroberfläche hinzufügen, sollten Sie den Frame auf einen angemesseneren Wert aktualisieren.

Examples

Erstellen eines UIButton

UIButtons können in einem Frame initialisiert werden:

Schnell

```
let button = UIButton(frame: CGRect(x: x, y: y, width: width, height: height))
```

Ziel c

```
UIButton *button = [[UIButton alloc] initWithFrame:CGRectMake(x, y, width, height)];
```

Ein bestimmter UIButton-Typ kann wie folgt erstellt werden:

Schnell

```
let button = UIButton(type: .Custom)
```

Ziel c

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

wobei type ein UIButtonType :

```
enum UIButtonType : Int {
    case Custom
    case System
    case DetailDisclosure
    case InfoLight
    case InfoDark
    case ContactAdd
    static var RoundedRectangle: UIButtonType { get }
}
```

Titel einstellen

Schnell

```
button.setTitle(titleString, forState: controlState)
```

Ziel c

```
[button setTitle:(NSString *) forState:(UIControlState)];
```

So legen Sie den Standardtitel auf "Hallo, Welt!"

Schnell

```
button.setTitle("Hello, World!", forState: .normal)
```

Ziel c

```
[button setTitle:@"Hello, World!" forState:UIControlStateNormal];
```

Titelfarbe einstellen

```
//Swift
button.setTitleColor(color, forState: controlState)

//Objective-C
[button setTitleColor:(nullable UIColor *) forState:(UIControlState)];
```

Um die Titelfarbe auf Blau zu setzen

```
//Swift
button.setTitleColor(.blue, for: .normal)

//Objective-C
[button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal]
```

Inhalte horizontal ausrichten

Schnell

```
//Align contents to the left of the frame
button.contentHorizontalAlignment = .left

//Align contents to the right of the frame
button.contentHorizontalAlignment = .right

//Align contents to the center of the frame
button.contentHorizontalAlignment = .center

//Make contents fill the frame
button.contentHorizontalAlignment = .fill
```

Ziel c

```
//Align contents to the left
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentLeft;

//Align contents to the right
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentRight;
```

```
//Align contents to the center
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentCenter;

//Align contents to fill the frame
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentFill;
```

Titelbezeichnung abrufen

Das darunterliegende Titeletikett kann, sofern vorhanden, mit abgerufen werden

Schnell

```
var label: UILabel? = button.titleLabel
```

Ziel c

```
UILabel *label = button.titleLabel;
```

Hiermit kann beispielsweise die Schriftart der Titelbezeichnung festgelegt werden

Schnell

```
button.titleLabel?.font = UIFont.boldSystemFontOfSize(12)
```

Ziel c

```
button.titleLabel.font = [UIFont boldSystemFontOfSize:12];
```

Deaktivieren eines UIButton

Eine Schaltfläche kann mit deaktiviert werden

Schnell

```
myButton.isEnabled = false
```

Ziel c:

```
myButton.enabled = NO;
```

Die Schaltfläche wird grau dargestellt:

Button

Wenn Sie nicht möchten, dass sich die Schaltfläche beim `adjustsImageWhenDisabled` der Schaltfläche ändert, setzen Sie `adjustsImageWhenDisabled` auf `false / NO`

Hinzufügen einer Aktion zu einem UIButton über Code (programmgesteuert)

Um einer Schaltfläche eine Methode hinzuzufügen, erstellen Sie zuerst eine Aktionsmethode:

Ziel c

```
-(void)someButtonAction:(id)sender {
    // sender is the object that was tapped, in this case its the button.
    NSLog(@"Button is tapped");
}
```

Schnell

```
func someButtonAction() {
    print("Button is tapped")
}
```

Um diese Aktionsmethode jetzt zu Ihrer Schaltfläche hinzuzufügen, müssen Sie die folgende Codezeile schreiben:

Ziel c

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Schnell

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.TouchUpInside)
```

Für den Parameter `ControlEvents` sind alle Member von `ENUM UIControlEvents` gültig.

Schriftart einstellen

Schnell

```
myButton.titleLabel?.font = UIFont(name: "YourFontName", size: 20)
```

Ziel c

```
myButton.titleLabel.font = [UIFont fontWithName:@"YourFontName" size:20];
```

Anhängen einer Methode an eine Schaltfläche

Um einer Schaltfläche eine Methode hinzuzufügen, erstellen Sie zuerst eine Aktionsmethode:

Ziel c

```
-(void) someButtonAction{
    NSLog(@"Button is tapped");
}
```

Schnell

```
func someButtonAction() {
    print("Button is tapped")
}
```

Um diese Aktionsmethode jetzt zu Ihrer Schaltfläche hinzuzufügen, müssen Sie die folgende Codezeile schreiben:

Ziel c

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Schnell

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.touchUpInside)
```

Für ControlEvents sind alle Mitglieder von `ENUM UIControlEvents` gültig.

Ermitteln Sie die Größe von UIButton streng nach Text und Schriftart

Verwenden Sie die Funktion `intrinsicContentSize` um die genaue Größe eines UIButton-Textes basierend auf seiner Schriftart zu ermitteln.

Schnell

```
button.intrinsicContentSize.width
```

Ziel c

```
button.intrinsicContentSize.width;
```

Bild einstellen

Schnell

```
button.setImage(UIImage(named:"test-image"), forState: .normal)
```

Ziel c

```
[self.button setImage:[UIImage imageNamed:@"test-image"] forState:UIControlStateNormal];
```

Mehrere Kontrollzustände

Sie können auch ein Bild für mehrere `UIControlStates` , um beispielsweise dasselbe Bild für den Status " Selected und " Highlighted `UIControlStates` :

Schnell

```
button.setImage(UIImage(named:"test-image"), forState:[.selected, .highlighted])
```

Ziel c

```
[self.button setImage:[UIImage imageNamed:@"test-image"]  
forState:UIControlStateSelected|:UIControlStateHighlighted];
```

UIButton online lesen: <https://riptutorial.com/de/ios/topic/516/uibutton>

Kapitel 158: UICollectionView

Examples

Erstellen Sie eine Sammlungsansicht programmgesteuert

Schnell

```
func createCollectionView() {
    let layout: UICollectionViewFlowLayout = UICollectionViewFlowLayout()
    let collectionView = UICollectionView(frame: CGRect(x: 0, y: 0, width: view.frame.width,
height: view.frame.height), collectionViewLayout: layout)
    collectionView.dataSource = self
    collectionView.delegate = self
    view.addSubview(collectionView)
}
```

Ziel c

```
- (void)createCollectionView {
    UICollectionViewFlowLayout *layout = [[UICollectionViewFlowLayout alloc] init];
    UICollectionView *collectionView = [[UICollectionView alloc] initWithFrame:CGRectMake(0,
0, self.view.frame.size.width, self.view.frame.size.height) collectionViewLayout:layout];
    [collectionView setDataSource:self];
    [collectionView setDelegate:self];
    [self.view addSubview:collectionView];
}
```

Swift - UICollectionViewDelegateFlowLayout

```
// MARK: - UICollectionViewDelegateFlowLayout
extension ViewController: UICollectionViewDelegateFlowLayout {
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAtIndexPath indexPath: NSIndexPath) -> CGSize {
        return CGSize(width: 50, height: 50)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, insetForSectionAtIndex section: Int) -> UIEdgeInsets {
        return UIEdgeInsets(top: 5, left: 5, bottom: 5, right: 5)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
}
```

Erstellen Sie eine UICollectionView

Initialisieren Sie eine `UICollectionView` mit einem `CGRect` Frame:

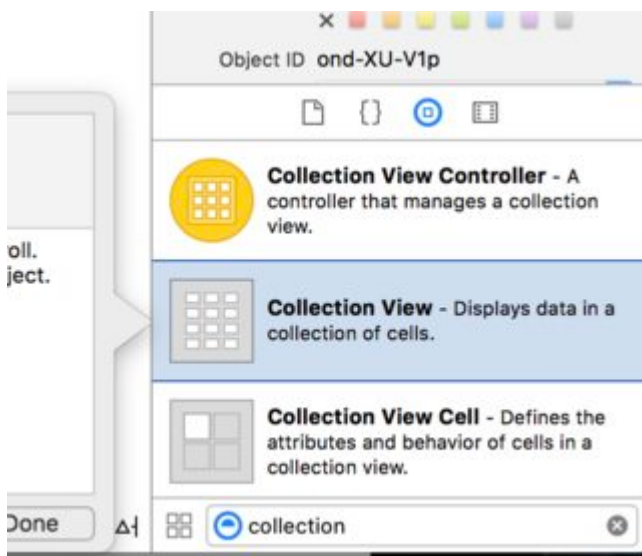
Schnell:

```
let collection = UICollectionView(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Ziel c:

```
UICollectionView *collection = [[UICollectionView alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

Sie können auch eine `UICollectionView` im Interface Builder erstellen



UICollectionView - Datenquelle

Jede Sammlungsansicht muss ein `DataSource` . Das `DataSource` ist der Inhalt, den Ihre App in der `UICollectionView` . Alle `DataSource` müssen `collectionView:cellForItemAtIndexPath:` Methoden `collectionView:numberOfItemsInSection:` und `collectionView:cellForItemAtIndexPath:` .

Erforderliche Methoden

Schnell

```
func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    // Return how many items in section
    let sectionArray = _data[section]
    return sectionArray.count
}

func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath: NSIndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCellWithReuseIdentifier(MyCellID)
    // If you use a custom cell class then cast the cell returned, like:
    // as! MyCollectionViewCellClass
    // or you will have errors when you try to use features of that class.
```

```

//Customize your cell here, default UICollectionViewCell do not contain any inherent
//text or image views (like UITableView), but some could be added,
//or a custom UICollectionViewCell sub-class could be used
return cell
}

```

Ziel c

```

- (NSInteger)collectionView:(UICollectionView*)collectionView
numberOfItemsInSection:(NSInteger)section {
    // Return how many items in section
    NSArray *sectionArray = [_data objectAtIndex:section];
    return [sectionArray count];
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
cellForItemAtIndexPath:(NSIndexPath *)indexPath {
    // Return a cell
    UICollectionViewCell *newCell = [self.collectionView
                                     dequeueReusableCellWithReuseIdentifier:MyCellID
                                     forIndexPath:indexPath];

    //Customize your cell here, default UICollectionViewCell do not contain any inherent
    //text or image views (like UITableView), but some could be added,
    //or a custom UICollectionViewCell sub-class could be used
    return newCell;
}

```

Einfaches Swift-Beispiel für eine Sammlungsansicht

Erstellen Sie ein neues Projekt

Es kann sich nur um eine Single View-Anwendung handeln.

Fügen Sie den Code hinzu

Erstellen Sie eine neue Cocoa Touch Class-Datei (Datei> Neu> Datei ...> iOS> Cocoa Touch Class). MyCollectionViewCell es MyCollectionViewCell . Diese Klasse enthält die Auslässe für die Ansichten, die Sie Ihrer Zelle im Storyboard hinzufügen.

```

import UIKit
class MyCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var myLabel: UILabel!
}

```

Wir werden diese Steckdose später anschließen.

Öffnen Sie ViewController.swift und stellen Sie sicher, dass Sie über den folgenden Inhalt verfügen:

```

import UIKit
class ViewController: UIViewController, UICollectionViewDataSource, UICollectionViewDelegate {

    let reuseIdentifier = "cell" // also enter this string as the cell identifier in the
    storyboard
    var items = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14",
"15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44",
"45", "46", "47", "48"]

    // MARK: - UICollectionViewDataSource protocol

    // tell the collection view how many cells to make
    func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int)
-> Int {
        return self.items.count
    }

    // make a cell for each cell index path
    func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath:
NSIndexPath) -> UICollectionViewCell {

        // get a reference to our storyboard cell
        let cell = collectionView.dequeueReusableCellWithReuseIdentifier(reuseIdentifier,
forIndexPath: indexPath) as! MyCollectionViewCell

        // Use the outlet in our custom class to get a reference to the UILabel in the cell
        cell.myLabel.text = self.items[indexPath.item]
        cell.backgroundColor = UIColor.yellowColor() // make cell more visible in our example
project

        return cell
    }

    // MARK: - UICollectionViewDelegate protocol

    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath) {
        // handle tap events
        print("You selected cell #\(indexPath.item)!")
    }
}

```

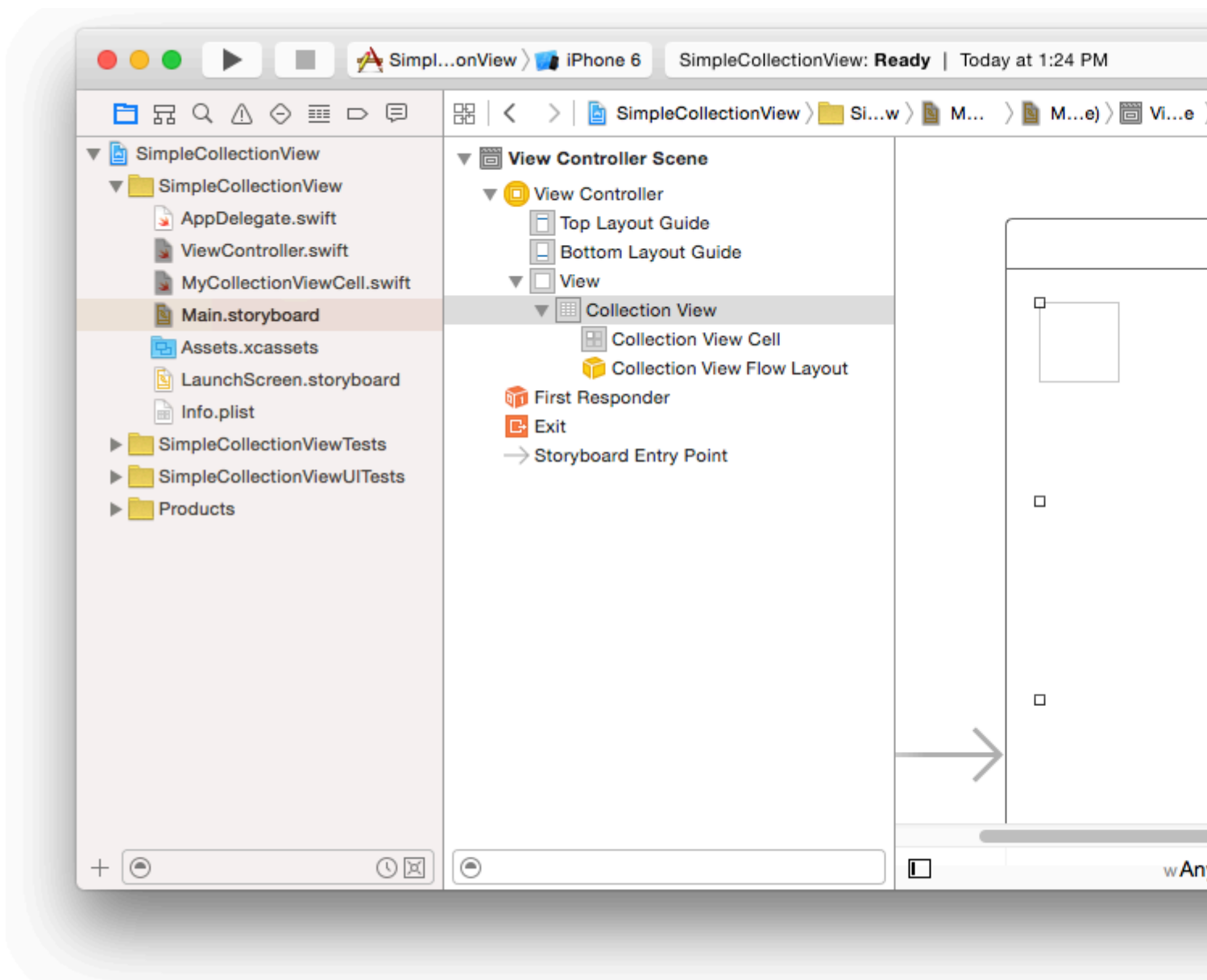
Anmerkungen

- UICollectionViewDataSource und UICollectionViewDelegate sind die Protokolle, denen die Sammlungsansicht folgt. Sie können auch das Protokoll UICollectionViewDelegateFlowLayout hinzufügen, um die Größe der Ansichten programmgesteuert zu ändern. UICollectionViewDelegateFlowLayout ist jedoch nicht erforderlich.
- Wir fügen nur einfache Zeichenfolgen in unser Raster ein, aber Sie könnten später auch Bilder erstellen.

Richten Sie das Storyboard ein

Ziehen Sie eine Sammlungsansicht in den View Controller in Ihrem Storyboard. Sie können

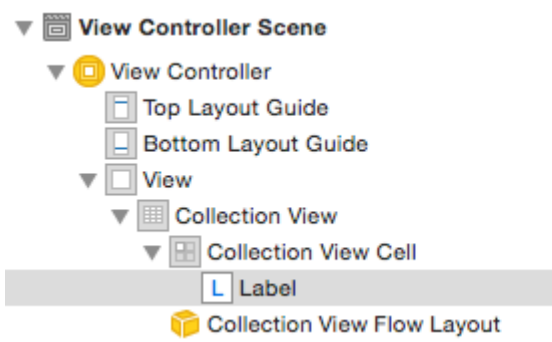
Einschränkungen hinzufügen, um die übergeordnete Ansicht zu füllen, wenn Sie möchten.



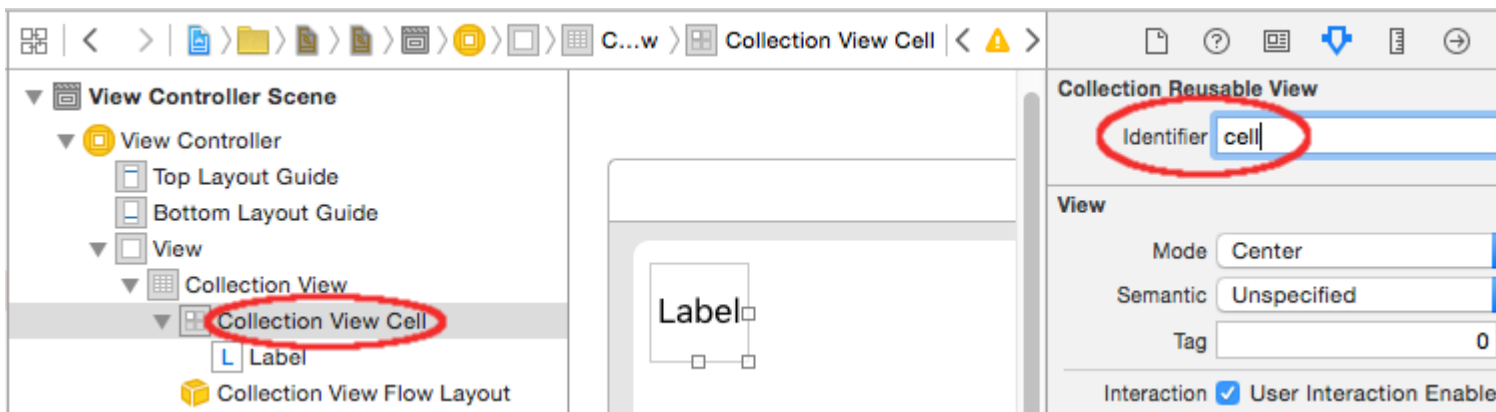
Vergewissern Sie sich, dass Ihre Standardwerte im Attribut-Inspector ebenfalls vorhanden sind

- Artikel: 1
- Layout: Fluss

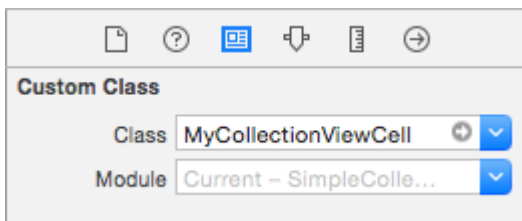
Die kleine Box oben links in der Sammlungsansicht ist eine Sammlungsansichtzelle. Wir werden es als unsere Prototypzelle verwenden. Ziehen Sie eine Beschriftung in die Zelle und zentrieren Sie sie. Sie können die Größe der Zellenränder ändern und Einschränkungen hinzufügen, um die Beschriftung zu zentrieren, wenn Sie möchten.



Schreiben Sie "Zelle" (ohne Anführungszeichen) in das Feld "Kennung" des Attribut-Inspektors für die Sammlungsansicht-Zelle. Beachten Sie, dass dies der gleiche Wert ist wie `let reuseIdentifier = "cell"` in `ViewController.swift`.

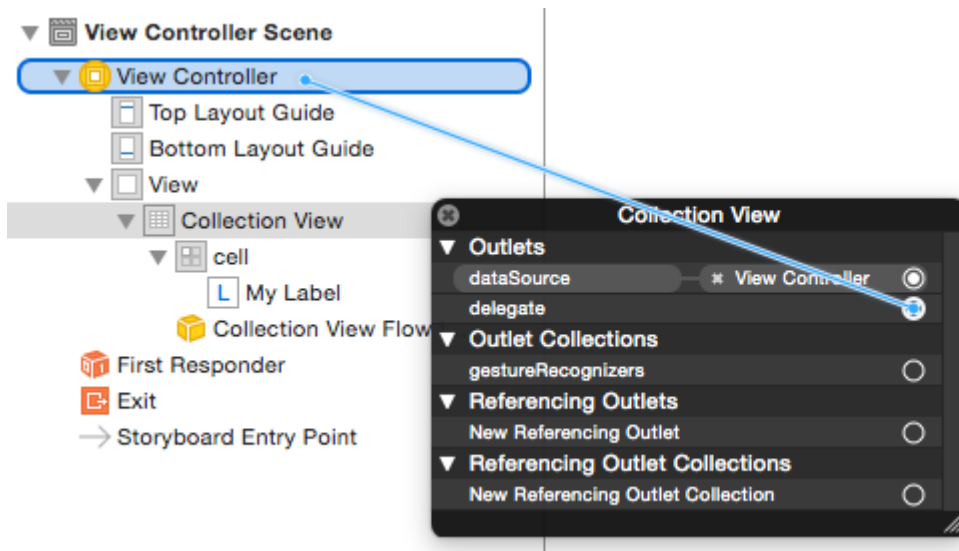


`MyCollectionViewCell` im Identitätsinspektor für die Zelle den Klassennamen `MyCollectionViewCell`, unserer benutzerdefinierten Klasse, die wir erstellt haben.



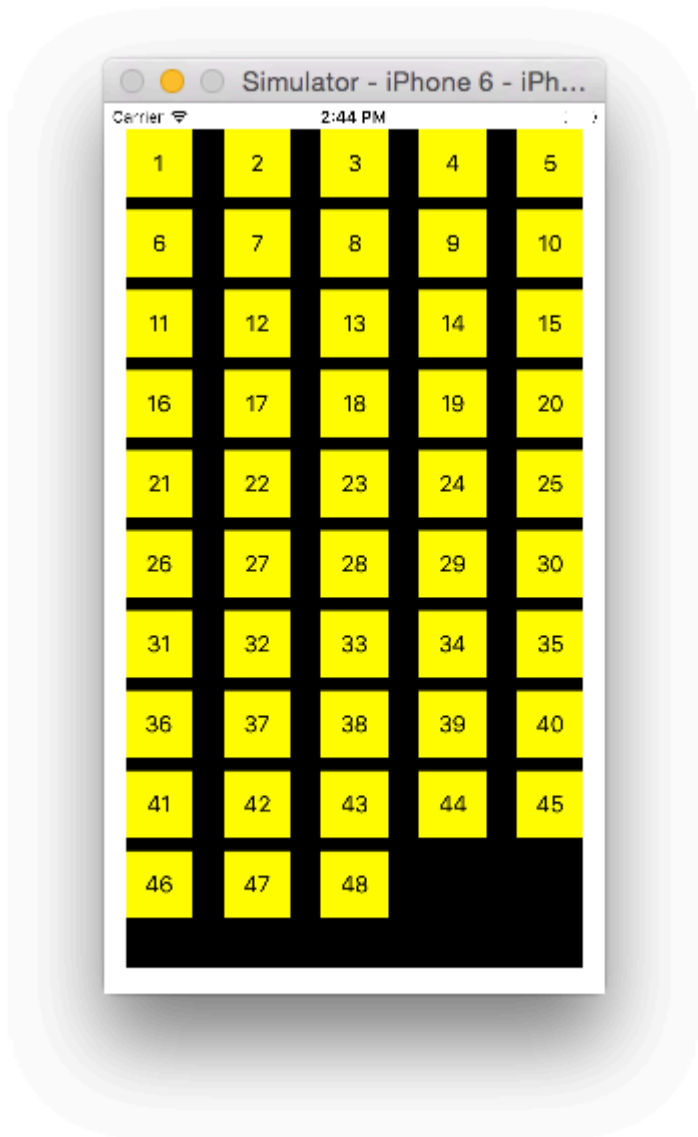
Schließen Sie die Auslässe an

- Haken Sie das Label in der Auflistungszelle an `myLabel` in der `MyCollectionViewCell` Klasse. (Sie können mit [gedrückter Maustaste ziehen](#).)
- `dataSource` den Collection View- `delegate` und die `dataSource` an den View Controller. (Klicken Sie in der Dokumentgliederung mit der rechten Maustaste auf die Sammlungsansicht. Klicken Sie dann auf den Pluspfeil und ziehen Sie ihn zum View Controller.)



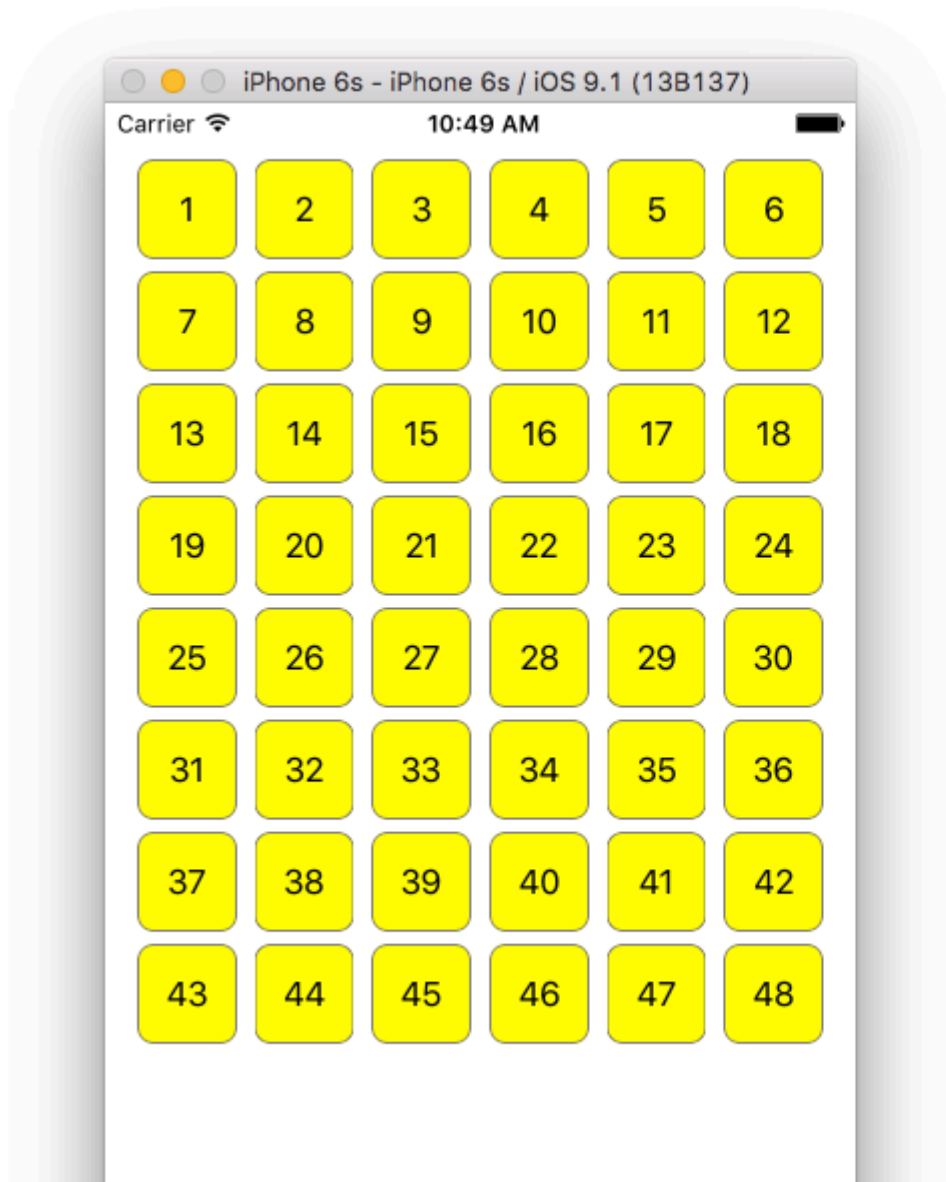
Fertig

Nach dem Hinzufügen von Einschränkungen zum Zentrieren der Beschriftung in der Zelle und Anheften der Sammlungsansicht an die Wände des übergeordneten Elements sieht es folgendermaßen aus.



Verbesserungen vornehmen

Wenn Sie das Erscheinungsbild verbessern möchten, lesen [Sie den ursprünglichen Beitrag, aus dem dieses Beispiel stammt](#) .



Weitere Studie

- [Ein einfaches UICollectionView-Tutorial](#)
- [UICollectionView-Lernprogramm Teil 1: Erste Schritte](#)
- [UICollectionView-Lernprogramm Teil 2: Wiederverwendbare Ansichten und Zellenauswahl](#)

Stapelaktualisierungen durchführen

Sie können komplexe Änderungen an Ihrer Sammlungsansicht mit der Methode `performBatchUpdates` animieren. Innerhalb des Aktualisierungsblocks können Sie mehrere Modifikationen angeben, um sie alle gleichzeitig zu animieren.

```
collectionView.performBatchUpdates({  
    // Perform updates  
}, nil)
```

Innerhalb des Aktualisierungsblocks können Sie Einfügungen, Löschungen, Verschiebungen und

Neuladen durchführen. So bestimmen Sie den zu verwendenden indexPath:

Art	NSIndexPath
Einfügung	Index in neuem Array
Streichung	Index im alten Array
Bewegung	from: altes Array bis: neues Array
Neu laden	entweder neues oder altes Array (es sollte keine Rolle spielen)

Sie sollten reload nur für Zellen aufrufen, die nicht verschoben wurden, deren Inhalt sich jedoch geändert hat. Es ist wichtig zu wissen, dass eine Verschiebung den Inhalt einer Zelle nicht aktualisiert, sondern nur deren Position verschiebt.

Um sicherzustellen, dass Ihre Batch - Aktualisierung korrekt durchgeführt werden, stellen Sie sicher, dass die Menge der indexPaths für deletion, move-from, und reload zu insertion move-to reload reload sind einzigartig, und der Satz von indexPaths für insertion, move-to und reload sind einzigartig.

Hier ist ein Beispiel für ein korrektes Stapel-Update:

```
let from = [1, 2, 3, 4, 5]
let to = [1, 3, 6, 4, 5]

collectionView.performBatchUpdates({
    collectionView.insertItemsAtIndexPaths([NSIndexPath(forItem: 2, inSection: 0)])
    collectionView.deleteItemsAtIndexPaths([NSIndexPath(forItem: 1, inSection: 0)])
    collectionView.moveItemAtIndexPath(NSIndexPath(forItem: 2, inSection: 0),
                                        toIndexPath: NSIndexPath(forItem: 1, inSection: 0))
}, nil)
```

UICollectionViewDelegate-Setup und Elementauswahl

Wenn eine Aktion an die UICollectionViewDelegate einer Sammlungsansicht

UICollectionViewDelegate, müssen Sie das Protokoll UICollectionViewDelegate implementieren.

UIViewController MyViewController, die Sammlungsansicht befindet sich in einem UIViewController MyViewController.

Ziel c

In Ihrem MyViewController.h wird erklärt, dass das UICollectionViewDelegate Protokoll wie UICollectionViewDelegate implementiert wird

```
@interface MyViewController : UIViewController <UICollectionViewDelegate, .../* previous existing delegate, as UICollectionViewDataSource */>
```

Schnell

Fügen Sie in Ihrer *MyViewController.swift* Folgendes hinzu

```
class MyViewController : UICollectionViewDelegate {  
}
```

Die Methode, die implementiert werden muss, ist

Ziel c

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath *)indexPath  
{  
}
```

Schnell

```
func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath: NSIndexPath)  
{  
}
```

Als Beispiel können wir die Hintergrundfarbe der ausgewählten Zelle auf Grün setzen.

Ziel c

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath *)indexPath  
{  
    UICollectionViewCell* cell = [collectionView cellForItemAtIndexPath:indexPath];  
    cell.backgroundColor = [UIColor greenColor];  
}
```

Schnell

```
class MyViewController : UICollectionViewDelegate {  
    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath: NSIndexPath)  
    {  
        var cell : UICollectionViewCell = collectionView.cellForItemAtIndexPath(indexPath)!  
        cell.backgroundColor = UIColor.greenColor()  
    }  
}
```

Verwalten Sie die Ansicht für mehrere Sammlungen mit DataSource und Flowlayout

Hier verwalten wir mehrere Delegierungsmethoden mit didSelect-Ereignissen.

```

extension ProductsVC: UICollectionViewDelegate, UICollectionViewDataSource{

    // MARK: - UICollectionViewDataSource
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
        guard collectionView == collectionCategory else {
            return arrOfProducts.count
        }
        return arrOfCategory.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {

        guard collectionView == collectionProduct else {
            let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"ProductCategoryCell", for: indexPath) as! ProductCategoryCell
            cell.viewBackground.layer.borderWidth = 0.5
            //Do some thing as per use
            return cell
        }

        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellIdentifier,
for: indexPath) as! ProductCell
        cell.contentView.layer.borderWidth = 0.5
        cell.contentView.layer.borderColor = UIColor.black.cgColor
        let json = arrOfProducts[indexPath.row]
        //Do something as per use

        return cell
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath) {
        guard collectionView == collectionCategory else {
            let json = arrOfProducts[indexPath.row]
            // Do something for collectionProduct here
            return
        }
        let json = arrOfCategory[indexPath.row] as [String: AnyObject]
        let id = json["cId"] as? String ?? ""
        // Do something
    }
}

extension ProductsVC: UICollectionViewDelegateFlowLayout{

    // MARK: - UICollectionViewDelegateFlowLayout
    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

        let collectionWidth = collectionView.bounds.width
        guard collectionView == collectionProduct else {
            var itemWidth = collectionWidth / 4 - 1;

            if(UI_USER_INTERFACE_IDIOM() == .pad) {
                itemWidth = collectionWidth / 4 - 1;
            }
            return CGSize(width: itemWidth, height: 50)
        }
    }
}

```

```

var itemWidth = collectionViewWidth / 2 - 1;
if(UI_USER_INTERFACE_IDIOM() == .pad) {
    itemWidth = collectionViewWidth / 4 - 1;
}
return CGSize(width: itemWidth, height: 250);
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {
    return 1
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {
    return 1
}
}

```

23-01-2017	24-01-2017	25-01-2017	
11:00	11:15	11:30	11:45
12:00	12:15	12:30	12:45
13:00	13:15	13:30	13:45
14:00	14:15	14:30	14:45
15:00	15:15	15:30	15:45
16:00	16:15	16:30	16:45

UICollectionView online lesen: <https://riptutorial.com/de/ios/topic/2399/uicollectionview>

Kapitel 159: UIColor

Examples

Erstellen eines UIColor

Es gibt viele Möglichkeiten, eine `UIColor` erstellen:

Schnell

- Verwenden einer der vordefinierten Farben:

```
let redColor = UIColor.redColor()
let blueColor: UIColor = .blueColor()

// In Swift 3, the "Color()" suffix is removed:
let redColor = UIColor.red
let blueColor: UIColor = .blue
```

Wenn der Compiler bereits weiß, dass es sich bei der Variablen um eine Instanz von `UIColor`, können Sie den gesamten Typ überspringen:

```
let view = UIView()
view.backgroundColor = .yellowColor()
```

- Verwenden des Graustufenwerts und des Alphas:

```
let grayscaleColor = UIColor(white: 0.5, alpha: 1.0)
```

- Farbton, Sättigung, Helligkeit und Alpha verwenden:

```
let hsbColor = UIColor(
    hue: 0.4,
    saturation: 0.3,
    brightness: 0.7,
    alpha: 1.0
)
```

- Verwenden der RGBA-Werte:

```
let rgbColor = UIColor(
    red: 30.0 / 255,
    green: 70.0 / 255,
    blue: 200.0 / 255,
    alpha: 1.0
)
```

- Verwenden eines Musterbildes:


```
let patternColor = UIColor(patternImage: UIImage(named: "myImage")!)
```

Ziel c

- Verwenden einer der vordefinierten Farben:

```
UIColor *redColor = [UIColor redColor];
```

- Verwenden des Graustufenwerts und des Alphas:

```
UIColor *grayscaleColor = [UIColor colorWithWhite: 0.5 alpha: 1.0];
```

- Farbton, Sättigung, Helligkeit und Alpha verwenden:

```
UIColor *hsbColor = [UIColor  
    colorWithHue: 0.4  
    saturation: 0.3  
    brightness: 0.7  
    alpha: 1.0  
];
```

- Verwenden der RGBA-Werte:

```
UIColor *rgbColor = [UIColor  
    colorWithRed: 30.0 / 255.0  
    green: 70.0 / 255.0  
    blue: 200.0 / 255.0  
    alpha: 1.0  
];
```

- Verwenden eines Musterbildes:

```
UIColor *pattenColor = [UIColor colorWithPatternImage:[UIImage  
    imageNamed:@"myImage.png"]];
```

Undokumentierte Methoden

Es gibt eine Vielzahl von undokumentierten Methoden für `UIColor` die alternative Farben oder Funktionen `UIColor`. Diese finden Sie in der [privaten Header-Datei von UIColor](#). Ich werde die Verwendung von zwei privaten Methoden dokumentieren, `styleString()` und

`_systemDestructiveTintColor()`.

`styleString`

Seit iOS 2.0 gibt es auf `UIColor` eine private `UIColor` namens `styleString` die eine RGB- oder RGBA-Zeichenfolgendarstellung der Farbe `whiteColor`, selbst für Farben wie `whiteColor` außerhalb des RGB- `whiteColor`.

Ziel c:

```

@interface UIColor (Private)

- (NSString *)styleString;

@end

// ...

[[UIColor whiteColor] styleString]; // rgb(255,255,255)
[[UIColor redColor] styleString]; // rgb(255,0,0)
[[UIColor lightTextColor] styleString]; // rgba(255,255,255,0.600000)

```

In Swift können Sie einen Überbrückungsheader verwenden, um die Schnittstelle anzuzeigen. Mit pure Swift müssen Sie ein @objc Protokoll mit der privaten Methode und unsafeBitCast UIColor mit dem Protokoll UIColor :

```

@objc protocol UIColorPrivate {
    func styleString() -> String
}

let white = UIColor.whiteColor()
let red = UIColor.redColor()
let lightTextColor = UIColor.lightTextColor()

let whitePrivate = unsafeBitCast(white, UIColorPrivate.self)
let redPrivate = unsafeBitCast(red, UIColorPrivate.self)
let lightTextColorPrivate = unsafeBitCast(lightTextColor, UIColorPrivate.self)

whitePrivate.styleString() // rgb(255,255,255)
redPrivate.styleString() // rgb(255,0,0)
lightTextColorPrivate.styleString() // rgba(255,255,255,0.600000)

```

_systemDestructiveTintColor()

Es gibt eine undokumentierte Klassenmethode für UIColor Namen _systemDestructiveTintColor die die rote Farbe _systemDestructiveTintColor die von den Schaltflächen des destruktiven Systems verwendet wird:

```

let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()

```

Es gibt ein nicht verwaltetes Objekt zurück, für das Sie .takeUnretainedValue() aufrufen .takeUnretainedValue() , da der .takeUnretainedValue() nicht auf unser eigenes Objekt übertragen wurde.

Wie bei jeder undokumentierten API sollten Sie beim Verwenden dieser Methode Vorsicht walten lassen:

```

if UIColor.respondsToSelector("_systemDestructiveTintColor") {
    if let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
as? UIColor {
        // use the color
    }
}

```

oder mit einem Protokoll:

```
@objc protocol UIColorPrivateStatic {
    func _systemDestructiveTintColor() -> UIColor
}

let privateClass = UIColor.self as! UIColorPrivateStatic
privateClass._systemDestructiveTintColor() // UIDeviceRGBColorSpace 1 0.231373 0.188235 1
```

Farbe mit Alpha-Komponente

Sie können die Deckkraft auf eine bestimmte `UIColor` ohne mithilfe des Initialisierers `init(red:_, green:_, blue:_, alpha:_)` eine neue erstellen zu müssen.

Schnell

```
let colorWithAlpha = UIColor.redColor().colorWithAlphaComponent(0.1)
```

Swift 3

```
//In Swift Latest Version
_ colorWithAlpha = UIColor.red.withAlphaComponent(0.1)
```

Ziel c

```
UIColor * colorWithAlpha = [[UIColor redColor] colorWithAlphaComponent:0.1];
```

Benutzerdefinierte Attribute auf den `CGColor`-Datentyp anwenden lassen

Standardmäßig akzeptiert Interface Builder den `CGColor` Datentyp nicht, um das Hinzufügen von `CGColor` mit benutzerdefinierten Attributen im Interface Builder zu ermöglichen. Vielleicht möchten Sie eine Erweiterung wie diese verwenden:

Schnelle Erweiterung:

```
extension CALayer {
    func borderUIColor() -> UIColor? {
        return borderColor != nil ? UIColor(CGColor: borderColor!) : nil
    }

    func setBorderUIColor(color: UIColor) {
        borderColor = color.CGColor
    }
}
```

Das neue benutzerdefinierte Attribut (`borderUIColor`) wird

problemlos erkannt und angewendet.

User Defined Runtime Attributes		
Key Path	Type	Value
layer.cornerRadius	Number	6.5
layer.borderWidth	Number	1
layer.clipsToBounds	Boolean	<input checked="" type="checkbox"/>
layer.borderColor	Color	<input type="color"/>

Erstellen eines UIColor aus Hexadezimalzahl oder Zeichenfolge

Sie können eine `UIColor` aus einer Hexadezimalzahl oder einem String erstellen, z. B. `0xff00cc`, `"#FFFFFF"`.

Schnell

Int Wert

```
extension UIColor {
  convenience init(hex: Int, alpha: CGFloat = 1.0) {
    let r = CGFloat((hex >> 16) & 0xff) / 255
    let g = CGFloat((hex >> 08) & 0xff) / 255
    let b = CGFloat((hex >> 00) & 0xff) / 255
    self.init(red: r, green: g, blue: b, alpha: alpha)
  }
}
```

Beispiel:

```
let color = UIColor(hex: 0xff00cc, alpha: 1.0)
```

Beachten Sie, dass für `alpha` der Standardwert `1.0` ist. Er kann daher wie folgt verwendet werden:

```
let color = UIColor(hex: 0xff00cc)
```

String-Wert

```
extension UIColor {
  convenience init(hexCode: String) {
    let hex =
hexCode.stringByTrimmingCharactersInSet(NSCharacterSet.alphanumericCharacterSet().invertedSet)
    var int = UInt32()
    NSScanner(string: hex).scanHexInt(&int)
    let a, r, g, b: UInt32

    switch hex.characters.count {
    case 3:
      (a, r, g, b) = (255, (int >> 8) * 17, (int >> 4 & 0xF) * 17, (int & 0xF) * 17)
    case 6:
      (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)
    case 8:

```

```

        (a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int >> 8 & 0xFF, int & 0xFF)
    default:
        (a, r, g, b) = (1, 1, 1, 0)
    }

    self.init(red: CGFloat(r) / 255, green: CGFloat(g) / 255, blue: CGFloat(b) / 255,
alpha: CGFloat(a) / 255)
    }
}

```

Verwendungsbeispiel:

Hex mit Alpha

```
let color = UIColor("#80FFFFFF")
```

Hex ohne Alpha (color Alpha entspricht 1,0)

```
let color = UIColor("#FFFFFF")
let color = UIColor("#FFF")
```

Ziel c

Int Wert

```

@interface UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha;
@end

@implementation UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha {
    return [UIColor colorWithRed:((CGFloat)((hex & 0xFF0000) >> 16))/255.0
                    green:((CGFloat)((hex & 0xFF00) >> 8))/255.0
                    blue:((CGFloat)(hex & 0xFF))/255.0
                    alpha:alpha];
}
@end

```

Beispiel:

```
UIColor *color = [UIColor colorWithHex:0xff00cc alpha:1.0];
```

String-Wert

```

- (UIColor*) hex:(NSString*)hexCode {

    NSString *noHashString = [hexCode stringByReplacingOccurrencesOfString:@"#"
withString:@""];
    NSScanner *scanner = [NSScanner scannerWithString:noHashString];
    [scanner setCharactersToBeSkipped:[NSCharacterSet symbolCharacterSet]];

    unsigned hex;
    if (![scanner scanHexInt:&hex]) return nil;
}

```

```

int a;
int r;
int g;
int b;

switch (noHashString.length) {
    case 3:
        a = 255;
        r = (hex >> 8) * 17;
        g = ((hex >> 4) & 0xF) * 17;
        b = ((hex >> 0) & 0xF) * 17;
        break;
    case 6:
        a = 255;
        r = (hex >> 16);
        g = (hex >> 8) & 0xFF;
        b = (hex) & 0xFF;
        break;
    case 8:
        a = (hex >> 24);
        r = (hex >> 16) & 0xFF;
        g = (hex >> 8) & 0xFF;
        b = (hex) & 0xFF;
        break;

    default:
        a = 255.0;
        r = 255.0;
        b = 255.0;
        g = 255.0;
        break;
}

return [UIColor colorWithRed:r / 255.0f green:g / 255.0f blue:b / 255.0f alpha:a / 255];
}

```

Verwendungsbeispiel:

Hex mit Alpha

```
UIColor* color = [self hex:@"#80FFFFFF"];
```

Hex ohne Alpha (color Alpha entspricht 1)

```
UIColor* color = [self hex:@"#FFFFFF"];
UIColor* color = [self hex:@"#FFF"];
```

Angepasste Helligkeit der Farbe von UIColor

Das folgende Codebeispiel gibt Ihnen eine angepasste Version dieser Farbe, wobei ein höherer Prozentsatz heller und ein niedrigerer Prozentsatz dunkler ist.

Ziel c

```
+ (UIColor *)adjustedColorForColor:(UIColor *)c : (double)percent
```

```

{
    if (percent < 0) percent = 0;

    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MAX(r * percent, 0.0)
                                green:MAX(g * percent, 0.0)
                                blue:MAX(b * percent, 0.0)
                                alpha:a];

    return nil;
}

```

Schnell

```

func adjustedColorForColor( c: UIColor, var percent: CGFloat) -> UIColor {
    if percent < 0 {
        percent = 0
    }

    var r,g,b,a: CGFloat
    r = 0.0
    g = 0.0
    b = 0.0
    a = 0.0

    if c.getRed(&r, green: &g, blue: &b, alpha: &a) {
        return UIColor(red: max(r * percent, 0.0), green: max(g * percent, 0.0), blue: max(b *
percent, 0.0), alpha: a)
    }

    return UIColor()
}

```

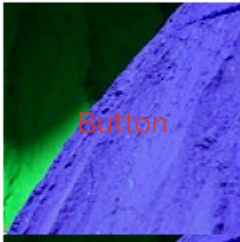
UIColor aus einem Bildmuster

Sie können ein `UIColor` Objekt mithilfe eines Bildmusters mithilfe der `UIColor(patternImage:_)` Methode `UIColor(patternImage:_)` .

```

btn.backgroundColor = UIColor(patternImage: UIImage(named: "image")!)

```



Hellerer und dunklerer Farbton einer bestimmten UIColor

Im folgenden Codebeispiel wird gezeigt, wie Sie eine hellere und dunklere Schattierung einer bestimmten Farbe erzielen können, die in Anwendungen mit dynamischen Designs nützlich ist

Für dunklere Farben

```
+ (UIColor *)darkerColorForColor:(UIColor *)c
{
    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MAX(r - 0.2, 0.0)
                                green:MAX(g - 0.2, 0.0)
                                blue:MAX(b - 0.2, 0.0)
                                alpha:a];
    return nil;
}
```

Für hellere Farben

```
+ (UIColor *)lighterColorForColor:(UIColor *)c
```



```
{
  CGFloat r, g, b, a;
  if ([c getRed:&r green:&g blue:&b alpha:&a])
    return [UIColor colorWithRed:MIN(r + 0.2, 1.0)
                          green:MIN(g + 0.2, 1.0)
                          blue:MIN(b + 0.2, 1.0)
                          alpha:a];
  return nil;
}
```

Siehe visuelle Unterschiede unten, unter Berücksichtigung der angegebenen Farbe ist `[UIColor orangeColor]`



UIColor online lesen: <https://riptutorial.com/de/ios/topic/956/UIColor>

Kapitel 160: UIControl - Ereignisbehandlung mit Blöcken

Examples

Einführung

Normalerweise fügen wir bei Verwendung von `UIControl` oder `UIButton` einen `selector` als `UIControl` `UIButton`, wenn ein Ereignis auf einer Schaltfläche oder einem Steuerelement auftritt, z.

Zum Beispiel würden wir Folgendes tun:

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(self.onButtonPress(_:)), for: .touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    func onButtonPress(_ button: UIButton!) {
        print("PRESSED")
    }
}
```

Wenn es um `selector`, muss der Compiler nur wissen, dass er existiert. Dies kann über ein `protocol` und nicht implementiert werden.

Zum Beispiel würde die folgende Anwendung Ihre Anwendung zum Absturz bringen:

```
import UIKit

@objc
protocol ButtonEvent {
    @objc optional func onButtonPress(_ button: UIButton)
}

class ViewController: UIViewController, ButtonEvent {
    @IBOutlet weak var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
    }
}
```

```

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(ButtonEvent.onButtonPress(_:)), for:
        .touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}

```

Dies liegt daran, dass Ihre Anwendung die `onButtonPress` Funktion NICHT implementiert.

Was wäre, wenn Sie all dies neben der Initialisierung der Schaltfläche tun könnten? Was wäre, wenn Sie keine Rückrufe angeben müssten und stattdessen Blöcke angeben könnten, die jederzeit hinzugefügt und entfernt werden können? Warum sollten Sie sich Gedanken über die Implementierung von Selektoren machen?

Lösung

```

import Foundation
import UIKit

protocol RemovableTarget {
    func enable();
    func disable();
}

extension UIControl {
    func addEventHandler(event: UIControlEvents, runnable: (control: UIControl) -> Void) -> RemovableTarget {

        class Target : RemovableTarget {
            private var event: UIControlEvents
            private weak var control: UIControl?
            private var runnable: (control: UIControl) -> Void

            private init(event: UIControlEvents, control: UIControl, runnable: (control: UIControl) -> Void) {
                self.event = event
                self.control = control
                self.runnable = runnable
            }

            @objc
            private func run(_ control: UIControl) {
                runnable(control: control)
            }

            private func enable() {
                control?.addTarget(self, action: #selector(Target.run(_:)), for: event)
                objc_setAssociatedObject(self, unsafeAddress(of: self), self,
                .OBJC_ASSOCIATION_RETAIN)
            }

            private func disable() {
                control?.removeTarget(self, action: #selector(Target.run(_:)), for:

```

```

self.event)
        objc_setAssociatedObject(self, unsafeAddress(of: self), nil,
        .OBJC_ASSOCIATION_ASSIGN)
    }
}

let target = Target(event: event, control: self, runnable: runnable)
target.enable()
return target
}
}

```

Das obige ist eine einfache Erweiterung von `UIControl`. Es fügt eine innere private Klasse hinzu, die über eine Callback-Funktion `func run(_ control: UIControl)`, die als Ereignisaktion verwendet wird.

Als Nächstes verwenden wir die `object association`, um das Ziel hinzuzufügen und zu entfernen, da es von `UIControl` nicht beibehalten wird.

Die Event-Handler-Funktion gibt ein `Protocol` zurück, um die inneren Abläufe der `Target` Klasse auszublenden, aber auch, `disable` das `Target` jederzeit `enable` und `disable` zu können.

Verwendungsbeispiel:

```

import Foundation
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        //Create a button.
        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))

        //Add an event action block/listener -- Handles Button Press.
        let target = button.addEventHandler(event: .touchUpInside) { (control) in
            print("Pressed")
        }

        self.view.addSubview(button)

        //Example of enabling/disabling the listener/event-action-block.
        DispatchQueue.main.after(when: DispatchTime.now() + 5) {
            target.disable() //Disable the listener.

            DispatchQueue.main.after(when: DispatchTime.now() + 5) {
                target.enable() //Enable the listener.
            }
        }
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}

```

```
}
```

UIViewController - Ereignisbehandlung mit Blöcken online lesen:

<https://riptutorial.com/de/ios/topic/3180/uicontrol---ereignisbehandlung-mit-blocken>

Kapitel 161: UIDatePicker

Bemerkungen

`UIDatePicker` erbt nicht von `UIPickerView`, verwaltet jedoch ein benutzerdefiniertes Auswahlfenster als Übersicht.

Examples

Erstellen Sie eine Datumsauswahl

Schnell

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200))
```

Ziel c

```
UIDatePicker *datePicker = [[UIDatePicker alloc] initWithFrame:CGRectMake(x: 0, y: 0, width: 320, height: 200)];
```

Minimum-Maximum-Datum einstellen

Sie können das minimale und das maximale Datum festlegen, das `UIDatePicker` anzeigen kann.

Mindestdatum

```
[datePicker setMinimumDate:[NSDate date]];
```

Maximales Datum

```
[datePicker setMaximumDate:[NSDate date]];
```

Modi

`UIDatePicker` verfügt über verschiedene Auswahlmodi.

```
enum UIDatePickerMode : Int {  
    case Time  
    case Date  
    case DateAndTime  
    case CountdownTimer  
}
```

- `Time` - Die Datumsauswahl zeigt Stunden, Minuten und (optional) eine AM / PM-Bezeichnung an.
- `Date` - Die Datumsauswahl zeigt Monate, Tage des Monats und Jahre an.
- `DateAndTime` - Die `DateAndTime` zeigt Datumsangaben (als einheitlicher Wochentag, Monat und Tag des Monats) sowie Stunden, Minuten und (optional) eine AM / PM-Bezeichnung an.
- `CountDownTimer` - Die Datumsauswahl zeigt Stunden- und Minutenwerte an, zum Beispiel [1 | 53]. Die Anwendung muss einen Timer einstellen, der im richtigen Intervall ausgelöst wird, und die Datumsauswahl einstellen, wenn die Sekunden abfallen.

Festlegen der Eigenschaft `datePickerMode`

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.datePickerMode = .Date
```

Minutenintervall einstellen

Sie können die Eigenschaft `minuteInterval` ändern, um das vom Minutenrad angezeigte Intervall `minuteInterval` . Der Standardwert ist 1, der Maximalwert ist 30.

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.minuteInterval = 15
```

Countdown-Dauer

Der `NSTimeInterval` Wert dieser Eigenschaft gibt die Sekunden an, ab denen die Datumsauswahl im Countdown-Timer-Modus heruntergezählt wird. Wenn der Modus der Datumsauswahl nicht `CountDownTimer` , wird dieser Wert ignoriert. Maximalwert ist 86,399 Sekunden (23:59).

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.countDownDuration = 60 * 60
```

UIDatePicker online lesen: <https://riptutorial.com/de/ios/topic/5643/uidatepicker>

Kapitel 162: UIDevice

Parameter

Eigentum	Beschreibung
Name	Der Name, der das Gerät identifiziert.
systemName: String	Der Name des Betriebssystems, das auf dem Gerät ausgeführt wird und vom Empfänger dargestellt wird.
Modell: String	Das Modell des Geräts.
systemVersion: String	Die aktuelle Version des Betriebssystems.

Bemerkungen

Die UIDevice-Klasse stellt eine Singleton-Instanz bereit, die das aktuelle Gerät darstellt. Von dieser Instanz können Sie Informationen zum Gerät abrufen, z. B. den zugewiesenen Namen, das Gerätemodell sowie den Namen und die Version des Betriebssystems.

Examples

Abrufen des Modellnamens des iOS-Geräts

Schnell 2

```
import UIKit

extension UIDevice {

    var modelName: String {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8 where value != 0 else { return identifier
            }
            return identifier + String(UnicodeScalar(UInt8(value)))
        }

        switch identifier {
        case "iPod5,1": return "iPod Touch 5"
        case "iPod7,1": return "iPod Touch 6"
        case "iPhone3,1", "iPhone3,2", "iPhone3,3": return "iPhone 4"
        case "iPhone4,1": return "iPhone 4s"
        }
    }
}
```



```

    case "iPhone5,1", "iPhone5,2":           return "iPhone 5"
    case "iPhone5,3", "iPhone5,4":           return "iPhone 5c"
    case "iPhone6,1", "iPhone6,2":           return "iPhone 5s"
    case "iPhone7,2":                         return "iPhone 6"
    case "iPhone7,1":                         return "iPhone 6 Plus"
    case "iPhone8,1":                         return "iPhone 6s"
    case "iPhone8,2":                         return "iPhone 6s Plus"
    case "iPhone9,1", "iPhone9,3":           return "iPhone 7"
    case "iPhone9,2", "iPhone9,4":           return "iPhone 7 Plus"
    case "iPhone8,4":                         return "iPhone SE"
    case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
    case "iPad3,1", "iPad3,2", "iPad3,3":     return "iPad 3"
    case "iPad3,4", "iPad3,5", "iPad3,6":     return "iPad 4"
    case "iPad4,1", "iPad4,2", "iPad4,3":     return "iPad Air"
    case "iPad5,3", "iPad5,4":               return "iPad Air 2"
    case "iPad2,5", "iPad2,6", "iPad2,7":     return "iPad Mini"
    case "iPad4,4", "iPad4,5", "iPad4,6":     return "iPad Mini 2"
    case "iPad4,7", "iPad4,8", "iPad4,9":     return "iPad Mini 3"
    case "iPad5,1", "iPad5,2":               return "iPad Mini 4"
    case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
    case "AppleTV5,3":                       return "Apple TV"
    case "i386", "x86_64":                   return "Simulator"
    default:                                  return identifier
}
}

if UIDevice.currentDevice().modelName == "iPhone 6 Plus" {
    // is an iPhone 6 Plus
}

```

Swift 3

```

import UIKit

public extension UIDevice {

    var modelName: String {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8 , value != 0 else { return identifier
        }

            return identifier + String(UnicodeScalar(UInt8(value)))
        }

        switch identifier {
        case "iPod5,1":           return "iPod Touch 5"
        case "iPod7,1":           return "iPod Touch 6"
        case "iPhone3,1", "iPhone3,2", "iPhone3,3": return "iPhone 4"
        case "iPhone4,1":         return "iPhone 4s"
        case "iPhone5,1", "iPhone5,2": return "iPhone 5"
        case "iPhone5,3", "iPhone5,4": return "iPhone 5c"
        case "iPhone6,1", "iPhone6,2": return "iPhone 5s"
        case "iPhone7,2":         return "iPhone 6"
        case "iPhone7,1":         return "iPhone 6 Plus"
        case "iPhone8,1":         return "iPhone 6s"
        case "iPhone8,2":         return "iPhone 6s Plus"
        case "iPhone9,1", "iPhone9,3": return "iPhone 7"

```

```

        case "iPhone9,2", "iPhone9,4":           return "iPhone 7 Plus"
        case "iPhone8,4":                       return "iPhone SE"
        case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
        case "iPad3,1", "iPad3,2", "iPad3,3":   return "iPad 3"
        case "iPad3,4", "iPad3,5", "iPad3,6":   return "iPad 4"
        case "iPad4,1", "iPad4,2", "iPad4,3":   return "iPad Air"
        case "iPad5,3", "iPad5,4":             return "iPad Air 2"
        case "iPad2,5", "iPad2,6", "iPad2,7":   return "iPad Mini"
        case "iPad4,4", "iPad4,5", "iPad4,6":   return "iPad Mini 2"
        case "iPad4,7", "iPad4,8", "iPad4,9":   return "iPad Mini 3"
        case "iPad5,1", "iPad5,2":             return "iPad Mini 4"
        case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
        case "AppleTV5,3":                     return "Apple TV"
        case "i386", "x86_64":                 return "Simulator"
        default:                               return identifier
    }
}

if UIDevice.current.modelName == "iPhone 7" {
    // is an iPhone 7
}

```

Akkustatus und Akkuladestand abrufen

```

override func viewDidLoad() {
    super.viewDidLoad()
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryStateDidChange:")), name: NSNotification.Name.UIDeviceBatteryStateDidChange,
object: nil)
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryLevelDidChange:")), name: NSNotification.Name.UIDeviceBatteryLevelDidChange,
object: nil)

    // Stuff...
}

func batteryStateDidChange(notification: NSNotification){
    // The stage did change: plugged, unplugged, full charge...
}

func batteryLevelDidChange(notification: NSNotification){

    let batteryLevel = UIDevice.current.batteryLevel
    if batteryLevel < 0.0 {
        print(" -1.0 means battery state is UIDeviceBatteryStateUnknown")
        return
    }

    print("Battery Level : \(batteryLevel * 100)%")
    // The battery's level did change (98%, 99%, ...)
}

```

Gerät identifizieren und bedienen

```

UIDevice *deviceInfo = [UIDevice currentDevice];
NSLog(@"Device Name %@", deviceInfo.name);

```

```

//Ex: myIphone6s
NSLog(@"System Name %@", deviceInfo.systemName);
//Device Name iPhone OS
NSLog(@"System Version %@", deviceInfo.systemVersion);
//System Version 9.3
NSLog(@"Model %@", deviceInfo.model);
//Model iPhone
NSLog(@"Localized Model %@", deviceInfo.localizedModel);
//Localized Model iPhone
int device=deviceInfo.userInterfaceIdiom;
//UIUserInterfaceIdiomPhone=0
//UIUserInterfaceIdiomPad=1
//UIUserInterfaceIdiomTV=2
//UIUserInterfaceIdiomCarPlay=3
//UIUserInterfaceIdiomUnspecified=-1
NSLog(@"identifierForVendor %@", deviceInfo.identifierForVendor);
//identifierForVendor <__NSConcreteUUID 0x7a10ae20> 556395DC-0EB4-4FD5-BC7E-B16F612ECC6D

```

Die Ausrichtung des Geräts ermitteln

```

UIDevice *deviceInfo = [UIDevice currentDevice];
int d = deviceInfo.orientation;

```

`deviceInfo.orientation` gibt einen `UIDeviceOrientation`-Wert zurück, der wie folgt dargestellt wird:

```

UIDeviceOrientationUnknown 0
UIDeviceOrientationPortrait 1
UIDeviceOrientationPortraitUpsideDown 2
UIDeviceOrientationLandscapeLeft 3
UIDeviceOrientationLandscapeRight 4
UIDeviceOrientationFaceUp 5
UIDeviceOrientationFaceDown 6

```

Auf Änderungen der Geräteorientierung in einem View Controller achten:

```

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    [[UIDevice currentDevice] beginGeneratingDeviceOrientationNotifications];
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(deviceOrientationDidChange)
                                             name:UIDeviceOrientationDidChangeNotification
                                             object:nil];
}

-(void)deviceOrientationDidChange
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (orientation == UIDeviceOrientationPortrait || orientation ==
UIDeviceOrientationPortraitUpsideDown) {
        [self changedToPortrait];
    } else if (orientation == UIDeviceOrientationLandscapeLeft || orientation ==
UIDeviceOrientationLandscapeRight) {
        [self changedToLandscape];
    }
}

-(void)changedToPortrait

```

```

{
    // Function Body
}

-(void)changedToLandscape
{
    // Function Body
}

```

So deaktivieren Sie die Überprüfung auf Orientierungsänderungen:

```

- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}

```

Den Batteriezustand des Geräts ermitteln

```

//Get permission for Battery Monitoring
[[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
UIDevice *myDevice = [UIDevice currentDevice];

[myDevice setBatteryMonitoringEnabled:YES];
double batLeft = (float)[myDevice batteryLevel] * 100;
NSLog(@"%.f",batLeft);

int d = myDevice.batteryState;
//Returns an Integer Value
//UIDeviceBatteryStateUnknown 0
//UIDeviceBatteryStateUnplugged 1
//UIDeviceBatteryStateCharging 2
//UIDeviceBatteryStateFull 3

//Using notifications for Battery Monitoring
-(void)startMonitoringForBatteryChanges
{
    // Enable monitoring of battery status
    [[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
    // Request to be notified when battery charge or state changes
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
    name:UIDeviceBatteryLevelDidChangeNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
    name:UIDeviceBatteryStateDidChangeNotification object:nil];
}

-(void) checkBatteryStatus
{
    NSLog(@"Battery Level is %.f",[[UIDevice currentDevice] batteryLevel]*100);
    int d=[[UIDevice currentDevice] batteryState];
    if (d==0)
    {
        NSLog(@"Unknown");
    }
    else if (d==1)
    {
        NSLog(@"Unplugged");
    }
    else if (d==2)
    {

```

```
        NSLog(@"Charging");
    }
    else if (d==3)
    {
        NSLog(@"Battery Full");
    }
}
```

Verwendung des Näherungssensors

```
//Enabling the proximity Sensor
- (void)viewWillAppear:(BOOL)animated {

    [super viewWillAppear:animated];
    [[UIDevice currentDevice] setProximityMonitoringEnabled:YES];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(sensorStateMonitor:) name:@"UIDeviceProximityStateDidChangeNotification"
object:nil];
}

- (void)sensorStateMonitor:(NSNotificationCenter *)notification
{
    if ([[UIDevice currentDevice] proximityState] == YES)
    {
        NSLog(@"Device is close to user.");
    }

    else
    {
        NSLog(@"Device is not closer to user.");
    }
}
```

UIDevice online lesen: <https://riptutorial.com/de/ios/topic/4878/uidevice>

Kapitel 163: UIFeedbackGenerator

Einführung

`UIFeedbackGenerator` und seine Unterklassen bieten eine öffentliche Schnittstelle für die Taptic Engine®, die auf iOS-Geräten ab iPhone 7 verfügbar ist. Haptics, gebrandet Taptics, bieten taktilen Feedback für Bildschirmereignisse. Während viele Systemsteuerelemente bereits `UIFeedbackGenerator`, können Entwickler `UIFeedbackGenerator` Unterklassen verwenden, um benutzerdefinierten Steuerelementen und anderen Ereignissen `UIFeedbackGenerator` hinzuzufügen. `UIFeedbackGenerator` ist eine abstrakte Klasse, die nicht direkt verwendet werden sollte, vielmehr verwenden Entwickler eine ihrer Unterklassen.

Examples

Trigger Impact Haptic

Das Beispiel zeigt, wie Sie mit einem `UIImpactFeedbackGenerator` nach einem Tastendruck eine `UIImpactFeedbackGenerator` auslösen.

Schnell

```
class ViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Impact", for: .normal)
        button.setTitleColor(UIColor.gray, for: .normal)
        return button
    }()

    // Choose between heavy, medium, and light for style
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .heavy)

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:
        .touchUpInside)

        // Primes feedback generator for upcoming events and reduces latency
        impactFeedbackGenerator.prepare()
    }

    func didPressButton(sender: UIButton)
    {
        // Triggers haptic
    }
}
```

```
        impactFeedbackGenerator.impactOccurred()
    }
}
```

Ziel c

```
@interface ViewController ()
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) UIButton *button;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressButton:)
    forControlEvents:UIControlEventTouchUpInside];

    // Choose between heavy, medium, and light for style
    self.impactFeedbackGenerator = [[UIImpactFeedbackGenerator alloc]
    initWithStyle:UIImpactFeedbackStyleHeavy];

    // Primes feedback generator for upcoming events and reduces latency
    [self.impactFeedbackGenerator prepare];
}

- (void)didPressButton:(UIButton *)sender
{
    // Triggers haptic
    [self.impactFeedbackGenerator impactOccurred];
}

#pragma mark - Lazy Init
- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc]init];
        _button.translatesAutoresizingMaskIntoConstraintsIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Impact" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor grayColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end
```

UIFeedbackGenerator online lesen: <https://riptutorial.com/de/ios/topic/10048/uifeedbackgenerator>

Kapitel 164: UIFont

Einführung

`UIFont` ist eine Klasse, die zum Abrufen und Festlegen von Informationen zur Schriftart verwendet wird. Es erbt von `NSObject` und entspricht `Hashable`, `Equatable`, `CVarArg` und `NSCopying`.

Examples

UIFont deklarieren und initialisieren

Sie können eine `UIFont` wie folgt deklarieren:

```
var font: UIFont!
```

`UIFont` verfügt über weitere `init()` Methoden:

- `UIFont.init(descriptor: UIFontDescriptor, size: CGFloat)`
- `UIFont.init(name: String, size: CGFloat)`

Daher können Sie eine `UIFont` wie `UIFont` initialisieren:

```
let font = UIFont(name: "Helvetica Neue", size: 15)
```

Die Standardschriftart ist `System`, Größe 17.

Ändern der Schriftart eines Labels

Um die Textschriftart einer Beschriftung zu ändern, müssen Sie auf die `font` zugreifen:

```
label.font = UIFont(name:"Helvetica Neue", size: 15)
```

Der obige Code ändert die Schriftart des Labels in `Helvetica Neue`, Größe 15. Beachten Sie, dass Sie den Namen der Schriftart korrekt buchstabieren müssen. Andernfalls wird dieser Fehler ausgegeben, da der oben angegebene initialisierte Wert optional ist.

Unerwartet wurde beim Entpacken eines optionalen Werts nichts gefunden

UIFont online lesen: <https://riptutorial.com/de/ios/topic/9792/UIFont>

Kapitel 165: UITapGestureRecognizer

Examples

UITapGestureRecognizer

Initialisieren Sie den `UITapGestureRecognizer` mit einem Ziel, in diesem Fall `self`, und einer `action` der es sich um eine Methode mit einem einzigen Parameter handelt: einem `UITapGestureRecognizer`.

Fügen Sie nach der Initialisierung es der Ansicht hinzu, in der es Taps erkennen soll.

Schnell

```
override func viewDidLoad() {
    super.viewDidLoad()
    let recognizer = UITapGestureRecognizer(target: self,
                                          action: #selector(handleTap(_:)))
    view.addGestureRecognizer(recognizer)
}

func handleTap(recognizer: UITapGestureRecognizer) {
}
```

Ziel c

```
- (void)viewDidLoad {
    [super viewDidLoad];
    UITapGestureRecognizer *recognizer =
        [[UITapGestureRecognizer alloc] initWithTarget:self
                                             action:@selector(handleTap:)];
    [self.view addGestureRecognizer:recognizer];
}

- (void)handleTap:(UITapGestureRecognizer *)recognizer {
}
```

Beispiel für die Entlassung der Tastatur durch UITapGestureRecognizer:

Zuerst erstellen Sie die Funktion zum Entfernen der Tastatur:

```
func dismissKeyboard() {
    view.endEditing(true)
}
```

Dann fügen Sie in Ihrem View-Controller eine Tipp-Gestenerkennung hinzu und rufen die gerade erstellte Methode auf

```
let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self, action:
"dismissKeyboard")
    view.addGestureRecognizer(tap)
```

Beispiel für das Abrufen der Gestenposition UITapGestureRecognizer (Swift 3):

```
func handleTap(gestureRecognizer: UITapGestureRecognizer) {
    print("tap working")
    if gestureRecognizer.state == UIGestureRecognizerState.recognized
    {
        print(gestureRecognizer.location(in: gestureRecognizer.view))
    }
}
```

UIPanGestureRecognizer

Pan-Gestenerkennung erkennen Ziehbewegungen. Im folgenden Beispiel wird einem View-Controller ein Bild hinzugefügt, und der Benutzer kann es auf dem Bildschirm ziehen.

Ziel c

```
- (void)viewDidLoad {
    [super viewDidLoad];

    UIImageView *imageView = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"imageToDrag"]];
    [imageView sizeToFit];
    imageView.userInteractionEnabled = YES;
    [self.view addSubview:imageView];

    UIPanGestureRecognizer *pan = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    [imageView addGestureRecognizer:pan];
}

- (void)handlePan:(UIPanGestureRecognizer *)recognizer {
    CGPoint translation = [recognizer translationInView:self.view];
    recognizer.view.center = CGPointMake(recognizer.view.center.x + translation.x,
recognizer.view.center.y + translation.y);
    [recognizer setTranslation:CGPointZero inView:self.view];
}
```

Schnell

```
override func viewDidLoad() {
    super.viewDidLoad()

    let imageView = UIImageView.init(image: UIImage.init(named: "imageToDrag"))
    imageView.sizeToFit()
    imageView.isUserInteractionEnabled = true
    self.view.addSubview(imageView)

    let pan = UIPanGestureRecognizer.init(target: self, action:
#selector(handlePan(recognizer:)))
    imageView.addGestureRecognizer(pan)
}
```

```

func handlePan(recognizer: UIPanGestureRecognizer) {
    let translation = recognizer.translation(in: self.view)
    if let view = recognizer.view {
        view.center = CGPoint(x: view.center.x + translation.x, y: view.center.y +
translation.y)
    }
    recognizer.setTranslation(CGPoint.zero, in: self.view)
}

```

Hinweis: Obwohl `UIPanGestureRecognizer` zum Erkennen von `UIPanGestureRecognizer` nützlich ist, verwenden Sie `UIPanGestureRecognizer`, wenn Sie nur eine grundlegende Geste wie `UISwipeGestureRecognizer`. `UIPanGestureRecognizer` ist die bessere Wahl, wenn Sie auf Methoden wie `translationInView:` oder `velocityInView:` zugreifen möchten.

UITapGestureRecognizer (Doppeltipp)

Beim Doppeltippen wird wie beim einfachen Tippen auch der `UITapGestureRecognizer`. Sie setzen einfach `numberOfTapsRequired` auf `2`.

Schnell

```

override func viewDidLoad() {
    super.viewDidLoad()

    // Double Tap
    let doubleTapGesture = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap))
    doubleTapGesture.numberOfTapsRequired = 2
    doubleTapView.addGestureRecognizer(doubleTapGesture)
}

// Double tap action
func handleDoubleTap() {
    label.text = "Double tap recognized"
}

```

Anmerkungen

- Ein Beispielprojekt finden Sie [hier](#).
- Sie könnten ein dreifaches `numberOfTapsRequired`, indem Sie `numberOfTapsRequired` auf `3`.

UILongPressGestureRecognizer

Mit dem `UILongPressGestureRecognizer` können Sie lange auf eine Ansicht drücken. Sie können die Verzögerung einstellen, bevor die Aktionsmethode aufgerufen wird.

Schnell

```

override func viewDidLoad() {
    super.viewDidLoad()

```

```

// Long Press
let longPressGesture = UILongPressGestureRecognizer(target: self, action:
#selector(handleLongPress(_:)))
    longPressView.addGestureRecognizer(longPressGesture)
}

// Long press action
func handleLongPress(gesture: UILongPressGestureRecognizer) {
    if gesture.state == UIGestureRecognizerState.Began {
        label.text = "Long press recognized"
    }
}
}

```

Anmerkungen

- Ein ausführlicheres Beispielprojekt finden Sie [hier](#) .
- Ändern Sie die `minimumPressDuration` , um die Länge des langen `minimumPressDuration` .

UISwipeGestureRecognizer

Mit Wischgesten können Sie darauf achten, wie der Benutzer den Finger schnell in eine bestimmte Richtung über den Bildschirm bewegt.

Schnell

```

override func viewDidLoad() {
    super.viewDidLoad()

    // Swipe (right and left)
    let swipeRightGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    let swipeLeftGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    swipeRightGesture.direction = UISwipeGestureRecognizerDirection.Right
    swipeLeftGesture.direction = UISwipeGestureRecognizerDirection.Left
    swipeView.addGestureRecognizer(swipeRightGesture)
    swipeView.addGestureRecognizer(swipeLeftGesture)
}

// Swipe action
func handleSwipe(gesture: UISwipeGestureRecognizer) {
    label.text = "Swipe recognized"

    // example task: animate view off screen
    let originalLocation = swipeView.center
    if gesture.direction == UISwipeGestureRecognizerDirection.Right {
        label.text = "Swipe right"
    } else if gesture.direction == UISwipeGestureRecognizerDirection.Left {
        label.text = "Swipe left"
    }
}
}

```

Ziel c

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    UISwipeGestureRecognizer *swipeLeft = [[UISwipeGestureRecognizer alloc] initWithTarget:self
    action:@selector(handleSwipe:)];
    UISwipeGestureRecognizer *swipeRight = [[UISwipeGestureRecognizer alloc] initWithTarget:self
    action:@selector(handleSwipe:)];

    // Setting the swipe direction.
    [swipeLeft setDirection:UISwipeGestureRecognizerDirectionLeft];
    [swipeRight setDirection:UISwipeGestureRecognizerDirectionRight];

    // Adding the swipe gesture on image view
    [self.view addGestureRecognizer:swipeLeft];
    [self.view addGestureRecognizer:swipeRight];
}
//Handling Swipe Gesture Events

- (void)handleSwipe:(UISwipeGestureRecognizer *)swipe {

    if (swipe.direction == UISwipeGestureRecognizerDirectionLeft) {
        NSLog(@"Left Swipe");
    }

    if (swipe.direction == UISwipeGestureRecognizerDirectionRight) {
        NSLog(@"Right Swipe");
    }
}
}

```

Anmerkungen

- Ein ausführlicheres Projektbeispiel finden Sie [hier](#) .

UIPinchGestureRecognizer

Prisen sind eine Zwei-Finger-Geste, bei der sich die Finger näher oder weiter voneinander bewegen. Diese Geste wird im Allgemeinen zum Ändern der Größe einer Ansicht verwendet.

Schnell

```

override func viewDidLoad() {
    super.viewDidLoad()

    // Pinch
    let pinchGesture = UIPinchGestureRecognizer(target: self, action:
    #selector(handlePinch(_:)))
    pinchView.addGestureRecognizer(pinchGesture)
}

// Pinch action
func handlePinch(gesture: UIPinchGestureRecognizer) {
    label.text = "Pinch recognized"

    if gesture.state == UIGestureRecognizerState.Changed {

```

```
        let transform = CGAffineTransformMakeScale(gesture.scale, gesture.scale)
        pinchView.transform = transform
    }
}
```

Anmerkungen

- Ein ausführlicheres Projektbeispiel finden Sie [hier](#) .

UIRotationGestureRecognizer

Mit dem `UIRotationGestureRecognizer` können zwei Finger um ein Zentrum herum `UIRotationGestureRecognizer` . Dies wird im Allgemeinen zum Drehen einer Ansicht verwendet.

Schnell

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Rotate
    let rotateGesture = UIRotationGestureRecognizer(target: self, action:
#selector(handleRotate(_:)))
    rotateView.addGestureRecognizer(rotateGesture)
}

// Rotate action
func handleRotate(gesture: UIRotationGestureRecognizer) {
    label.text = "Rotate recognized"

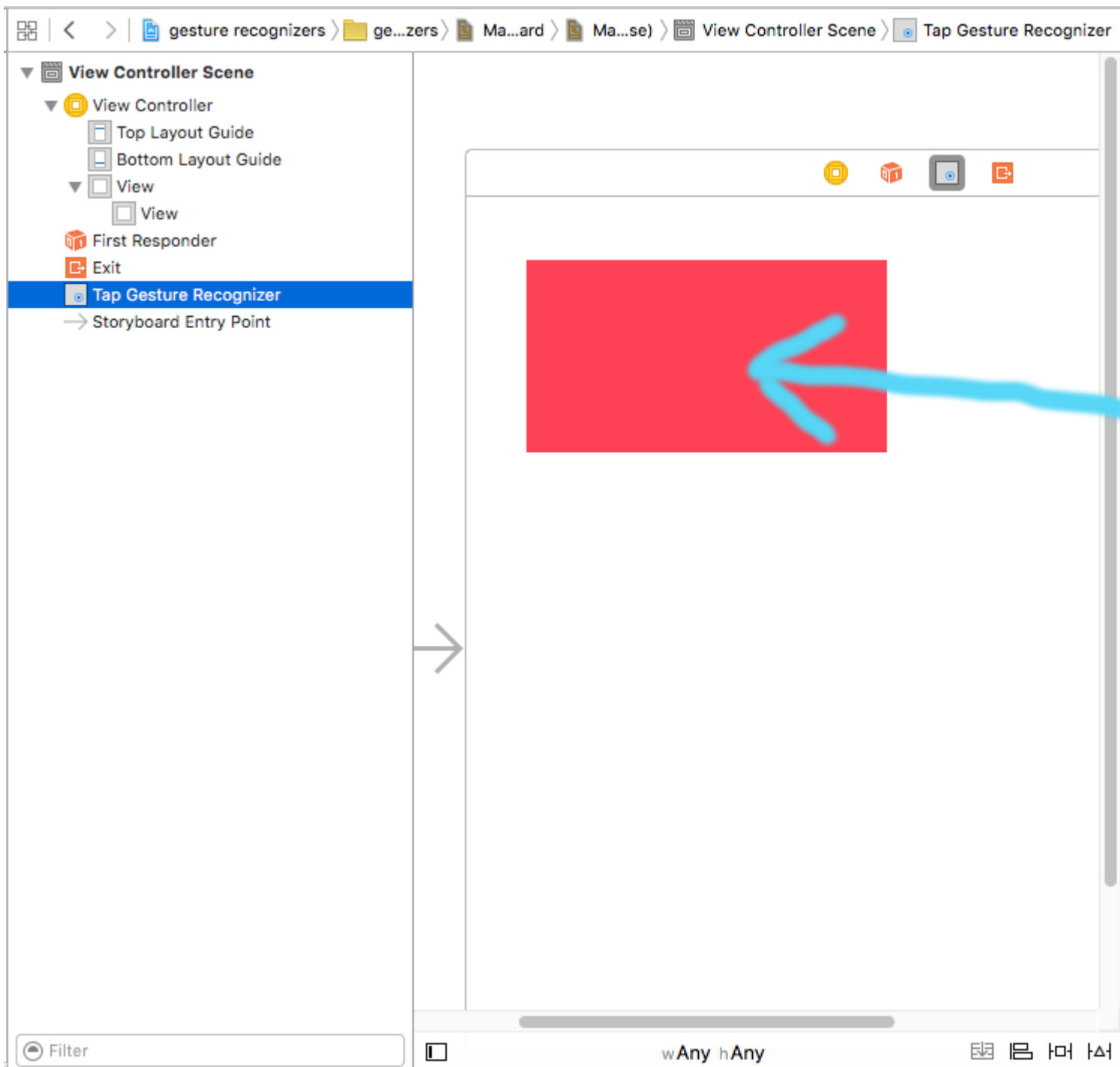
    if gesture.state == UIGestureRecognizerState.Changed {
        let transform = CGAffineTransformMakeRotation(gesture.rotation)
        rotateView.transform = transform
    }
}
```

Anmerkungen

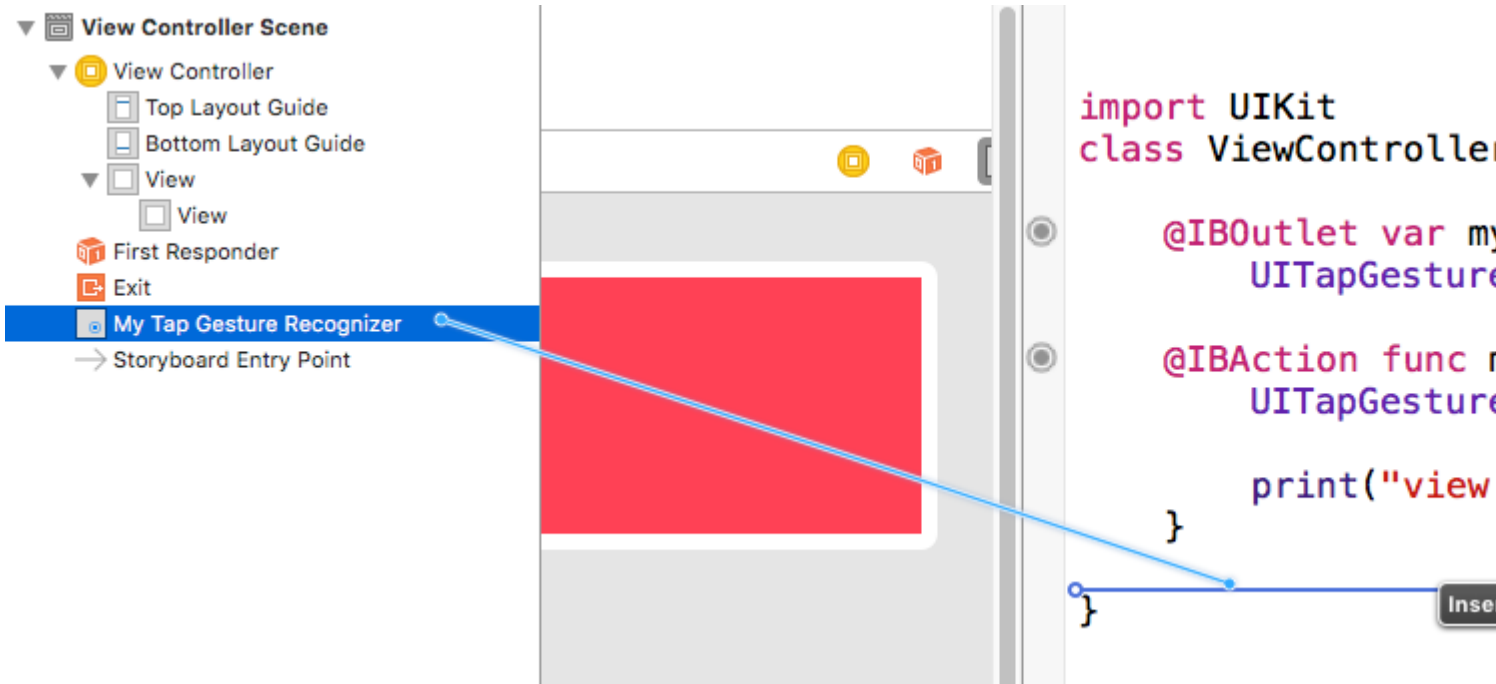
- Ein Beispielprojekt finden Sie [hier](#) .

Hinzufügen einer Gestenerkennung im Interface Builder

Ziehen Sie eine Gestenerkennung aus der Objektbibliothek in Ihre Ansicht.



Ziehen Sie die Steuerung von der Geste in der Dokumentgliederung in Ihren View Controller-Code, um einen Auslass und eine Aktion zu erstellen.



Anmerkungen

- Dieses Beispiel stammt aus [diesem umfassenderen Beispielprojekt](#), das Gestenerkennungsfunktionen demonstriert.

UIGestureRecognizer online lesen: <https://riptutorial.com/de/ios/topic/1289/uigesturerecognizer>

Kapitel 166: UIImage

Bemerkungen

Apple-Entwicklerthema für [UIImage](#)

Examples

UIImage erstellen

Mit lokalem Image

Schnell

```
let image = UIImage(named: "imageFromBundleOrAsset")
```

Ziel c

```
UIImage *image = [UIImage imageNamed:@"imageFromBundleOrAsset"];
```

Hinweis

Die Methode `imageNamed` den Inhalt des Bildes im Arbeitsspeicher. Wenn Sie viele große Bilder auf diese Weise laden, kann dies zu unzureichenden Speicherwarnungen führen, die dazu führen, dass die App beendet wird. Dies kann behoben werden, indem die Methode `imageWithContentsOfFile` von `UIImage`, die kein Caching verwendet.

Mit NSData

Schnell

```
let imageData = Data(base64Encoded: imageString, options:
Data.Base64DecodingOptions.ignoreUnknownCharacters)

let image = UIImage(data: imageData!)
```

Mit UIColor

Schnell

```
let color = UIColor.red
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 UIGraphicsGetCurrentContext()!.setFillColor(color.cgColor)
 UIGraphicsGetCurrentContext()!.fill(CGRect(origin: .zero, size: size))
 let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 UIGraphicsEndImageContext()
```

Ziel c

```
UIColor *color=[UIColor redColor];
CGRect frame = CGRectMake(0, 0, 80, 100);
UIGraphicsBeginImageContext(frame.size);
CGContextRef context = UIGraphicsGetCurrentContext();
CGContextSetFillColorWithColor(context, [color CGColor]);
CGContextFillRect(context, frame);
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
```

Mit Dateiinhalt

Ziel c

Beispiel:

```
UIImage *image = [UIImage imageWithContentsOfFile:[NSBundle mainBundle]
pathForResource:@"cellCountry objectForKey:@"Country_Flag" ofType:nil];
```

Array verwenden:

Beispiel:

```
NSMutableArray *imageArray = [[NSMutableArray alloc] init];

for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {
    NSString *fileName = [NSString stringWithFormat:@"%d.jpg", self.myPhoto];

    // check if a file exists
    if ([UIImage imageNamed:fileName]) {
        // if it exists, add it to the array
        [imageArray addObject:[UIImage imageWithContentsOfFile:[NSBundle
mainBundle]pathForResource:[NSString stringWithFormat:@"%d", fileName] ofType:@""]];
    } else {
        break;
    }
}
}
```

// Bildarray für Animationen hier verwenden:

```
self.myImageView.animationImages = imageArray;
```

Bildobjekte mit Dateinhalt erstellen und initialisieren

Erstellen und Zurückgeben eines Bildobjekts durch Laden der Bilddaten aus der Datei am angegebenen Pfad.

Beispiel:

```
UIImage *image = [UIImage imageWithContentsOfFile:[NSBundle mainBundle]
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil];
```

Array verwenden:

Beispiel

```
NSMutableArray *imageArray = [[NSMutableArray alloc] init];

for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {
    NSString *fileName = [NSString stringWithFormat:@"%d.jpg", self.myPhoto];

    // check if a file exists
    if ([UIImage imageNamed:fileName]) {
        // if it exists, add it to the array
        [imageArray addObject:[UIImage imageWithContentsOfFile:[NSBundle
mainBundle]pathForResource:[NSString stringWithFormat:@"%d", fileName] ofType:@""]];
    } else {
        break;
    }
}

//Using image array for animations here
self.myImageView.animationImages = imageArray;
```

Veränderbare Bild mit Kappen

Im Beispiel einer unten abgebildeten Nachrichtenblase: Die Ecken des Bildes sollten unverändert bleiben, was von `UIEdgeInsets` festgelegt wird, aber der Rahmen und die Bildmitte sollten sich auf die neue Größe erstrecken.



```
let insets = UIEdgeInsetsMake(12.0, 20.0, 22.0, 12.0)
let image = UIImage(named: "test")
image?.resizableImageWithCapInsets(insets, resizingMode: .Stretch)
```

Bilder vergleichen

Die Methode `isEqual:` ist der einzige zuverlässige Weg, um zu bestimmen, ob zwei Bilder die gleichen Bilddaten enthalten. Die von Ihnen erstellten Bildobjekte unterscheiden sich möglicherweise voneinander, selbst wenn Sie sie mit den gleichen zwischengespeicherten Bilddaten initialisieren. Die einzige Möglichkeit, ihre Gleichheit zu bestimmen, ist die Verwendung der Methode `isEqual:` mit der die tatsächlichen Bilddaten verglichen werden. Listing 1 zeigt die korrekten und falschen Möglichkeiten, Bilder zu vergleichen.

Quelle: [Apple-Dokumentation](#)

Schnell

```
// Load the same image twice.
let image1 = UIImage(named: "MyImage")
let image2 = UIImage(named: "MyImage")

// The image objects may be different, but the contents are still equal
if let image1 = image1, image1.isEqual(image2) {
    // Correct. This technique compares the image data correctly.
}

if image1 == image2 {
    // Incorrect! Direct object comparisons may not work.
}
```

Ziel c

```
// Load the same image twice.
UIImage* image1 = [UIImage imageNamed:@"MyImage"];
UIImage* image2 = [UIImage imageNamed:@"MyImage"];

// The image objects may be different, but the contents are still equal
if ([image1 isEqual:image2]) {
    // Correct. This technique compares the image data correctly.
}

if (image1 == image2) {
    // Incorrect! Direct object comparisons may not work.
}
```

Erstellen Sie UIImage mit UIColor

Schnell

```

let color = UIColor.redColor()
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 CGContextSetFillColorWithColor(UIGraphicsGetCurrentContext(), color.CGColor)
 CGContextFillRect(UIGraphicsGetCurrentContext(), CGRect(origin: .zero, size: size))
 let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 UIGraphicsEndImageContext()

```

Swift 3

```

let color = UIColor.red()
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 if let context = UIGraphicsGetCurrentContext() {
     context.setFillColor(color.cgColor)
     context.fill(CGRect(origin: .zero, size: size))
     let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 }
 UIGraphicsEndImageContext()

```

Ziel c:

Fügen Sie diese Methode als Erweiterung von `UIImage` :

```

+ (UIImage *)createImageWithColor: (UIColor *)color {
    CGRect rect=CGRectMake(0.0f, 0.0f, 1.0f, 1.0f);
    UIGraphicsBeginImageContext(rect.size);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetFillColorWithColor(context, [color CGColor]);
    CGContextFillRect(context, rect);

    UIImage *theImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return theImage;
}

```

Farbverlauf mit Farben

Erstellen eines `UIImage` mit Farben in `CGRect`

Schnell:

```

extension UIImage {
    static func gradientImageWithBounds(bounds: CGRect, colors: [CGColor]) -> UIImage {
        let gradientLayer = CAGradientLayer()
        gradientLayer.frame = bounds
        gradientLayer.colors = colors

        UIGraphicsBeginImageContext(gradientLayer.bounds.size)
        gradientLayer.render(in: UIGraphicsGetCurrentContext()!)
    }
}

```

```

        let image = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return image!
    }
}

```

Verwendungszweck:

```

let image = UIImage.gradientImageWithBounds(CGRect(x: 0, y: 0, width: 200, height: 200),
colors: [UIColor.yellowColor().CGColor, UIColor.blueColor().CGColor])

```

Ziel c:

```

+ (UIImage *)gradientImageWithBounds:(CGRect)bounds colors:(NSArray *)colors {
    CAGradientLayer *gradientLayer = [CAGradientLayer layer];
    gradientLayer.frame = bounds;
    gradientLayer.colors = colors;

    UIGraphicsBeginImageContext(gradientLayer.bounds.size);
    [gradientLayer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}

```

Hintergrundfarbe für Farbverläufe

```

+ (CALayer *)gradientBGLayerForBounds:(CGRect)bounds colors:(NSArray *)colors
{
    CAGradientLayer * gradientBG = [CAGradientLayer layer];
    gradientBG.frame = bounds;
    gradientBG.colors = colors;
    return gradientBG;
}

```

Konvertieren Sie UIImage in / aus der base64-Kodierung

Codierung

```

//convert the image to NSData first
let imageData:NSData = UIImagePNGRepresentation(image)!
// convert the NSData to base64 encoding
let strBase64:String =
imageData.base64EncodedStringWithOptions(.Encoding64CharacterLineLength)

```

Dekodierung

```

let dataDecoded:NSData = NSData(base64EncodedString: strBase64, options:
NSDataBase64DecodingOptions(rawValue: 0))!
let decodedimage:UIImage = UIImage(data: dataDecoded)!

```

Machen Sie eine Momentaufnahme einer UIView

```

//Here self.webView is the view whose screenshot I need to take
//The screenshot is saved in jpg format in the application directory to avoid any loss of
quality in retina display devices i.e. all current devices running iOS 10
 UIGraphicsBeginImageContextWithOptions(self.webView.bounds.size, NO, [UIScreen
 mainScreen].scale);
 [self.webView.layer renderInContext:UIGraphicsGetCurrentContext()];
 UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
 NSString *jpgPath = [NSHomeDirectory() stringByAppendingPathComponent:@"Documents/Test.jpg"];
 [UIImageJPEGRepresentation(image, 1.0) writeToFile:jpgPath atomically:YES];
 UIImage *pop=[[UIImage alloc] initWithContentsOfFile:jpgPath];
 //pop is the final image in jpg format and high quality with the exact resolution of the view
 you selected in pixels and not just points

```

Wenden Sie UIColor auf UIImage an

Verwenden Sie dasselbe UIImage mit einer Basisanwendung mit mehreren Designs, indem Sie UIColor wie folgt auf die UIImage-Instanz anwenden.

```

// *** Create an UIImage instance with RenderingMode AlwaysTemplate ***
 UIImage *imgMenu = [[UIImage imageNamed:@"iconMenu"
 imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate];

// *** Now Apply `tintColor` to `UIImageView` of UIImageView or UIButton and convert image in
given color ***
 [btn setImage:imgMenu forState:UIControlStateNormal]; // Set UIImage in UIButton.

 [button.imageView setTintColor:[UIColor blueColor]]; // It changes image color of UIButton to
blue color

```

Nehmen wir an, Sie möchten dasselbe mit UIImageView machen und dann den folgenden Code verwenden

```

 [imageView setImage:imgMenu]; // Assign UIImage to UIImageView
 [imageView setTintColor:[UIColor greenColor]]; // Change imageview image color to green.
 [imageView setTintColor:[UIColor redColor]]; // Change imageview image color to red.

```

Ändern Sie die UIImage-Farbe

Swift Diese Erweiterung zu UIImage hinzufügen:

```

extension UIImage {
    func maskWithColor(color: UIColor) -> UIImage? {

        let maskImage = self.CGImage
        let width = self.size.width
        let height = self.size.height
        let bounds = CGRectMake(0, 0, width, height)

        let colorSpace = CGColorSpaceCreateDeviceRGB()
        let bitmapInfo = CGBitmapInfo(rawValue: CGImageAlphaInfo.PremultipliedLast.rawValue)
        let bitmapContext = CGContextCreate(nil, Int(width), Int(height), 8, 0,
        colorSpace, bitmapInfo.rawValue) //needs rawValue of bitmapInfo

        CGContextClipToMask(bitmapContext, bounds, maskImage)
    }
}

```

```
CGContextSetFillColorWithColor(bitmapContext, color.CGColor)
CGContextFillRect(bitmapContext, bounds)

//is it nil?
if let cImage = CGContextCreateImage(bitmapContext) {
    let coloredImage = UIImage(CGImage: cImage)

    return coloredImage
} else {
    return nil
}
}
```

Dann, um die Farbe Ihres UIImage zu ändern

```
my_image.maskWithColor(UIColor.blueColor())
```

Gefunden [bei diesem Link](#)

UIImage online lesen: <https://riptutorial.com/de/ios/topic/1409/uiimage>

Kapitel 167: UIImagePickerControllerController

Einführung

UIImagePickerController bietet eine nahezu sofort einsatzbereite Lösung, mit der der Benutzer ein Bild von seinem Gerät auswählen oder mit der Kamera ein Bild aufnehmen und dieses Bild präsentieren kann. Durch die Konformität mit dem UIImagePickerControllerDelegate können Sie eine Logik erstellen, die in Ihrer App angibt, wie das Bild dargestellt werden soll und was damit (mithilfe von didFinishPickingMediaWithInfo) zu tun ist. Außerdem kann festgelegt werden, ob der Benutzer ein Bild auswählen oder ein Bild aufnehmen möchte (mithilfe von UIImagePickerControllerDidCancel).

Examples

Allgemeine Verwendung von UIImagePickerController

Schritt 1: Erstellen Sie den Controller, legen Sie den Delegaten fest und entsprechen Sie dem Protokoll

```
//Swift
class ImageUploadViewController: UIViewController, UIImagePickerControllerDelegate,
UIImagePickerControllerDelegate {

    let imagePickerController = UIImagePickerController()

    override func viewDidLoad() {
        super.viewDidLoad()
        imagePickerController.delegate = self
    }
}

//Objective-C
@interface ImageUploadViewController : UIViewController
<UIImagePickerControllerDelegate, UINavigationControllerDelegate> {

    UIImagePickerController *imagePickerController;
}

@end

@implementation ImageUploadViewController

- (void)viewDidLoad {

    [super viewDidLoad];

    imagePickerController.delegate = self;
}

@end
```

Hinweis: Wir implementieren zwar nichts in `UINavigationControllerDelegate` definiertes

`UINavigationControllerDelegate`, aber `UIImagePickerController` erbt von `UINavigationController` und ändert das Verhalten von `UINavigationController`. Daher müssen wir noch sagen, dass unser View-Controller mit `UINavigationControllerDelegate`.

Schritt 2: Wann immer Sie `UIImagePickerController`:

```
//Swift
self.imagePickerController.sourceType = .Camera // options: .Camera , .PhotoLibrary ,
.SavedPhotosAlbum
self.presentViewController(self.imagePickerController, animated: true, completion: nil)

//Objective-C
imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera; // options:
UIImagePickerControllerSourceTypeCamera, UIImagePickerControllerSourceTypePhotoLibrary,
UIImagePickerControllerSourceTypeSavedPhotosAlbum
[self presentViewController:imagePickerController animated:YES completion:nil];
```

Schritt 3: Implementieren Sie die Delegatmethoden:

```
//Swift
func imagePickerController(picker: UIImagePickerController, didFinishPickingMediaWithInfo
info: [String : AnyObject]) {
    if let pickedImage = info[UIImagePickerControllerOriginalImage] as? UIImage {
        // You have pickedImage now, do your logic here
    }
    self.dismissViewControllerAnimated(true, completion: nil)
}

func imagePickerControllerDidCancel(picker: UIImagePickerController) {
    self.dismissViewControllerAnimated(true, completion: nil)
}

//Objective-C
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {

    UIImage *pickedImage = info[UIImagePickerControllerOriginalImage];

    if (pickedImage) {

        //You have pickedImage now, do your logic here

    }

    [self dismissViewControllerAnimated:YES completion:nil];
}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {

    [self dismissViewControllerAnimated:YES completion:nil];
}
```

UIImagePickerController online lesen:

<https://riptutorial.com/de/ios/topic/3023/uiimagepickercontroller>

Kapitel 168: UIImageView

Examples

Erstellen Sie eine UIImageView

Um eine `UIImageView` programmgesteuert zu erstellen, müssen Sie nur eine Instanz von `UIImageView` :

```
//Swift
let imageView = UIImageView()

//Objective-C
UIImageView *imageView = [[UIImageView alloc] init];
```

Sie können die Größe und Position des `UIImageView` mit einem `CGRect` :

```
//Swift
imageView.frame = CGRect(x: 0, y: 0, width: 200, height: 200)

//Objective-C
imageView.frame = CGRectMake(0,0,200,200);
```

Oder Sie können die Größe während der Initialisierung einstellen:

```
//Swift
UIImageView(frame: CGRect(x: 0, y: 0, width: 200, height: 200))

//Objective-C
UIImageView *imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0,0,200,200);

//Alternative way of defining frame for UIImageView
UIImageView *imageView = [[UIImageView alloc] init];
CGRect imageViewFrame = imageView.frame;
imageViewFrame.size.width = 200;
imageViewFrame.size.height = 200;
imageViewFrame.origin.x = 0;
imageViewFrame.origin.y = 0;
imageView.frame = imageViewFrame;
```

Hinweis: Sie müssen `UIKit` importieren, um eine `UIImageView` .

Zuweisen eines Bildes zu einer UIImageView

Sie können einem `UIImageView` während der Initialisierung oder später mithilfe der `image` `UIImageView` ein Bild zuweisen:

```
//Swift
UIImageView(image: UIImage(named: "image1"))
```

```
UIImageView(image: UIImage(named: "image1"), highlightedImage: UIImage(named: "image2"))

imageView.image = UIImage(named: "image1")

//Objective-C
[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"];

[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"] highlightedImage:[UIImage
imageNamed:@"image2"]];

imageView.image = [UIImage imageNamed:@"image1"];
```

Animieren einer UIImageView

Sie können eine `UIImageView` Animation animieren, indem Sie mithilfe der Animationseigenschaften von `UIImageView` schnell Bilder in einer Sequenz `UIImageView` :

```
imageView.animationImages = [UIImage(named: "image1")!,
                             UIImage(named: "image2")!,
                             UIImage(named: "image3")!,
                             UIImage(named: "image4")!,
                             UIImage(named: "image5")!,
                             UIImage(named: "image6")!,
                             UIImage(named: "image7")!,
                             UIImage(named: "image8")!]

imageView.animationDuration = 0.3
imageView.animationRepeatCount = 1
```

Die `animationImages` Eigenschaft ist ein `Array` von `UIImage`s , das beim Auslösen der Animation von oben nach unten durchlaufen wird.

Die Eigenschaft `animationDuration` ist ein `Double` , das angibt, für wie viele Sekunden die Animation ausgeführt wird.

Die `animationRepeatCount` Eigenschaft ist ein `Int` , das angibt, wie oft die Animation ausgeführt wird.

Um die Animation zu starten und zu stoppen, können Sie die entsprechenden Methoden dazu aufrufen:

```
imageView.startAnimating()
imageView.stopAnimating()
```

Es gibt eine Methode `isAnimating()` die einen `Boolean` Wert zurückgibt, der angibt, ob die Animation gerade läuft oder nicht.

Bitte beachten Sie, dass dies nicht sehr effizient ist, um Animationen zu erstellen: Dies ist ziemlich langsam und ressourcenintensiv. Verwenden Sie für bessere Ergebnisse Layer oder Sprites

Bild zu einem Kreis oder abgerundet machen

Dieses Beispiel zeigt, wie eine `UIView` oder `UIImageView` , die mit einem Radius wie `UIImageView` gerundet wird:



Ziel c

```
someImageView.layer.cornerRadius = CGRectGetHeight(someImageView.frame) / 2;  
someImageView.clipsToBounds = YES;
```

Schnell

```
someImageView.layer.cornerRadius = someImageView.frame.height/2  
// this should alleviate the performance hit that adding transparency may cause - see  
// http://stackoverflow.com/a/6254531/189804  
// Be sure to check scrolling performance with Instruments if you take this approach.  
someImageView.layer.shouldRasterize = true  
someImageView.clipsToBounds = true // All parts of the image that are outside its bounds (the  
// frame) are cut out (makes the rounded corners visible)
```

Es wird vorgeschlagen, dass, wenn Sie automatisches Layout, dass Sie den setzen `someImageView.layer.cornerRadius` Code in `viewDidLayoutSubviews`. Dadurch kann der `cornerRadius` des `cornerRadius` aktualisiert werden, wenn sich die Bildgröße ändert.

```
override func viewDidLayoutSubviews() {  
    super.viewDidLayoutSubviews()  
}
```

```
someImageView.layer.cornerRadius = someImageView.frame.size.width/2
someImageView.layer.masksToBounds = true
}
```

UIImage mit Label maskiert

Dadurch wird das Bild auf die Form der Buchstaben des Etiketts maskiert:

Ziel c

```
self.maskImage.layer.mask = self.maskLabel.layer;
self.maskImage.layer.masksToBounds = YES;
```

Swift 3

```
maskImageView.mask = maskLabel
maskImageView.masksToBounds = true
```

Hier ist das Ergebnis:



Ändern Sie die Farbe eines Bildes

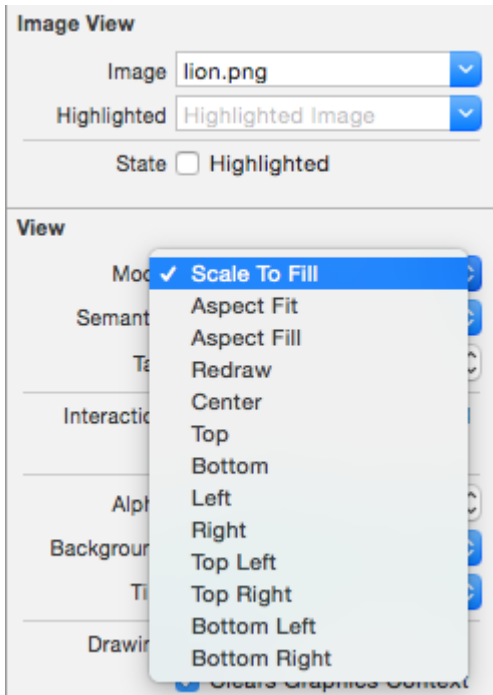
```
//Swift
imageView.tintColor = UIColor.redColor()
imageView.image = imageView.image?.imageWithRenderingMode(.AlwaysTemplate)

//Swift 3
imageView.tintColor = UIColor.red
imageView.image = imageView.image?.withRenderingMode(.alwaysTemplate)

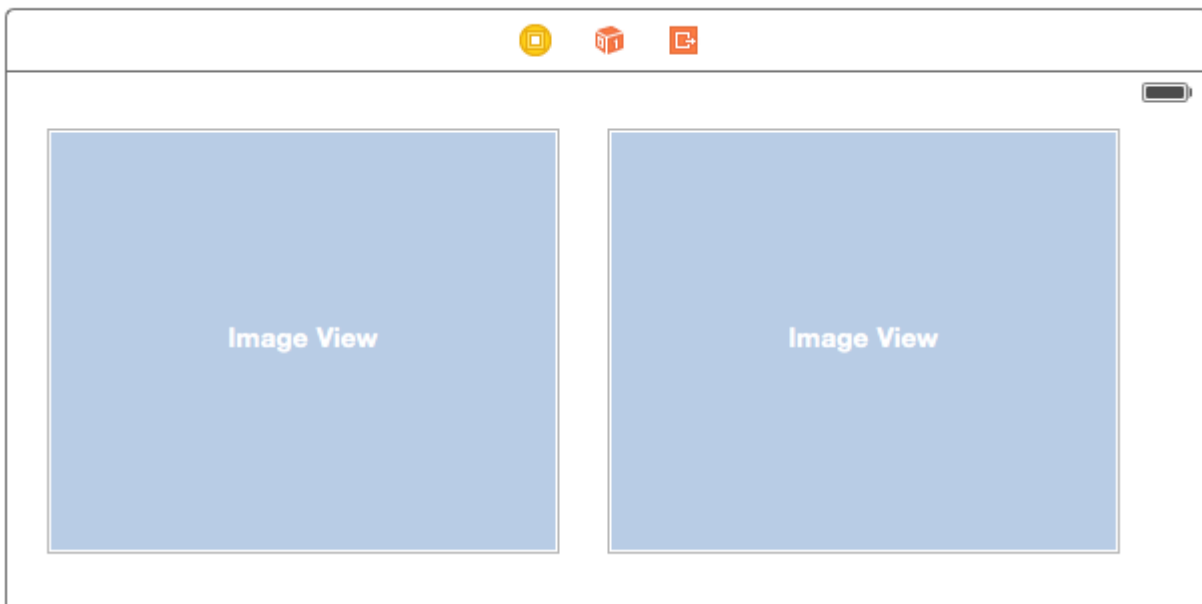
//Objective-C
imageView.tintColor = [UIColor redColor];
imageView.image = [imageView.image imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate]
```

Wie wirkt sich die Mode-Eigenschaft auf ein Bild aus?

Die **Content-Mode-Eigenschaft** einer Ansicht gibt an, wie der Inhalt angeordnet werden soll. Im Interface Builder können die verschiedenen Modi im Informations-Inspektor ausgewählt werden.



Verwenden Sie zwei Bildansichten, um zu sehen, wie die verschiedenen Modi funktionieren.

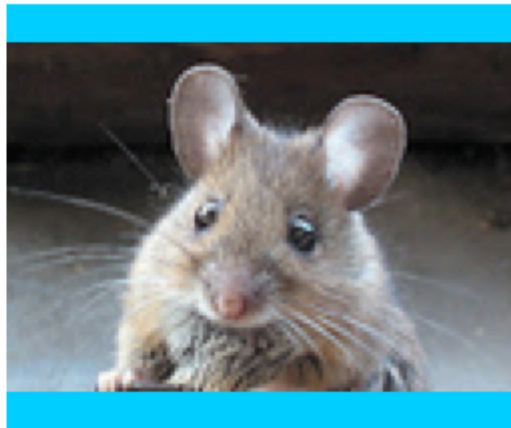


Zum Füllen skalieren



Die `UIImageView` und -breiten werden entsprechend der Größe der `UIImageView` .

Aspect Fit



Die längste Seite (entweder Höhe oder Breite) des Bildes wird entsprechend der Ansicht gestreckt. Dadurch wird das Bild so groß wie möglich, wobei immer noch das gesamte Bild angezeigt wird und die Höhe oder Breite nicht verzerrt wird. (Ich setze den `UIImageView` Hintergrund auf Blau, damit die Größe klar ist.)

Aspekt füllen



Die kürzeste Seite (entweder Höhe oder Breite) des Bildes wird entsprechend der Ansicht gestreckt. Wie bei "Aspect Fit" werden die Proportionen des Bildes nicht von ihrem ursprünglichen Seitenverhältnis verzerrt.

Neu zeichnen

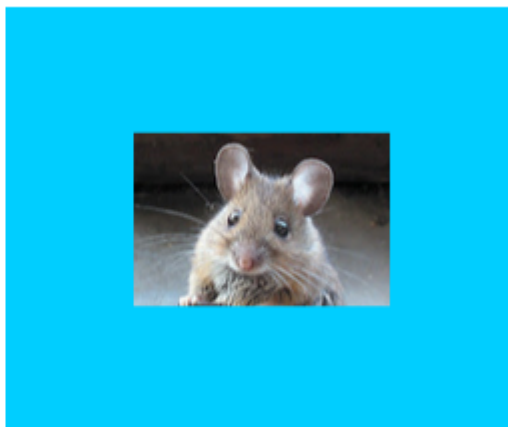


Neuzeichnen ist nur für benutzerdefinierte Ansichten, die ihre eigene Skalierung und Größenänderung vornehmen müssen. Wir verwenden keine benutzerdefinierte Ansicht, daher sollten Sie Redraw nicht verwenden. Beachten Sie, dass `UIImageView` uns hier genau dasselbe Ergebnis liefert wie Scale to Fill, aber es `UIImageView` mehr Arbeit hinter den Kulissen.

Über Redraw sagt die [Apple-Dokumentation](#) :

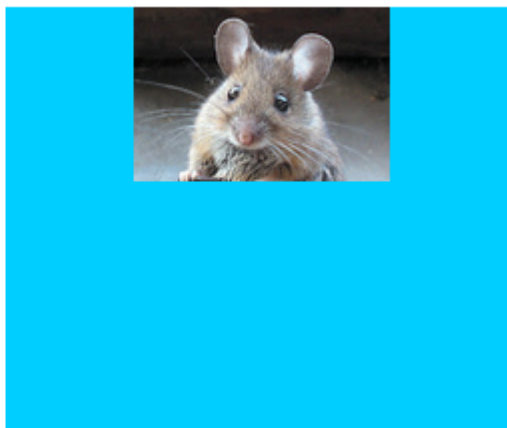
Inhaltsmodi eignen sich zum Wiederherstellen des Inhalts Ihrer Ansicht. Sie können den Inhaltsmodus jedoch auch auf den `UIViewContentModeRedraw` Wert setzen, wenn Sie möchten, dass Ihre benutzerdefinierten Ansichten speziell während Skalierungs- und Größenänderungsvorgängen neu gezeichnet werden. Wenn Sie den Inhaltsmodus Ihrer Ansicht auf diesen Wert setzen, muss das System die `drawRect: -Methode` Ihrer Ansicht als Reaktion auf Geometrieänderungen `drawRect: .` Im Allgemeinen sollten Sie diesen Wert möglichst vermeiden, und Sie sollten ihn auf keinen Fall mit den Standardsystemansichten verwenden.

Center



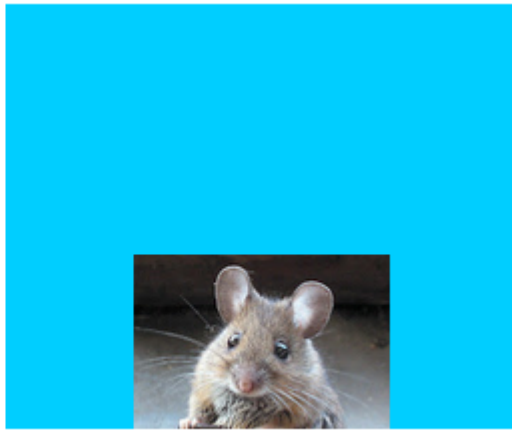
Das Bild wird in der Ansicht zentriert, aber Länge und Breite des Bildes werden nicht gestreckt.

oben



Die Oberkante des Bildes wird horizontal oben in der Ansicht zentriert und die Länge und Breite des Bildes werden nicht gestreckt.

Unterseite



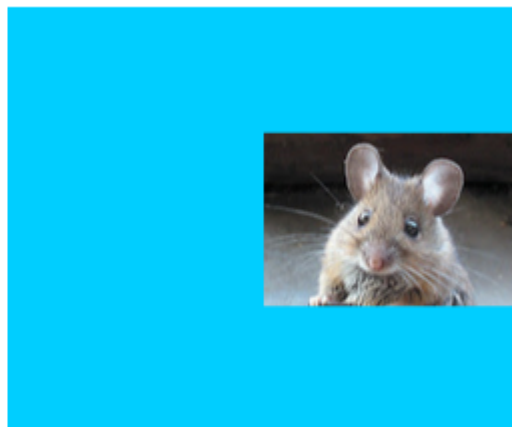
Der untere Bildrand wird am unteren Rand der Ansicht horizontal zentriert, und die Länge und Breite des Bildes werden nicht gestreckt.

Links



Der linke Rand des Bildes ist vertikal links von der Ansicht zentriert, und die Länge und Breite des Bildes werden nicht gestreckt.

Recht



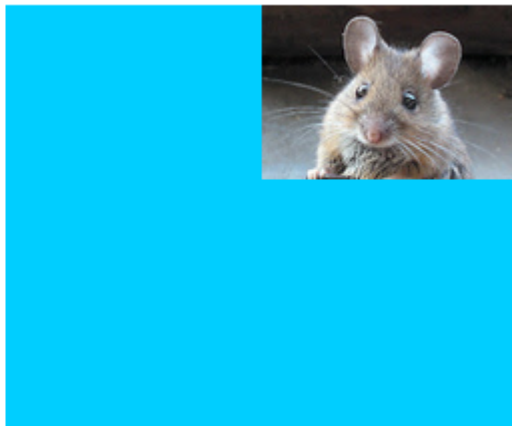
Der rechte Rand des Bildes ist vertikal rechts von der Ansicht zentriert, und die Länge und Breite des Bildes werden nicht gestreckt.

Oben links



Die obere linke Ecke des Bildes befindet sich in der oberen linken Ecke der Ansicht. Die Länge und Breite des Bildes werden nicht gestreckt.

Oben rechts



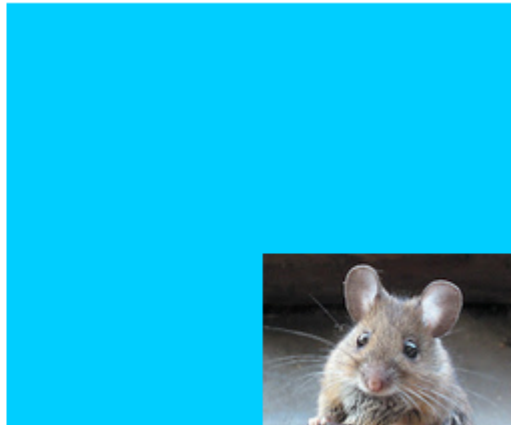
Die obere rechte Ecke des Bildes befindet sich in der rechten oberen Ecke der Ansicht. Die Länge und Breite des Bildes werden nicht gestreckt.

Unten links



Die untere linke Ecke des Bildes befindet sich in der unteren linken Ecke der Ansicht. Die Länge und Breite des Bildes werden nicht gestreckt.

Unten rechts



Die rechte untere Ecke des Bildes befindet sich in der rechten unteren Ecke der Ansicht. Die Länge und Breite des Bildes werden nicht gestreckt.

Anmerkungen

- Dieses Beispiel stammt ursprünglich von [hier](#) .
- Wenn der Inhalt (in unserem Fall das Bild) die gleiche Größe wie die Ansicht hat (in unserem Fall die `UIImageView`), macht das Ändern des Inhaltsmodus keinen erkennbaren Unterschied.
- In [dieser](#) und [dieser](#) Frage finden Sie eine Diskussion über Inhaltsmodi für andere Ansichten als `UIImageView` .
- Um den Inhaltsmodus programmgesteuert festzulegen, führen Sie in Swift folgende Schritte aus:

```
imageView.contentMode = UIViewContentMode.scaleToFill
```

```
imageView.contentMode = UIViewContentMode.scaleAspectFit
imageView.contentMode = UIViewContentMode.scaleAspectFill
imageView.contentMode = UIViewContentMode.redraw
imageView.contentMode = UIViewContentMode.center
imageView.contentMode = UIViewContentMode.top
imageView.contentMode = UIViewContentMode.bottom
imageView.contentMode = UIViewContentMode.left
imageView.contentMode = UIViewContentMode.right
imageView.contentMode = UIViewContentMode.topLeft
imageView.contentMode = UIViewContentMode.topRight
imageView.contentMode = UIViewContentMode.bottomLeft
imageView.contentMode = UIViewContentMode.bottomRight
```

UIImageView online lesen: <https://riptutorial.com/de/ios/topic/695/uiimageview>

Kapitel 169: UIKit Dynamics

Einführung

UIKit Dynamics ist eine vollständige, in UIKit integrierte reale Physik-Engine. Sie können Schnittstellen erstellen, die sich real anfühlen, indem Sie Verhaltensweisen wie Schwerkraft, Anhänge, Kollision und Kräfte hinzufügen. Sie definieren die physischen Merkmale, die Ihre Oberflächenelemente übernehmen sollen, und die Dynamik-Engine kümmert sich um den Rest.

Bemerkungen

Beim Importieren von UIKit Dynamics sollten Sie unbedingt beachten, dass Ansichten, die vom Animator positioniert werden, nicht ohne weiteres mit anderen gängigen iOS-Layout-Methoden positioniert werden können.

Neulinge bei UIKit Dynamics haben oft mit diesem wichtigen Vorbehalt zu kämpfen. Das Platzieren von Einschränkungen in einer Ansicht, die ebenfalls ein Element von `UIDynamicBehavior` führt möglicherweise zu Verwirrung, da sowohl die Auto-Layout-Engine als auch die dynamische Animator-Engine um die entsprechende Position kämpfen. In ähnlicher Weise führt der Versuch, den Rahmen direkt von einer vom Animator gesteuerten Ansicht festzulegen, zu unruhigen Animationen und unerwarteter Platzierung. Wenn Sie eine Ansicht als Element zu einem `UIDynamicBehavior` bedeutet dies, dass der Animator die Verantwortung für das Positionieren einer Ansicht übernimmt. `UIDynamicBehavior` sollten Änderungen der Ansichtspositionen durch den Animator implementiert werden.

Der Rahmen einer Ansicht, der von einem dynamischen Animator aktualisiert wird, kann festgelegt werden. Dem Animator sollte jedoch unmittelbar danach mitgeteilt werden, dass das interne Modell des Animators der Ansichtshierarchie aktualisiert wird. Wenn ich beispielsweise `UILabel`, ein `label`, das ein Element eines `UIGravityBehavior` kann ich es an den oberen `UIGravityBehavior` des Bildschirms verschieben, um zu sehen, wie es wieder fällt, indem es sagt:

Schnell

```
label.frame = CGRect(x: 0.0, y: 0.0, width: label.intrinsicContentSize.width, height:
label.intrinsicContentSize.height)
dynamicAnimator.updateItem(usingCurrentState: label)
```

Ziel c

```
self.label.frame = CGRectMake(0.0, 0.0, self.label.intrinsicContentSize.width,
self.label.intrinsicContentSize.height);
[self.dynamicAnimator updateItemUsingCurrentState: self.label];
```

Danach wendet der Animator das Schwerkraftverhalten von der neuen Position des Etiketts an.

Eine andere verbreitete Technik ist die Verwendung von `UIDynamicBehaviors` zum Positionieren von Ansichten. Wenn beispielsweise eine Ansicht unter einem Berührungseignis positioniert werden soll, ist das Erstellen eines `UIAttachmentBehavior` und das Aktualisieren des `anchorPoint` in entweder `touchesMoved` oder der Aktion von `UIGestureRecognizer` eine effektive Strategie.

Examples

Das fallende Quadrat

Lassen Sie uns ein Quadrat in der Mitte unserer Ansicht zeichnen und es nach unten fallen lassen und an der unteren Kante anhalten, wobei Sie mit der unteren Bildschirmgrenze kollidieren.




```

@IBOutlet var animationView: UIView!
var squareView: UIView!
var collision: UICollisionBehavior!
var animator: UIDynamicAnimator!
var gravity: UIGravityBehavior!

override func viewDidLoad() {
    super.viewDidLoad()
    let squareSize = CGSize(width: 30.0, height: 30.0)
    let centerPoint = CGPoint(x: self.animationView.bounds.midX - (squareSize.width/2), y:
self.animationView.bounds.midY - (squareSize.height/2))
    let frame = CGRect(origin: centerPoint, size: squareSize)
    squareView = UIView(frame: frame)
    squareView.backgroundColor = UIColor.orangeColor()
    animationView.addSubview(squareView)
    animator = UIDynamicAnimator(referenceView: view)
    gravity = UIGravityBehavior(items: [squareView])
    animator.addBehavior(gravity)
    collision = UICollisionBehavior(items: [square])
    collision.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collision)
}

```

Schnelle Ansicht basierend auf Gestengeschwindigkeit

In diesem Beispiel wird gezeigt, wie eine Ansicht eine Schwenkgeste verfolgt und auf physikbasierte Weise abweicht.

Carrier 11:34 AM



Schnell

```

class ViewController: UIViewController
{
    // Adjust to change speed of view from flick
    let magnitudeMultiplier: CGFloat = 0.0008

    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()
}

```

```

}()

lazy var gravity: UIGravityBehavior =
{
    let gravity = UIGravityBehavior(items: [self.orangeView])
    return gravity
}()

lazy var collision: UICollisionBehavior =
{
    let collision = UICollisionBehavior(items: [self.orangeView])
    collision.translatesReferenceBoundsIntoBoundary = true
    return collision
}()

lazy var orangeView: UIView =
{
    let widthHeight: CGFloat = 40.0
    let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
    orangeView.backgroundColor = UIColor.orange
    self.view.addSubview(orangeView)
    return orangeView
}()

lazy var panGesture: UIPanGestureRecognizer =
{
    let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
    return panGesture
}()

lazy var attachment: UIAttachmentBehavior =
{
    let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
    return attachment
}()

override func viewDidLoad()
{
    super.viewDidLoad()
    dynamicAnimator.addBehavior(gravity)
    dynamicAnimator.addBehavior(collision)
    orangeView.addGestureRecognizer(panGesture)
}

override func viewDidLoadSubviews()
{
    super.viewDidLoadSubviews()
    orangeView.center = view.center
    dynamicAnimator.updateItem(usingCurrentState: orangeView)
}

func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: view)
    let velocity = sender.velocity(in: view)
    let magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y))
    switch sender.state
    {
        case .began:

```

```

        attachment.anchorPoint = location
        dynamicAnimator.addBehavior(attachment)
    case .changed:
        attachment.anchorPoint = location
    case .cancelled, .ended, .failed, .possible:
        let push = UIPushBehavior(items: [self.orangeView], mode: .instantaneous)
        push.pushDirection = CGVector(dx: velocity.x, dy: velocity.y)
        push.magnitude = magnitude * magnitudeMultiplier
        dynamicAnimator.removeBehavior(attachment)
        dynamicAnimator.addBehavior(push)
    }
}
}

```

Ziel c

```

@interface ViewController ()

@property (nonatomic, assign) CGFloat magnitudeMultiplier;
@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UIGravityBehavior *gravity;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.gravity];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.orangeView addGestureRecognizer:self.panGesture];
    // Adjust to change speed of view from flick
    self.magnitudeMultiplier = 0.0008f;
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    CGFloat magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y));
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {

```

```

        self.attachment.anchorPoint = location;
    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
            sender.state == UIGestureRecognizerStateEnded ||
            sender.state == UIGestureRecognizerStateFailed ||
            sender.state == UIGestureRecognizerStatePossible)
    {
        UIPushBehavior *push = [[UIPushBehavior alloc] initWithItems:@[self.orangeView]
mode:UIPushBehaviorModeInstantaneous];
        push.pushDirection = CGVectorMake(velocity.x, velocity.y);
        push.magnitude = magnitude * self.magnitudeMultiplier;
        [self.dynamicAnimator removeBehavior:self.attachment];
        [self.dynamicAnimator addBehavior:push];
    }
}

#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UIGravityBehavior *)gravity
{
    if (!_gravity)
    {
        _gravity = [[UIGravityBehavior alloc] initWithItems:@[self.orangeView]];
    }
    return _gravity;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc] initWithItems:@[self.orangeView]];
        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
    return _collision;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc] initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)

```

```

    {
        _panGesture = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)
    {
        _attachment = [[UIAttachmentBehavior alloc] initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
}

@end

```

"Sticky Corners" -Effekt mit UIFieldBehaviors

Dieses Beispiel zeigt, wie Sie einen Effekt erzielen können, der dem von FaceTime ähnelt, wenn eine Ansicht angezogen wird, sobald sie in einen bestimmten Bereich eintritt, in diesem Fall zwei Bereiche oben und unten.

Carrier 12:59 PM



Schnell

```

class ViewController: UIViewController
{
    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()

    lazy var collision: UICollisionBehavior =
    {
        let collision = UICollisionBehavior(items: [self.orangeView])
    }()
}

```

```

        collision.translatesReferenceBoundsIntoBoundary = true
        return collision
    }()

    lazy var fieldBehaviors: [UIFieldBehavior] =
    {
        var fieldBehaviors = [UIFieldBehavior]()
        for _ in 0 ..< 2
        {
            let field = UIFieldBehavior.springField()
            field.addItem(self.orangeView)
            fieldBehaviors.append(field)
        }
        return fieldBehaviors
    }()

    lazy var itemBehavior: UIDynamicItemBehavior =
    {
        let itemBehavior = UIDynamicItemBehavior(items: [self.orangeView])
        // Adjust these values to change the "stickiness" of the view
        itemBehavior.density = 0.01
        itemBehavior.resistance = 10
        itemBehavior.friction = 0.0
        itemBehavior.allowsRotation = false
        return itemBehavior
    }()

    lazy var orangeView: UIView =
    {
        let widthHeight: CGFloat = 40.0
        let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
        orangeView.backgroundColor = UIColor.orange
        self.view.addSubview(orangeView)
        return orangeView
    }()

    lazy var panGesture: UIPanGestureRecognizer =
    {
        let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
        return panGesture
    }()

    lazy var attachment: UIAttachmentBehavior =
    {
        let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
        return attachment
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        dynamicAnimator.addBehavior(collision)
        dynamicAnimator.addBehavior(itemBehavior)
        for field in fieldBehaviors
        {
            dynamicAnimator.addBehavior(field)
        }

        orangeView.addGestureRecognizer(panGesture)
    }

```

```

}

override func viewDidLoadSubviews()
{
    super.viewDidLoadSubviews()

    orangeView.center = view.center
    dynamicAnimator.updateItem(usingCurrentState: orangeView)

    for (index, field) in fieldBehaviors.enumerated()
    {
        field.position = CGPoint(x: view.bounds
            .midX, y: view.bounds.height * (0.25 + 0.5 * CGFloat(index)))
        field.region = UIRegion(size: CGSize(width: view.bounds.width, height:
view.bounds.height * 0.5))
    }
}

func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: view)
    let velocity = sender.velocity(in: view)
    switch sender.state
    {
    case .began:
        attachment.anchorPoint = location
        dynamicAnimator.addBehavior(attachment)
    case .changed:
        attachment.anchorPoint = location
    case .cancelled, .ended, .failed, .possible:
        itemBehavior.addLinearVelocity(velocity, for: self.orangeView)
        dynamicAnimator.removeBehavior(attachment)
    }
}
}

```

Ziel c

```

@interface ViewController ()

@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;
@property (nonatomic, strong) UIDynamicItemBehavior *itemBehavior;
@property (nonatomic, strong) NSArray <UIFieldBehavior *> *fieldBehaviors;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.dynamicAnimator addBehavior:self.itemBehavior];
    for (UIFieldBehavior *field in self.fieldBehaviors)
    {

```

```

        [self.dynamicAnimator addBehavior:field];
    }

    [self.orangeView addGestureRecognizer:self.panGesture];
}

- (void)viewDidLoadSubviews
{
    [super viewDidLoadSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];

    for (NSInteger i = 0; i < self.fieldBehaviors.count; i++)
    {
        UITextField *field = self.fieldBehaviors[i];
        field.position = CGPointMake(CGRectGetMidX(self.view.bounds),
CGRectGetHeight(self.view.bounds) * (0.25f + 0.5f * i));
        field.region = [[UIRegion
alloc] initWithSize:CGSizeMake(CGRectGetWidth(self.view.bounds),
CGRectGetHeight(self.view.bounds) * 0.5)];
    }
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        self.attachment.anchorPoint = location;
    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
sender.state == UIGestureRecognizerStateEnded ||
sender.state == UIGestureRecognizerStateFailed ||
sender.state == UIGestureRecognizerStatePossible)
    {
        [self.itemBehavior addLinearVelocity:velocity forItem:self.orangeView];
        [self.dynamicAnimator removeBehavior:self.attachment];
    }
}

#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc] initWithItems:@[self.orangeView]];
    }
}

```



```

        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
    return _collision;
}

- (NSArray <UIFieldBehavior *> *)fieldBehaviors
{
    if (!_fieldBehaviors)
    {
        NSMutableArray *fields = [[NSMutableArray alloc] init];
        for (NSInteger i = 0; i < 2; i++)
        {
            UIFieldBehavior *field = [UIFieldBehavior springField];
            [field addItem:self.orangeView];
            [fields addObject:field];
        }
        _fieldBehaviors = fields;
    }
    return _fieldBehaviors;
}

- (UIDynamicItemBehavior *)itemBehavior
{
    if (!_itemBehavior)
    {
        _itemBehavior = [[UIDynamicItemBehavior alloc] initWithItems:@[self.orangeView]];
        // Adjust these values to change the "stickiness" of the view
        _itemBehavior.density = 0.01;
        _itemBehavior.resistance = 10;
        _itemBehavior.friction = 0.0;
        _itemBehavior.allowsRotation = NO;
    }
    return _itemBehavior;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc] initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)
    {
        _panGesture = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)

```

```

    {
        _attachment = [[UIAttachmentBehavior alloc] initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
}

@end

```

Weitere Informationen zu `UIFieldBehaviors` Sie in der [WWDC-Sitzung 2015 "Neuerungen in UIKit Dynamics und Visual Effects"](#) und im zugehörigen [Beispielcode](#) .

UIDynamicBehavior-gesteuerte benutzerdefinierte Überblendung



Dieses Beispiel zeigt, wie Sie einen benutzerdefinierten Präsentationsübergang erstellen, der von einem zusammengesetzten `UIDynamicBehavior` . Wir können mit dem Erstellen eines Präsentations-View-Controllers beginnen, der einen Modal darstellt.

Schnell

```

class PresentingViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Present", for: .normal)
        button.setTextColor(UIColor.blue, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {

```

```

    super.viewDidLoad()
    button.addTarget(self, action: #selector(self.didPressPresent), for: .touchUpInside)
}

func didPressPresent()
{
    let modal = ModalViewController()
    modal.view.frame = CGRect(x: 0.0, y: 0.0, width: 200.0, height: 200.0)
    modal.modalPresentationStyle = .custom
    modal.transitioningDelegate = modal
    self.present(modal, animated: true)
}
}

```

Ziel c

```

@interface PresentingViewController ()
@property (nonatomic, strong) UIButton *button;
@end

@implementation PresentingViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
    forControlEvents:UIControlEventTouchUpInside];
}

- (void)didPressPresent
{
    ModalViewController *modal = [[ModalViewController alloc] init];
    modal.view.frame = CGRectMake(0.0, 0.0, 200.0, 200.0);
    modal.modalPresentationStyle = UIModalPresentationCustom;
    modal.transitioningDelegate = modal;
    [self presentViewController:modal animated:YES completion:nil];
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Present" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end

```

Wenn Sie auf die Schaltfläche "Anwesend" `ModalViewController`, erstellen Sie einen `ModalViewController` legen den Präsentationsstil auf `.custom` und setzen dessen `transitioningDelegate` auf sich. Dies ermöglicht uns, einen Animator zu verkaufen, der den modalen Übergang steuert.

Wir stellen auch den Rahmen der Ansicht von `modal` so ein, dass er kleiner als der gesamte Bildschirm ist.

Schauen wir uns jetzt `ModalViewController` :

Schnell

```
class ModalViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Dismiss", for: .normal)
        button.setTitleColor(.white, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressDismiss), for: .touchUpInside)
        view.backgroundColor = .red
        view.layer.cornerRadius = 15.0
    }

    func didPressDismiss()
    {
        dismiss(animated: true)
    }
}

extension ModalViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting: UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 1.5, isAppearing: true)
    }

    func animationController(forDismissed dismissed: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 4.0, isAppearing: false)
    }
}
```

Ziel c

```
@interface ModalViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) UIButton *button;
```

```

@end

@implementation ModalViewController

- (void) viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
    forControlEvents:UIControlEventTouchUpInside];
    self.view.backgroundColor = [UIColor redColor];
    self.view.layer.cornerRadius = 15.0f;
}

- (void) didPressPresent
{
    [self dismissViewControllerAnimated:YES completion:nil];
}

- (UIButton *) button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Dismiss" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

-
(id<UIViewControllerAnimatedTransitioning>) animationControllerForPresentedController:(UIViewController
*)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return [[DropOutAnimator alloc] initWithDuration: 1.5 appearing:YES];
}

-
(id<UIViewControllerAnimatedTransitioning>) animationControllerForDismissedController:(UIViewController
*)dismissed
{
    return [[DropOutAnimator alloc] initWithDuration:4.0 appearing:NO];
}

@end

```

Hier erstellen wir den angezeigten View-Controller. Da es sich bei `ModalViewController` um ein eigenes `transitioningDelegate`, ist es auch dafür verantwortlich, ein Objekt zu `ModalViewController`, das die `transitioningDelegate` verwaltet. Für uns heißt das, eine Instanz unserer zusammengesetzten `UIDynamicBehavior` Unterklasse `UIDynamicBehavior`.

Unser Animator wird zwei verschiedene Übergänge haben: einen für die Präsentation und einen für die Ablehnung. Zum Präsentieren wird die Ansicht des Präsentations-View-Controllers von oben eingeblendet. Und zum Abschalten scheint die Aussicht von einem Seil zu schwingen und

fällt dann heraus. Da `DropOutAnimator` mit `UIViewControllerAnimatedTransitioning` Großteil dieser Arbeit bei der Implementierung von `func animateTransition(using transitionContext: UIViewControllerContextTransitioning)` .

Schnell

```
class DropOutAnimator: UIDynamicBehavior
{
    let duration: TimeInterval
    let isAppearing: Bool

    var transitionContext: UIViewControllerContextTransitioning?
    var hasElapsedTimeExceededDuration = false
    var finishTime: TimeInterval = 0.0
    var collisionBehavior: UICollisionBehavior?
    var attachmentBehavior: UIAttachmentBehavior?
    var animator: UIDynamicAnimator?

    init(duration: TimeInterval = 1.0, isAppearing: Bool)
    {
        self.duration = duration
        self.isAppearing = isAppearing
        super.init()
    }
}

extension DropOutAnimator: UIViewControllerAnimatedTransitioning
{
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
    {
        // Get relevant views and view controllers from transitionContext
        guard let fromVC = transitionContext.viewController(forKey: .from),
              let toVC = transitionContext.viewController(forKey: .to),
              let fromView = fromVC.view,
              let toView = toVC.view else { return }

        let containerView = transitionContext.containerView
        let duration = self.transitionDuration(using: transitionContext)

        // Hold reference to transitionContext to notify it of completion
        self.transitionContext = transitionContext

        // Create dynamic animator
        let animator = UIDynamicAnimator(referenceView: containerView)
        animator.delegate = self
        self.animator = animator

        // Presenting Animation
        if self.isAppearing
        {
            fromView.isUserInteractionEnabled = false

            // Position toView just off-screen
            let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
            var toViewInitialFrame = toView.frame
            toViewInitialFrame.origin.y -= toViewInitialFrame.height
            toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
toViewInitialFrame.width * 0.5
            toView.frame = toViewInitialFrame
```

```

containerView.addSubview(toView)

// Prevent rotation and adjust bounce
let bodyBehavior = UIDynamicItemBehavior(items: [toView])
bodyBehavior.elasticity = 0.7
bodyBehavior.allowsRotation = false

// Add gravity at exaggerated magnitude so animation doesn't seem slow
let gravityBehavior = UIGravityBehavior(items: [toView])
gravityBehavior.magnitude = 10.0

// Set collision bounds to include off-screen view and have collision in center
// where our final view should come to rest
let collisionBehavior = UICollisionBehavior(items: [toView])
let insets = UIEdgeInsets(top: toViewInitialFrame.minY, left: 0.0, bottom:
fromViewInitialFrame.height * 0.5 - toViewInitialFrame.height * 0.5, right: 0.0)
collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
self.collisionBehavior = collisionBehavior

// Keep track of finish time in case we need to end the animator before the
animator pauses
self.finishTime = duration + (self.animator?.elapsedTime ?? 0.0)

// Closure that is called after every "tick" of the animator
// Check if we exceed duration
self.action =
{ [weak self] in
    guard let strongSelf = self,
        (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }

    strongSelf.hasElapsedTimeExceededDuration = true
    strongSelf.animator?.removeBehavior(strongSelf)
}

// `DropOutAnimator` is a composit behavior, so add child behaviors to self
self.addChildBehavior(collisionBehavior)
self.addChildBehavior(bodyBehavior)
self.addChildBehavior(gravityBehavior)

// Add self to dynamic animator
self.animator?.addBehavior(self)
}
// Dismissing Animation
else
{
    // Create allow rotation and have a elastic item
    let bodyBehavior = UIDynamicItemBehavior(items: [fromView])
    bodyBehavior.elasticity = 0.8
    bodyBehavior.angularResistance = 5.0
    bodyBehavior.allowsRotation = true

    // Create gravity with exaggerated magnitude
    let gravityBehavior = UIGravityBehavior(items: [fromView])
    gravityBehavior.magnitude = 10.0

    // Collision boundary is set to have a floor just below the bottom of the screen
    let collisionBehavior = UICollisionBehavior(items: [fromView])
    let insets = UIEdgeInsets(top: 0.0, left: -1000, bottom: -225, right: -1000)
    collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
    self.collisionBehavior = collisionBehavior
}

```

```

// Attachment behavior so view will have effect of hanging from a rope
let offset = UIOffset(horizontal: 70.0, vertical: fromView.bounds.height * 0.5)
var anchorPoint = CGPoint(x: fromView.bounds.maxX - 40.0, y: fromView.bounds.minY)
anchorPoint = containerView.convert(anchorPoint, from: fromView)
let attachmentBehavior = UIAttachmentBehavior(item: fromView, offsetFromCenter:
offset, attachedToAnchor: anchorPoint)
attachmentBehavior.frequency = 3.0
attachmentBehavior.damping = 3.0
self.attachmentBehavior = attachmentBehavior

// `DropOutAnimator` is a composit behavior, so add child behaviors to self
self.addChildBehavior(collisionBehavior)
self.addChildBehavior(bodyBehavior)
self.addChildBehavior(gravityBehavior)
self.addChildBehavior(attachmentBehavior)

// Add self to dynamic animator
self.animator?.addBehavior(self)

// Animation has two parts part one is hanging from rope.
// Part two is bouncing off-screen
// Divide duration in two
self.finishTime = (2.0 / 3.0) * duration + (self.animator?.elapsedTime ?? 0.0)

// After every "tick" of animator check if past time limit
self.action =
{ [weak self] in
    guard let strongSelf = self,
        (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }
        strongSelf.hasElapsedTimeExceededDuration = true
        strongSelf.animator?.removeBehavior(strongSelf)
    }
}

func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
{
    // Return the duration of the animation
    return self.duration
}
}

extension DropOutAnimator: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        // Animator has reached stasis
        if self.isAppearing
        {
            // Check if we are out of time
            if self.hasElapsedTimeExceededDuration
            {
                // Move to final positions
                let toView = self.transitionContext?.viewController(forKey: .to)?.view
                let containerView = self.transitionContext?.containerView
                toView?.center = containerView?.center ?? .zero
                self.hasElapsedTimeExceededDuration = false
            }
        }
    }
}

```



```

    }

    // Clean up and call completion

self.transitionContext?.completeTransition(!(self.transitionContext?.transitionWasCancelled ??
false))
    self.childBehaviors.forEach { self.removeChildBehavior($0) }
    animator.removeAllBehaviors()
    self.transitionContext = nil
}
else
{
    if let attachmentBehavior = self.attachmentBehavior
    {
        // If we have an attachment, we are at the end of part one and start part two.
        self.removeChildBehavior(attachmentBehavior)
        self.attachmentBehavior = nil
        animator.addBehavior(self)
        let duration = self.transitionDuration(using: self.transitionContext)
        self.finishTime = 1.0 / 3.0 * duration + animator.elapsedTime
    }
    else
    {
        // Clean up and call completion
        let fromView = self.transitionContext?.viewController(forKey: .from)?.view
        let toView = self.transitionContext?.viewController(forKey: .to)?.view
        fromView?.removeFromSuperview()
        toView?.isUserInteractionEnabled = true

self.transitionContext?.completeTransition(!(self.transitionContext?.transitionWasCancelled ??
false))
        self.childBehaviors.forEach { self.removeChildBehavior($0) }
        animator.removeAllBehaviors()
        self.transitionContext = nil
    }
}
}
}
}

```

Ziel c

```

@interface ObjcDropOutAnimator() <UIDynamicAnimatorDelegate,
UIViewControllerAnimatedTransitioning>
@property (nonatomic, strong) id<UIViewControllerContextTransitioning> transitionContext;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@property (nonatomic, assign) NSTimeInterval finishTime;
@property (nonatomic, assign) BOOL elapsedTimeExceededDuration;
@property (nonatomic, assign, getter=isAppearing) BOOL appearing;
@property (nonatomic, assign) NSTimeInterval duration;
@property (nonatomic, strong) UIAttachmentBehavior *attachBehavior;
@property (nonatomic, strong) UICollisionBehavior * collisionBehavior;

@end

@implementation ObjcDropOutAnimator

- (instancetype) initWithDuration: (NSTimeInterval) duration appearing: (BOOL) appearing
{
    self = [super init];
}

```

```

if (self)
{
    _duration = duration;
    _appearing = appearing;
}
return self;
}

- (void) animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    // Get relevant views and view controllers from transitionContext
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *fromView = fromVC.view;
    UIView *toView = toVC.view;

    UIView *containerView = transitionContext.containerView;
    NSTimeInterval duration = [self transitionDuration:transitionContext];

    // Hold reference to transitionContext to notify it of completion
    self.transitionContext = transitionContext;

    // Create dynamic animator
    UIDynamicAnimator *animator = [[UIDynamicAnimator
alloc] initWithReferenceView:containerView];
    animator.delegate = self;
    self.animator = animator;

    // Presenting Animation
    if (self.isAppearing)
    {
        fromView.userInteractionEnabled = NO;

        // Position toView just above screen
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;
        toView.frame = toViewInitialFrame;

        [containerView addSubview:toView];

        // Prevent rotation and adjust bounce
        UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior
alloc] initWithItems:@[toView]];
        bodyBehavior.elasticity = 0.7;
        bodyBehavior.allowsRotation = NO;

        // Add gravity at exaggerated magnitude so animation doesn't seem slow
        UIGravityBehavior *gravityBehavior = [[UIGravityBehavior
alloc] initWithItems:@[toView]];
        gravityBehavior.magnitude = 10.0f;

        // Set collision bounds to include off-screen view and have collision floor in center
        // where our final view should come to rest
        UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc] initWithItems:@[toView]];

```

```

    UIEdgeInsets insets = UIEdgeInsetsMake(CGRectGetMinY(toViewInitialFrame), 0.0,
CGRectGetHeight(fromViewInitialFrame) * 0.5 - CGRectGetHeight(toViewInitialFrame) * 0.5, 0.0);
    [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
    self.collisionBehavior = collisionBehavior;

    // Keep track of finish time in case we need to end the animator before the animator
pauses
    self.finishTime = duration + self.animator.elapsedTime;

    // Closure that is called after every "tick" of the animator
    // Check if we exceed duration
    __weak ObjcDropOutAnimator *weakSelf = self;
    self.action = ^{
        __strong ObjcDropOutAnimator *strongSelf = weakSelf;
        if (strongSelf)
        {
            if (strongSelf.animator.elapsedTime >= strongSelf.finishTime)
            {
                strongSelf.elapsedTimeExceededDuration = YES;
                [strongSelf.animator removeBehavior:strongSelf];
            }
        }
    };

    // `DropOutAnimator` is a composit behavior, so add child behaviors to self
    [self addChildBehavior:collisionBehavior];
    [self addChildBehavior:bodyBehavior];
    [self addChildBehavior:gravityBehavior];

    // Add self to dynamic animator
    [self.animator addBehavior:self];
}
// Dismissing Animation
else
{
    // Allow rotation and have a elastic item
    UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior alloc]
initWithItems:@[fromView]];
    bodyBehavior.elasticity = 0.8;
    bodyBehavior.angularResistance = 5.0;
    bodyBehavior.allowsRotation = YES;

    // Create gravity with exaggerated magnitude
    UIGravityBehavior *gravityBehavior = [[UIGravityBehavior alloc]
initWithItems:@[fromView]];
    gravityBehavior.magnitude = 10.0f;

    // Collision boundary is set to have a floor just below the bottom of the screen
    UICollisionBehavior *collisionBehavior = [[UICollisionBehavior alloc]
initWithItems:@[fromView]];
    UIEdgeInsets insets = UIEdgeInsetsMake(0, -1000, -225, -1000);
    [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
    self.collisionBehavior = collisionBehavior;

    // Attachment behavior so view will have effect of hanging from a rope
    UIOffset offset = UIOffsetMake(70, -(CGRectGetHeight(fromView.bounds) / 2.0));

    CGPoint anchorPoint = CGPointMake(CGRectGetMaxX(fromView.bounds) - 40,
CGRectGetMinY(fromView.bounds));
    anchorPoint = [containerView convertPoint:anchorPoint fromView:fromView];
    UIAttachmentBehavior *attachBehavior = [[UIAttachmentBehavior alloc]

```

```

initWithItem:fromView offsetFromCenter:offset attachedToAnchor:anchorPoint];
    attachBehavior.frequency = 3.0;
    attachBehavior.damping = 0.3;
    attachBehavior.length = 40;
    self.attachBehavior = attachBehavior;

    // `DropOutAnimator` is a composit behavior, so add child behaviors to self
    [self addChildBehavior:collisionBehavior];
    [self addChildBehavior:bodyBehavior];
    [self addChildBehavior:gravityBehavior];
    [self addChildBehavior:attachBehavior];

    // Add self to dynamic animator
    [self.Animator addBehavior:self];

    // Animation has two parts part one is hanging from rope.
    // Part two is bouncing off-screen
    // Divide duration in two
    self.finishTime = (2./3.) * duration + [self.Animator elapsedTime];

    // After every "tick" of animator check if past time limit
    __weak ObjcDropOutAnimator *weakSelf = self;
    self.action = ^{
        __strong ObjcDropOutAnimator *strongSelf = weakSelf;
        if (strongSelf)
        {
            if ([strongSelf.Animator elapsedTime] >= strongSelf.finishTime)
            {
                strongSelf.elapsedTimeExceededDuration = YES;
                [strongSelf.Animator removeBehavior:strongSelf];
            }
        }
    };
}

-
(NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return self.duration;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    // Animator has reached stasis
    if (self.isAppearing)
    {
        // Check if we are out of time
        if (self.elapsedTimeExceededDuration)
        {
            // Move to final positions
            UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
            UIView *containerView = [self.transitionContext containerView];
            toView.center = containerView.center;
            self.elapsedTimeExceededDuration = NO;
        }

        // Clean up and call completion
        [self.transitionContext completeTransition:[self.transitionContext
transitionWasCancelled]];
    }
}

```

```

    for (UIDynamicBehavior *behavior in self.childBehaviors)
    {
        [self removeChildBehavior:behavior];
    }
    [animator removeAllBehaviors];
    self.transitionContext = nil;
}
// Dismissing
else
{
    if (self.attachBehavior)
    {
        // If we have an attachment, we are at the end of part one and start part two.
        [self removeChildBehavior:self.attachBehavior];
        self.attachBehavior = nil;
        [animator addBehavior:self];
        NSTimeInterval duration = [self transitionDuration:self.transitionContext];
        self.finishTime = 1./3. * duration + [animator elapsedTime];
    }
    else
    {
        // Clean up and call completion
        UIView *fromView = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey].view;
        UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
        [fromView removeFromSuperview];
        toView.userInteractionEnabled = YES;

        [self.transitionContext completeTransition:[self.transitionContext
transitionWasCancelled]];
        for (UIDynamicBehavior *behavior in self.childBehaviors)
        {
            [self removeChildBehavior:behavior];
        }
        [animator removeAllBehaviors];
        self.transitionContext = nil;
    }
}
}
}

```

Als zusammengesetztes Verhalten kann `DropOutAnimator` eine Reihe verschiedener Verhaltensweisen kombinieren, um seine Animationen zu präsentieren und zu verwerfen.

`DropOutAnimator` auch, wie der `action` eines Verhaltens verwendet wird, um die Positionen seiner Elemente sowie die verstrichene Zeit zu überprüfen. `DropOutAnimator` Technik kann zum Entfernen von Ansichten verwendet werden, die sich `DropOutAnimator` bewegen oder Animationen abschneiden, deren Stasis noch nicht erreicht wurde.

Weitere Informationen zur [WWDC-Sitzung 2013 "Fortgeschrittene Techniken mit UIKit Dynamics"](#) sowie [SOLPresentingFun](#)

Schattenübergang mit realer Physik unter Verwendung von UIDynamicBehaviors

In diesem Beispiel wird gezeigt, wie Sie einen interaktiven Präsentationsübergang mit "realer" Physik vornehmen können, die dem Benachrichtigungsbildschirm von iOS ähnelt.

Swipe Down From Top

Zunächst benötigen wir einen Präsentations-View-Controller, über dem der Schirm angezeigt wird. Dieser View-Controller fungiert auch als `UIViewControllerTransitioningDelegate` für unseren präsentierten View-Controller und wird Animatoren für unseren Übergang anbieten. Also werden wir Instanzen unserer interaktiven Animatoren erstellen (einen zum Präsentieren, einen zum Ablehnen). Wir erstellen auch eine Instanz des Schattenansicht-Controllers, die in diesem Beispiel nur ein View-Controller mit einem Label ist. Da die gesamte Interaktion mit der gleichen Schwenkbewegung erfolgen soll, übergeben wir dem interaktiven Ansichts-Controller und dem Schatten Verweise an unsere interaktiven Animatoren.

Schnell

```
class ViewController: UIViewController
{
    var presentingAnimator: ShadeAnimator!
    var dismissingAnimator: ShadeAnimator!
    let shadeVC = ShadeViewController()

    lazy var label: UILabel =
    {
        let label = UILabel()
        label.textColor = .blue
        label.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(label)
        label.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        label.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        return label
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        label.text = "Swipe Down From Top"
        presentingAnimator = ShadeAnimator(isAppearing: true, presentingVC: self, presentedVC:
```

```

shadeVC, transitionDelegate: self)
    dismissingAnimator = ShadeAnimator(isAppearing: false, presentingVC: self,
presentedVC: shadeVC, transitionDelegate: self)
    }
}
extension ViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting:
UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func animationController(forDismissed dismissed: UIViewController) ->
UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func interactionControllerForPresentation(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return presentingAnimator
    }

    func interactionControllerForDismissal(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return dismissingAnimator
    }
}
}

```

Ziel c

```

@interface ObjCViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) ShadeAnimator *presentingAnimator;
@property (nonatomic, strong) ShadeAnimator *dismissingAnimator;
@property (nonatomic, strong) UILabel *label;
@property (nonatomic, strong) ShadeViewController *shadeVC;
@end

@implementation ObjCViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.label.text = @"Swipe Down From Top";
    self.shadeVC = [[ShadeViewController alloc] init];
    self.presentingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:YES presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
    self.dismissingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:NO presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
}

- (UILabel *)label
{
    if (!_label)
    {
        _label = [[UILabel alloc] init];
    }
}

```

```

        _label.textColor = [UIColor blueColor];
        _label.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_label];
        [_label.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_label.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
    }
    return _label;
}

#pragma mark - UINavigationControllerTransitioningDelegate

-
(id<UINavigationControllerAnimatedTransitioning>)animationControllerForPresentedController:(UINavigationController
*)presented presentingController:(UINavigationController *)presenting
sourceController:(UINavigationController *)source
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UINavigationControllerAnimatedTransitioning>)animationControllerForDismissedController:(UINavigationController
*)dismissed
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UINavigationControllerInteractiveTransitioning>)interactionControllerForPresentation:(id<UINavigationController
*)presentation
{
    return self.presentingAnimator;
}

-
(id<UINavigationControllerInteractiveTransitioning>)interactionControllerForDismissal:(id<UINavigationController
*)dismissal
{
    return self.dismissingAnimator;
}

@end

```

Wir möchten eigentlich immer nur unseren Farbton durch einen interaktiven Übergang präsentieren wollen, aber weil `UINavigationControllerTransitioningDelegate` funktioniert, wenn wir keinen regulären Animationscontroller zurückgeben, wird unser interaktiver Controller niemals verwendet. Aus diesem Grund erstellen wir eine `EmptyAnimator` Klasse, die `UINavigationControllerAnimatedTransitioning` entspricht.

Schnell

```

class EmptyAnimator: NSObject
{
}

extension EmptyAnimator: UINavigationControllerAnimatedTransitioning
{
}

```



```

func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
{

}

func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
{
    return 0.0
}
}

```

Ziel c

```

@implementation EmptyAnimator

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

}

- (NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return 0.0;
}

@end

```

Schließlich müssen wir tatsächlich das schaffen `ShadeAnimator`, die eine Unterklasse von ist `UIDynamicBehavior`, die entspricht `UIViewControllerInteractiveTransitioning`.

Schnell

```

class ShadeAnimator: UIDynamicBehavior
{
    // Whether we are presenting or dismissing
    let isAppearing: Bool

    // The view controller that is not the shade
    weak var presentingVC: UIViewController?

    // The view controller that is the shade
    weak var presentedVC: UIViewController?

    // The delegate will vend the animator
    weak var transitionDelegate: UIViewControllerTransitioningDelegate?

    // Feedback generator for haptics on collisions
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)

    // The context given to the animator at the start of the transition
    var transitionContext: UIViewControllerContextTransitioning?

    // Time limit of the dynamic part of the animation
    var finishTime: TimeInterval = 4.0
}

```

```

// The Pan Gesture that drives the transition. Not using EdgePan because triggers
Notifications screen
lazy var pan: UIPanGestureRecognizer =
{
    let pan = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
    return pan
}()

// The dynamic animator that we add `ShadeAnimator` to
lazy var animator: UIDynamicAnimator! =
{
    let animator = UIDynamicAnimator(referenceView: self.transitionContext!.containerView)
    return animator
}()

// init with all of our dependencies
init(isAppearing: Bool, presentingVC: UIViewController, presentedVC: UIViewController,
transitionDelegate: UIViewControllerTransitioningDelegate)
{
    self.isAppearing = isAppearing
    self.presentingVC = presentingVC
    self.presentedVC = presentedVC
    self.transitionDelegate = transitionDelegate
    super.init()
    self.impactFeedbackGenerator.prepare()

    if isAppearing
    {
        self.presentingVC?.view.addGestureRecognizer(pan)
    }
    else
    {
        self.presentedVC?.view.addGestureRecognizer(pan)
    }
}

// Setup and moves shade view controller to just above screen if appearing
func setupViewsForTransition(with transitionContext: UIViewControllerContextTransitioning)
{
    // Get relevant views and view controllers from transitionContext
    guard let fromVC = transitionContext.viewController(forKey: .from),
        let toVC = transitionContext.viewController(forKey: .to),
        let toView = toVC.view else { return }

    let containerView = transitionContext.containerView

    // Hold refrence to transitionContext to notify it of completion
    self.transitionContext = transitionContext
    if isAppearing
    {
        // Position toView just off-screen
        let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
        var toViewInitialFrame = toView.frame
        toViewInitialFrame.origin.y -= toViewInitialFrame.height
        toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
toViewInitialFrame.width * 0.5
        toView.frame = toViewInitialFrame
    }
}

```

```

        containerView.addSubview(toView)
    }
    else
    {
        fromVC.view.addGestureRecognizer(pan)
    }
}

// Handles the entire interaction from presenting/dismissing to completion
func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: transitionContext?.containerView)
    let velocity = sender.velocity(in: transitionContext?.containerView)
    let fromVC = transitionContext?.viewController(forKey: .from)
    let toVC = transitionContext?.viewController(forKey: .to)

    let touchStartHeight: CGFloat = 90.0
    let touchLocationFromBottom: CGFloat = 20.0

    switch sender.state
    {
    case .began:
        let beginLocation = sender.location(in: sender.view)
        if isAppearing
        {
            guard beginLocation.y <= touchStartHeight,
                  let presentedVC = self.presentedVC else { break }
            presentedVC.modalPresentationStyle = .custom
            presentedVC.transitioningDelegate = transitionDelegate
            presentingVC?.present(presentedVC, animated: true)
        }
        else
        {
            guard beginLocation.y >= (sender.view?.frame.height ?? 0.0) - touchStartHeight
            else { break }
            presentedVC?.dismiss(animated: true)
        }
    case .changed:
        guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
        UIView.animate(withDuration: 0.2)
        {
            view.frame.origin.y = location.y - view.bounds.height +
touchLocationFromBottom
        }

        transitionContext?.updateInteractiveTransition(view.frame.maxY / view.frame.height
        )
    case .ended, .cancelled:
        guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
        let isCancelled = isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0)
        addAttachmentBehavior(with: view, isCancelled: isCancelled)
        addCollisionBehavior(with: view)
        addItemBehavior(with: view)

        animator.addBehavior(self)
        animator.delegate = self

        self.action =
        { [weak self] in
            guard let strongSelf = self else { return }

```

```

        if strongSelf.imator.elapsedTime > strongSelf.finishTime
        {
            strongSelf.imator.removeAllBehaviors()
        }
        else
        {
            strongSelf.transitionContext?.updateInteractiveTransition(view.frame.maxY
/ view.frame.height
            )
        }
    }
    default:
        break
    }
}

// Add collision behavior that causes bounce when finished
func addCollisionBehavior(with view: UIView)
{
    let collisionBehavior = UICollisionBehavior(items: [view])
    let insets = UIEdgeInsets(top: -view.bounds.height, left: 0.0, bottom: 0.0, right:
0.0)
    collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
    collisionBehavior.collisionDelegate = self
    self.addChildBehavior(collisionBehavior)
}

// Add attachment behavior that pulls shade either to top or bottom
func addAttachmentBehavior(with view: UIView, isCancelled: Bool)
{
    let anchor: CGPoint
    switch (isAppearing, isCancelled)
    {
        case (true, true), (false, false):
            anchor = CGPoint(x: view.center.x, y: -view.frame.height)
        case (true, false), (false, true):
            anchor = CGPoint(x: view.center.x, y: view.frame.height)
    }
    let attachmentBehavior = UIAttachmentBehavior(item: view, attachedToAnchor: anchor)
    attachmentBehavior.damping = 0.1
    attachmentBehavior.frequency = 3.0
    attachmentBehavior.length = 0.5 * view.frame.height
    self.addChildBehavior(attachmentBehavior)
}

// Makes view more bouncy
func addItemBehavior(with view: UIView)
{
    let itemBehavior = UIDynamicItemBehavior(items: [view])
    itemBehavior.allowsRotation = false
    itemBehavior.elasticity = 0.6
    self.addChildBehavior(itemBehavior)
}
}
extension ShadeAnimator: UIDynamicAnimatorDelegate
{
    // Determines transition has ended
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        guard let transitionContext = self.transitionContext else { return }

```

```

let fromVC = transitionContext.viewController(forKey: .from)
let toVC = transitionContext.viewController(forKey: .to)
guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
switch (view.center.y < 0.0, isAppearing)
{
case (true, true), (true, false):
    view.removeFromSuperview()
    transitionContext.finishInteractiveTransition()
    transitionContext.completeTransition(!isAppearing)
case (false, true):
    toVC?.view.frame = transitionContext.finalFrame(for: toVC!)
    transitionContext.finishInteractiveTransition()
    transitionContext.completeTransition(true)
case (false, false):
    fromVC?.view.frame = transitionContext.initialFrame(for: fromVC!)
    transitionContext.cancelInteractiveTransition()
    transitionContext.completeTransition(false)
}
childBehaviors.forEach { removeChildBehavior($0) }
animator.removeAllBehaviors()
self.animator = nil
self.transitionContext = nil
}
}
extension ShadeAnimator: UICollisionBehaviorDelegate
{
    // Triggers haptics
    func collisionBehavior(_ behavior: UICollisionBehavior, beganContactFor item:
UIDynamicItem, withBoundaryIdentifier identifier: NSCopying?, at p: CGPoint)
    {
        guard p.y > 0.0 else { return }
        impactFeedbackGenerator.impactOccurred()
    }
}
extension ShadeAnimator: UIViewControllerInteractiveTransitioning
{
    // Starts transition
    func startInteractiveTransition(_ transitionContext: UIViewControllerContextTransitioning)
    {
        setupViewsForTransition(with: transitionContext)
    }
}
}

```

Ziel c

```

@interface ShadeAnimator() <UIDynamicAnimatorDelegate, UICollisionBehaviorDelegate>
@property (nonatomic, assign) BOOL isAppearing;
@property (nonatomic, weak) UIViewController *presentingVC;
@property (nonatomic, weak) UIViewController *presentedVC;
@property (nonatomic, weak) NSObject<UIViewControllerTransitioningDelegate>
*transitionDelegate;
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) id<UIViewControllerContextTransitioning> transitionContext;
@property (nonatomic, assign) NSTimeInterval finishTime;
@property (nonatomic, strong) UIPanGestureRecognizer *pan;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@end

@implementation ShadeAnimator

```

```

- (instancetype)initWithIsAppearing:(BOOL)isAppearing presentingVC:(UIViewController
*)presentingVC presentedVC:(UIViewController *)presentedVC
transitionDelegate:(id<UIViewControllerTransitioningDelegate>)transitionDelegate
{
    self = [super init];
    if (self)
    {
        _isAppearing = isAppearing;
        _presentingVC = presentingVC;
        _presentedVC = presentedVC;
        _transitionDelegate = transitionDelegate;
        _impactFeedbackGenerator = [[UIImpactFeedbackGenerator
alloc]initWithStyle:UIImpactFeedbackStyleLight];
        [_impactFeedbackGenerator prepare];
        if (_isAppearing)
        {
            [_presentingVC.view addGestureRecognizer:self.pan];
        }
        else
        {
            [_presentedVC.view addGestureRecognizer:self.pan];
        }
    }
    return self;
}

#pragma mark - Lazy Init
- (UIPanGestureRecognizer *)pan
{
    if (!_pan)
    {
        _pan = [[UIPanGestureRecognizer alloc]initWithTarget:self
action:@selector(handlePan:)];
    }
    return _pan;
}

- (UIDynamicAnimator *)animator
{
    if (!_animator)
    {
        _animator = [[UIDynamicAnimator
alloc]initWithReferenceView:self.transitionContext.containerView];
    }
    return _animator;
}

#pragma mark - Setup
-
(void)setupViewForTransitionWithContext:(id<UIViewControllerContextTransitioning>)transitionContext
{
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *toView = toVC.view;
    UIView *containerView = transitionContext.containerView;
    self.transitionContext = transitionContext;
    if (self.isAppearing)

```

```

    {
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;

        [containerView addSubview:toView];
    }
else
    {
        [fromVC.view addGestureRecognizer:self.pan];
    }
}

#pragma mark - Gesture
- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.transitionContext.containerView];
    CGPoint velocity = [sender velocityInView:self.transitionContext.containerView];
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];

    CGFloat touchStartHeight = 90.0;
    CGFloat touchLocationFromBottom = 20.0;

    if (sender.state == UIGestureRecognizerStateBegan)
    {
        CGPoint beginLocation = [sender locationInView:sender.view];
        if (self.isAppearing)
        {
            if (beginLocation.y <= touchStartHeight)
            {
                self.presentedVC.modalPresentationStyle = UIModalPresentationCustom;
                self.presentedVC.transitioningDelegate = self.transitionDelegate;
                [self.presentingVC presentViewController:self.presentedVC animated:YES
completion:nil];
            }
        }
        else
        {
            if (beginLocation.y >= [sender locationInView:sender.view].y - touchStartHeight)
            {
                [self.presentedVC dismissViewControllerAnimated:true completion:nil];
            }
        }
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        [UIView animateWithDuration:0.2 animations:^(
            CGRect frame = view.frame;
            frame.origin.y = location.y - CGRectGetHeight(view.bounds) +
touchLocationFromBottom;
            view.frame = frame;
        )];
        [self.transitionContext updateInteractiveTransition:CGRectGetMaxY(view.frame) /
CGRectGetHeight(view.frame)];
    }
}

```

```

    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        BOOL isCancelled = self.isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0);
        [self addAttachmentBehaviorWithView:view isCancelled:isCancelled];
        [self addCollisionBehaviorWithView:view];
        [self addItemBehaviorWithView:view];

        [self.animator addBehavior:self];
        self.animator.delegate = self;

        __weak ShadeAnimator *weakSelf = self;
        self.action =
        ^{
            if (weakSelf.animator.elapsedTime > weakSelf.finishTime)
            {
                [weakSelf.animator removeAllBehaviors];
            }
            else
            {
                [weakSelf.transitionContext
updateInteractiveTransition:CGRectGetMaxY(view.frame) / CGRectGetHeight(view.frame)];
            }
        };
    }
}

#pragma mark - UIViewControllerInteractiveTransitioning
- (void)startInteractiveTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    [self setupViewForTransitionWithContext:transitionContext];
}

#pragma mark - Behaviors
- (void)addCollisionBehaviorWithView:(UIView *)view
{
    UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc] initWithItems:@[view]];
    UIEdgeInsets insets = UIEdgeInsetsMake(-CGRectGetHeight(view.bounds), 0.0, 0.0, 0.0);
    [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
    collisionBehavior.collisionDelegate = self;
    [self addChildBehavior:collisionBehavior];
}

- (void)addItemBehaviorWithView:(UIView *)view
{
    UIDynamicItemBehavior *itemBehavior = [[UIDynamicItemBehavior
alloc] initWithItems:@[view]];
    itemBehavior.allowsRotation = NO;
    itemBehavior.elasticity = 0.6;
    [self addChildBehavior:itemBehavior];
}

- (void)addAttachmentBehaviorWithView:(UIView *)view isCancelled:(BOOL)isCancelled
{
    CGPoint anchor;
    if ((self.isAppearing && isCancelled) || (!self.isAppearing && isCancelled))
    {

```



```

        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    else
    {
        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior alloc] initWithItem:view
attachedToAnchor:anchor];
    attachmentBehavior.damping = 0.1;
    attachmentBehavior.frequency = 3.0;
    attachmentBehavior.length = 0.5 * CGRectGetHeight(view.frame);
    [self addChildBehavior:attachmentBehavior];
}

#pragma mark - UICollisionBehaviorDelegate
- (void)collisionBehavior:(UICollisionBehavior *)behavior
beganContactForItem:(id<UIDynamicItem>)item withBoundaryIdentifier:(id<NSCopying>)identifier
atPoint:(CGPoint)p
{
    if (p.y > 0.0)
    {
        [self.impactFeedbackGenerator impactOccurred];
    }
}

#pragma mark - UIDynamicAnimatorDelegate
- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *view = self.isAppearing ? toVC.view : fromVC.view;
    if (view.center.y < 0.0 && (self.isAppearing || !self.isAppearing))
    {
        [view removeFromSuperview];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:!self.isAppearing];
    }
    else if (view.center.y >= 0.0 && self.isAppearing)
    {
        toVC.view.frame = [self.transitionContext finalFrameForViewController:toVC];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:YES];
    }
    else
    {
        fromVC.view.frame = [self.transitionContext initialFrameForViewController:fromVC];
        [self.transitionContext cancelInteractiveTransition];
        [self.transitionContext completeTransition:NO];
    }
    for (UIDynamicBehavior *behavior in self.childBehaviors)
    {
        [self removeChildBehavior:behavior];
    }
    [animator removeAllBehaviors];
    self.animator = nil;
    self.transitionContext = nil;
}

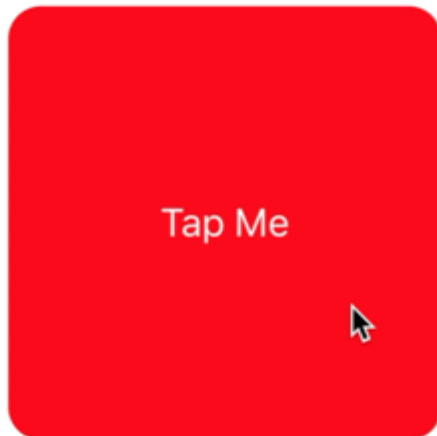
@end

```

Der Animator löst den Beginn des Übergangs aus, wenn die Schwenkgeste beginnt. Und verschiebt einfach die Ansicht, wenn sich die Geste ändert. Wenn die Geste jedoch endet, bestimmt `UIDynamicBehaviors`, ob der Übergang abgeschlossen oder abgebrochen werden soll. Zu diesem Zweck werden Anhängen- und Kollisionsverhalten verwendet. Weitere Informationen finden Sie in der [WWDC-Sitzung 2013 "Fortgeschrittene Techniken mit UIKit Dynamics"](#).

Positionieren Sie dynamische Animationspositionen in Grenzen

In diesem Beispiel wird `UIDynamicItem`, wie das `UIDynamicItem` Protokoll angepasst wird, um Positionsänderungen einer dynamisch animierten Ansicht in Begrenzungsänderungen `UIButton`, um ein `UIButton` zu erstellen, das sich elastisch erweitert und zusammenzieht.



Um zu beginnen, müssen wir ein neues Protokoll erstellen, das `UIDynamicItem` implementiert, `UIDynamicItem` aber auch eine einstellbare und abrufbare `bounds` .

Schnell

```
protocol ResizableDynamicItem: UIDynamicItem
{
    var bounds: CGRect { set get }
}
extension UIView: ResizableDynamicItem {}
```

Ziel c

```
@protocol ResizableDynamicItem <UIDynamicItem>
@property (nonatomic, readwrite) CGRect bounds;
@end
```

Daraufhin erstellen wir ein Wrapper-Objekt, das ein `UIDynamicItem` Objekt `UIDynamicItem` , die Änderungen in der Mitte jedoch auf die Breite und Höhe des Elements abbilden. Wir werden auch Passthroughs für `bounds` und `transform` des zugrunde liegenden Elements bereitstellen. Dies bewirkt, dass alle Änderungen, die der dynamische Animator an den mittleren X- und Y-Werten des zugrunde liegenden Elements vornimmt, auf die Breite und Höhe des Elements angewendet werden.

Schnell

```
final class PositionToBoundsMapping: NSObject, UIDynamicItem
{
    var target: ResizableDynamicItem

    init(target: ResizableDynamicItem)
    {
        self.target = target
        super.init()
    }

    var bounds: CGRect
    {
        get
        {
            return self.target.bounds
        }
    }

    var center: CGPoint
    {
        get
        {
            return CGPoint(x: self.target.bounds.width, y: self.target.bounds.height)
        }
    }

    set
```

```

        {
            self.target.bounds = CGRect(x: 0.0, y: 0.0, width: newValue.x, height: newValue.y)
        }
    }

    var transform: CGAffineTransform
    {
        get
        {
            return self.target.transform
        }

        set
        {
            self.target.transform = newValue
        }
    }
}

```

Ziel c

```

@interface PositionToBoundsMapping ()
@property (nonatomic, strong) id<ResizableDynamicItem> target;
@end

@implementation PositionToBoundsMapping

- (instancetype)initWithTarget:(id<ResizableDynamicItem>)target
{
    self = [super init];
    if (self)
    {
        _target = target;
    }
    return self;
}

- (CGRect)bounds
{
    return self.target.bounds;
}

- (CGPoint)center
{
    return CGPointMake(self.target.bounds.size.width, self.target.bounds.size.height);
}

- (void)setCenter:(CGPoint)center
{
    self.target.bounds = CGRectMake(0, 0, center.x, center.y);
}

- (CGAffineTransform)transform
{
    return self.target.transform;
}

- (void)setTransform:(CGAffineTransform)transform
{

```

```

        self.target.transform = transform;
    }

@end

```

Schließlich erstellen wir einen `UIViewController`, der eine Schaltfläche enthält. Wenn die Schaltfläche gedrückt wird, erstellen wir `PositionToBoundsMapping` mit der Schaltfläche als umschlossenes dynamisches Element. Wir erstellen ein `UIAttachmentBehavior` an seiner aktuellen Position und fügen dann ein sofortiges `UIPushBehavior`. Da wir jedoch die Grenzen der Änderungen festgelegt haben, bewegt sich die Schaltfläche nicht, sondern wächst und schrumpft.

Schnell

```

final class ViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton(frame: CGRect(x: 0.0, y: 0.0, width: 300.0, height: 200.0))
        button.backgroundColor = .red
        button.layer.cornerRadius = 15.0
        button.setTitle("Tap Me", for: .normal)
        self.view.addSubview(button)
        return button
    }()

    var buttonBounds = CGRect.zero
    var animator: UIDynamicAnimator?

    override func viewDidLoad()
    {
        super.viewDidLoad()
        view.backgroundColor = .white
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:
        .touchUpInside)
        buttonBounds = button.bounds
    }

    override func viewDidLayoutSubviews()
    {
        super.viewDidLayoutSubviews()
        button.center = view.center
    }

    func didPressButton(sender: UIButton)
    {
        // Reset bounds so if button is press twice in a row, previous changes don't propogate
        button.bounds = buttonBounds
        let animator = UIDynamicAnimator(referenceView: view)

        // Create mapping
        let buttonBoundsDynamicItem = PositionToBoundsMapping(target: button)

        // Add Attachment behavior
        let attachmentBehavior = UIAttachmentBehavior(item: buttonBoundsDynamicItem,
        attachedToAnchor: buttonBoundsDynamicItem.center)

        // Higher frequency faster oscillation

```

```

attachmentBehavior.frequency = 2.0

// Lower damping longer oscillation lasts
attachmentBehavior.damping = 0.1
animator.addBehavior(attachmentBehavior)

let pushBehavior = UIPushBehavior(items: [buttonBoundsDynamicItem], mode:
.instantaneous)

// Change angle to determine how much height/ width should change 45° means
height:width is 1:1
pushBehavior.angle = .pi / 4.0

// Larger magnitude means bigger change
pushBehavior.magnitude = 30.0
animator.addBehavior(pushBehavior)
pushBehavior.active = true

// Hold reference so animator is not released
self.animator = animator
}
}

```

Ziel c

```

@interface ViewController ()
@property (nonatomic, strong) UIButton *button;
@property (nonatomic, assign) CGRect buttonBounds;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor whiteColor];
    [self.button addTarget:self action:@selector(didTapButton:)
forControlEvents:UIControlEventTouchUpInside];
    self.buttonBounds = self.button.bounds;
}

- (void)viewDidLoadSubviews
{
    [super viewDidLoadSubviews];
    self.button.center = self.view.center;
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] initWithFrame:CGRectMake(0.0, 0.0, 200.0, 200.0)];
        _button.backgroundColor = [UIColor redColor];
        _button.layer.cornerRadius = 15.0;
        [_button setTitle:@"Tap Me" forState:UIControlStateNormal];
        [self.view addSubview:_button];
    }
    return _button;
}

```

```

}

- (void)didTapButton:(id)sender
{
    self.button.bounds = self.buttonBounds;
    UIDynamicAnimator *animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    PositionToBoundsMapping *buttonBoundsDynamicItem = [[PositionToBoundsMapping
alloc] initWithTarget:sender];
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior
alloc] initWithItem:buttonBoundsDynamicItem attachedToAnchor:buttonBoundsDynamicItem.center];
    [attachmentBehavior setFrequency:2.0];
    [attachmentBehavior setDamping:0.3];
    [animator addBehavior:attachmentBehavior];

    UIPushBehavior *pushBehavior = [[UIPushBehavior alloc]
initWithItems:@[buttonBoundsDynamicItem] mode:UIPushBehaviorModeInstantaneous];
    pushBehavior.angle = M_PI_4;
    pushBehavior.magnitude = 2.0;
    [animator addBehavior:pushBehavior];

    [pushBehavior setActive:TRUE];

    self.animator = animator;
}

@end

```

Weitere Informationen finden Sie im [UIKit Dynamics-Katalog](#)

UIKit Dynamics online lesen: <https://riptutorial.com/de/ios/topic/9479/uikit-dynamics>

Kapitel 170: UIKit Dynamics mit UICollectionView

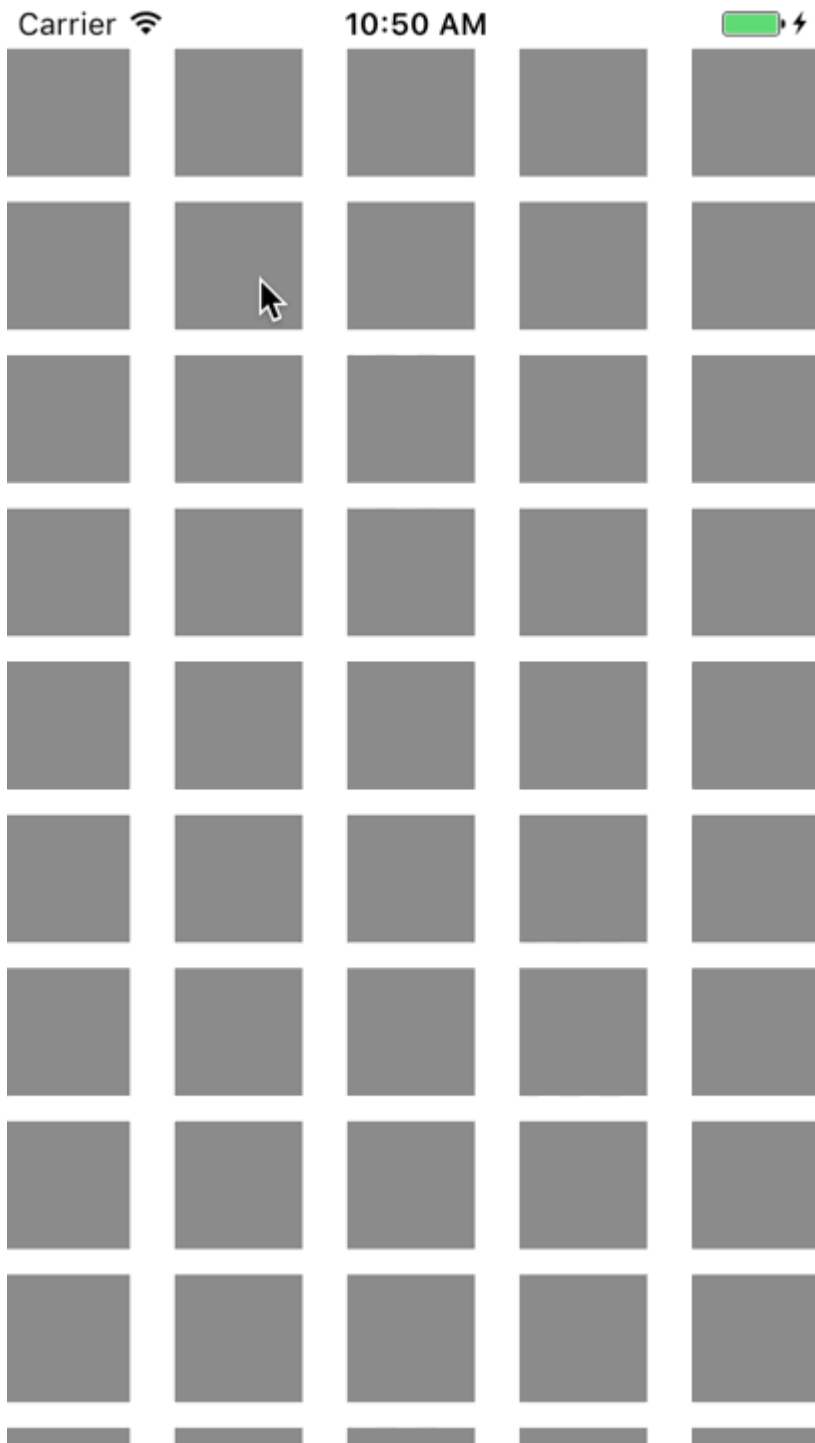
Einführung

UIKit Dynamics ist eine in UIKit integrierte Physik-Engine. UIKit Dynamics bietet eine Reihe von APIs, die Interoperabilität mit einer `UICollectionView` und einem `UICollectionViewLayout`

Examples

Erstellen eines benutzerdefinierten Ziehverhaltens mit `UIDynamicAnimator`

In diesem Beispiel wird `UIDynamicBehavior`, wie ein benutzerdefiniertes `UIDynamicBehavior` durch Unterklassen von `UIDynamicBehavior` und Unterklassen von `UICollectionViewFlowLayout`. In diesem Beispiel haben wir `UICollectionView`, mit dem mehrere Elemente ausgewählt werden können. Mit einer langen `UIDynamicAnimator` können diese Elemente dann in einer elastischen, "federnden" Animation gezogen werden, die von einem `UIDynamicAnimator`.



Das Verschleppungsverhalten wird durch Kombinieren eines Verhaltens auf niedriger Ebene, das den `UIAttachmentBehavior` für die Ecken eines `UIDynamicItem` und eines Verhaltens auf hoher Ebene, das das Verhalten auf niedriger Ebene für eine Reihe von `UIDynamicItems` .

Wir können damit beginnen, dieses Verhalten auf niedriger Ebene zu erstellen. Wir rufen `RectangleAttachmentBehavior`

Schnell

```
final class RectangleAttachmentBehavior: UIDynamicBehavior
{
    init(item: UIDynamicItem, point: CGPoint)
```

```

{
  // Higher frequency more "ridged" formation
  let frequency: CGFloat = 8.0

  // Lower damping longer animation takes to come to rest
  let damping: CGFloat = 0.6

  super.init()

  // Attachment points are four corners of item
  let points = self.attachmentPoints(for: point)

  let attachmentBehaviors: [UIAttachmentBehavior] = points.map
  {
    let attachmentBehavior = UIAttachmentBehavior(item: item, attachedToAnchor: $0)
    attachmentBehavior.frequency = frequency
    attachmentBehavior.damping = damping
    return attachmentBehavior
  }

  attachmentBehaviors.forEach
  {
    addChildBehavior($0)
  }
}

func updateAttachmentLocation(with point: CGPoint)
{
  // Update anchor points to new attachment points
  let points = self.attachmentPoints(for: point)
  let attachments = self.childBehaviors.flatMap { $0 as? UIAttachmentBehavior }
  let pairs = zip(points, attachments)
  pairs.forEach { $0.1.anchorPoint = $0.0 }
}

func attachmentPoints(for point: CGPoint) -> [CGPoint]
{
  // Width and height should be close to the width and height of the item
  let width: CGFloat = 40.0
  let height: CGFloat = 40.0

  let topLeft = CGPoint(x: point.x - width * 0.5, y: point.y - height * 0.5)
  let topRight = CGPoint(x: point.x + width * 0.5, y: point.y - height * 0.5)
  let bottomLeft = CGPoint(x: point.x - width * 0.5, y: point.y + height * 0.5)
  let bottomRight = CGPoint(x: point.x + width * 0.5, y: point.y + height * 0.5)
  let points = [topLeft, topRight, bottomLeft, bottomRight]
  return points
}
}

```

Ziel c

```

@implementation RectangleAttachmentBehavior

- (instancetype) initWithItem: (id<UIDynamicItem>) item point: (CGPoint) point
{
  CGFloat frequency = 8.0f;
  CGFloat damping = 0.6f;
  self = [super init];
}

```

```

if (self)
{
    NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
    for (NSValue *value in pointValues)
    {
        UIAttachmentBehavior *attachment = [[UIAttachmentBehavior alloc] initWithItem:item
attachedToAnchor:[value CGPointValue]];
        attachment.frequency = frequency;
        attachment.damping = damping;
        [self addChildBehavior:attachment];
    }
}
return self;
}

- (void)updateAttachmentLocationWithPoint:(CGPoint)point
{
    NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
    for (NSInteger i = 0; i < pointValues.count; i++)
    {
        NSValue *pointValue = pointValues[i];
        UIAttachmentBehavior *attachment = self.childBehaviors[i];
        attachment.anchorPoint = [pointValue CGPointValue];
    }
}

- (NSArray <NSValue *> *)attachmentPointValuesForPoint:(CGPoint)point
{
    CGFloat width = 40.0f;
    CGFloat height = 40.0f;

    CGPoint topLeft = CGPointMake(point.x - width * 0.5, point.y - height * 0.5);
    CGPoint topRight = CGPointMake(point.x + width * 0.5, point.y - height * 0.5);
    CGPoint bottomLeft = CGPointMake(point.x - width * 0.5, point.y + height * 0.5);
    CGPoint bottomRight = CGPointMake(point.x + width * 0.5, point.y + height * 0.5);

    NSArray <NSValue *> *pointValues = @[[NSValue valueWithCGPoint:topLeft], [NSValue
valueWithCGPoint:topRight], [NSValue valueWithCGPoint:bottomLeft], [NSValue
valueWithCGPoint:bottomRight]];
    return pointValues;
}

@end

```

Als Nächstes können wir das übergeordnete Verhalten erstellen, das eine Reihe von `RectangleAttachmentBehavior` kombiniert.

Schnell

```

final class DragBehavior: UIDynamicBehavior
{
    init(items: [UIDynamicItem], point: CGPoint)
    {
        super.init()
        items.forEach
        {
            let rectAttachment = RectangleAttachmentBehavior(item: $0, point: point)
            self.addChildBehavior(rectAttachment)
        }
    }
}

```

```

    }
}

func updateDragLocation(with point: CGPoint)
{
    // Tell low-level behaviors location has changed
    self.childBehaviors.flatMap { $0 as? RectangleAttachmentBehavior }.forEach {
$0.updateAttachmentLocation(with: point) }
}
}

```

Ziel c

```

@implementation DragBehavior

- (instancetype)initWithItems:(NSArray <id<UIDynamicItem>> *)items point: (CGPoint)point
{
    self = [super init];
    if (self)
    {
        for (id<UIDynamicItem> item in items)
        {
            RectangleAttachmentBehavior *rectAttachment = [[RectangleAttachmentBehavior
alloc] initWithItem:item point:point];
            [self addChildBehavior:rectAttachment];
        }
    }
    return self;
}

- (void)updateDragLocationWithPoint:(CGPoint)point
{
    for (RectangleAttachmentBehavior *rectAttachment in self.childBehaviors)
    {
        [rectAttachment updateAttachmentLocationWithPoint:point];
    }
}

@end

```

Jetzt, da unsere Verhaltensweisen vorhanden sind, müssen Sie sie als Nächstes in unsere Sammlungsansicht aufnehmen. Da wir normalerweise ein Standardraster-Layout wünschen, können wir `UICollectionViewFlowLayout` Unterklasse `UICollectionViewFlowLayout` und nur beim Ziehen Attribute ändern. Dies `layoutAttributesForElementsInRect` hauptsächlich durch das Überschreiben von `layoutAttributesForElementsInRect` und die Verwendung der `UIDynamicAnimator's` Convenience-Methode `itemsInRect` .

Schnell

```

final class DraggableLayout: UICollectionViewFlowLayout
{
    // Array that holds dragged index paths
    var indexPathsForDraggingElements: [IndexPath]?
}

```

```

// The dynamic animator that will animate drag behavior
var animator: UIDynamicAnimator?

// Custom high-level behavior that dictates drag animation
var dragBehavior: DragBehavior?

// Where dragging starts so can return there once dragging ends
var startDragPoint = CGPoint.zero

// Bool to keep track if dragging has ended
var isFinishedDragging = false

// Method to inform layout that dragging has started
func startDragging(indexPaths selectedIndexPaths: [IndexPath], from point: CGPoint)
{
    indexPathsForDraggingElements = selectedIndexPaths
    animator = UIDynamicAnimator(collectionViewLayout: self)
    animator?.delegate = self

    // Get all of the draggable attributes but change zIndex so above other cells
    let draggableAttributes: [UICollectionViewLayoutAttributes] =
selectedIndexPaths.flatMap {
        let attribute = super.layoutAttributesForItem(at: $0)
        attribute?.zIndex = 1
        return attribute
    }

    startDragPoint = point

    // Add them to high-level behavior
    dragBehavior = DragBehavior(items: draggableAttributes, point: point)

    // Add high-level behavior to animator
    animator?.addBehavior(dragBehavior!)
}

func updateDragLocation(_ point: CGPoint)
{
    // Tell high-level behavior that point has updated
    dragBehavior?.updateDragLocation(with: point)
}

func endDragging()
{
    isFinishedDragging = true

    // Return high-level behavior to starting point
    dragBehavior?.updateDragLocation(with: startDragPoint)
}

func clearDraggedIndexPaths()
{
    // Reset state for next drag event
    animator = nil
    indexPathsForDraggingElements = nil
    isFinishedDragging = false
}

override func layoutAttributesForElements(in rect: CGRect) ->
[UICollectionViewLayoutAttributes]?

```

```

    {
        let existingAttributes: [UICollectionViewLayoutAttributes] =
super.layoutAttributesForElements(in: rect) ?? []
        var allAttributes = [UICollectionViewLayoutAttributes]()

        // Get normal flow layout attributes for non-drag items
        for attributes in existingAttributes
        {
            if (indexPathsForDraggingElements?.contains(attributes.indexPath) ?? false) ==
false
                {
                    allAttributes.append(attributes)
                }
        }

        // Add dragged item attributes by asking animator for them
        if let animator = self.animator
        {
            let animatorAttributes: [UICollectionViewLayoutAttributes] = animator.items(in:
rect).flatMap { $0 as? UICollectionViewLayoutAttributes }
            allAttributes.append(contentsOf: animatorAttributes)
        }
        return allAttributes
    }
}
extension DraggableLayout: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        // Animator has paused and done dragging; reset state
        guard isFinishedDragging else { return }
        clearDraggedIndexPaths()
    }
}
}

```

Ziel c

```

@interface DraggableLayout () <UIDynamicAnimatorDelegate>
@property (nonatomic, strong) NSArray <NSIndexPath *> *indexPathsForDraggingElements;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@property (nonatomic, assign) CGPoint startDragPoint;
@property (nonatomic, assign) BOOL finishedDragging;
@property (nonatomic, strong) DragBehavior *dragBehavior;
@end

@implementation DraggableLayout

- (void)startDraggingWithIndexPaths:(NSArray <NSIndexPath *> *)selectedIndexPaths
fromPoint:(CGPoint)point
{
    self.indexPathsForDraggingElements = selectedIndexPaths;
    self.animator = [[UIDynamicAnimator alloc] initWithCollectionViewLayout:self];
    self.animator.delegate = self;
    NSMutableArray *draggableAttributes = [[NSMutableArray
alloc] initWithCapacity:selectedIndexPaths.count];
    for (NSIndexPath *indexPath in selectedIndexPaths)
    {
        UICollectionViewLayoutAttributes *attributes = [super
layoutAttributesForItemAtIndexPath:indexPath];
    }
}

```

```

        attributes.zIndex = 1;
        [draggableAttributes addObject:attributes];
    }
    self.startDragPoint = point;
    self.dragBehavior = [[DragBehavior alloc] initWithItems:draggableAttributes point:point];
    [self.animator addBehavior:self.dragBehavior];
}

- (void)updateDragLoactionWithPoint:(CGPoint)point
{
    [self.dragBehavior updateDragLocationWithPoint:point];
}

- (void)endDragging
{
    self.finishedDragging = YES;
    [self.dragBehavior updateDragLocationWithPoint:self.startDragPoint];
}

- (void)clearDraggedIndexPath
{
    self.animator = nil;
    self.indexPathsForDraggingElements = nil;
    self.finishedDragging = NO;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    if (self.finishedDragging)
    {
        [self clearDraggedIndexPath];
    }
}

- (NSArray<UICollectionViewLayoutAttributes *>
*)layoutAttributesForElementsInRect:(CGRect)rect
{
    NSArray *existingAttributes = [super layoutAttributesForElementsInRect:rect];
    NSMutableArray *allAttributes = [[NSMutableArray
alloc] initWithCapacity:existingAttributes.count];
    for (UICollectionViewLayoutAttributes *attributes in existingAttributes)
    {
        if (![self.indexPathsForDraggingElements containsObject:attributes.indexPath])
        {
            [allAttributes addObject:attributes];
        }
    }
    [allAttributes addObjectsFromArray:[self.animator itemsInRect:rect]];
    return allAttributes;
}

@end

```

Zum Schluss erstellen wir einen Ansichts-Controller, der unsere `UICollectionView` und unsere lange `UICollectionView` .

Schnell

```

final class ViewController: UIViewController
{
    // Collection view that displays cells
    lazy var collectionView: UICollectionView =
    {
        let collectionView = UICollectionView(frame: .zero, collectionViewLayout:
DraggableLayout())
        collectionView.backgroundColor = .white
        collectionView.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(collectionView)
        collectionView.topAnchor.constraint(equalTo:
self.topAnchor).isActive = true
        collectionView.leadingAnchor.constraint(equalTo: self.view.leadingAnchor).isActive =
true
        collectionView.trailingAnchor.constraint(equalTo: self.view.trailingAnchor).isActive =
true
        collectionView.bottomAnchor.constraint(equalTo:
self.bottomAnchor).isActive = true

        return collectionView
    }()

    // Gesture that drives dragging
    lazy var longPress: UILongPressGestureRecognizer =
    {
        let longPress = UILongPressGestureRecognizer(target: self, action:
#selector(self.handleLongPress(sender:)))
        return longPress
    }()

    // Array that holds selected index paths
    var selectedIndexPaths = [IndexPath]()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        collectionView.delegate = self
        collectionView.dataSource = self
        collectionView.register(UICollectionViewCell.self, forCellWithReuseIdentifier: "Cell")
        collectionView.addGestureRecognizer(longPress)
    }

    func handleLongPress(sender: UILongPressGestureRecognizer)
    {
        guard let draggableLayout = collectionView.collectionViewLayout as? DraggableLayout
else { return }
        let location = sender.location(in: collectionView)
        switch sender.state
        {
            case .began:
                draggableLayout.startDragging(indexPaths: selectedIndexPaths, from: location)
            case .changed:
                draggableLayout.updateDragLocation(location)
            case .ended, .failed, .cancelled:
                draggableLayout.endDragging()
            case .possible:
                break
        }
    }
}

extension ViewController: UICollectionViewDelegate, UICollectionViewDataSource

```



```

{
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section:
Int) -> Int
    {
        return 1000
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell
    {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "Cell", for:
indexPath)
        cell.backgroundColor = .gray
        if selectedIndexPaths.contains(indexPath) == true
        {
            cell.backgroundColor = .red
        }
        return cell
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath)
    {
        // Bool that determines if cell is being selected or unselected
        let isSelected = !selectedIndexPaths.contains(indexPath)
        let cell = collectionView.cellForItem(at: indexPath)
        cell?.backgroundColor = isSelected ? .red : .gray
        if isSelected
        {
            selectedIndexPaths.append(indexPath)
        }
        else
        {
            selectedIndexPaths.remove(at: selectedIndexPaths.index(of: indexPath!))
        }
    }
}
}

```

Ziel c

```

@interface ViewController () <UICollectionViewDelegate, UICollectionViewDataSource>
@property (nonatomic, strong) UICollectionView *collectionView;
@property (nonatomic, strong) UILongPressGestureRecognizer *longPress;
@property (nonatomic, strong) NSMutableArray <NSIndexPath *> *selectedIndexPaths;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.collectionView.delegate = self;
    self.collectionView.dataSource = self;
    [self.collectionView registerClass:[UICollectionViewCell class]
forCellWithReuseIdentifier:@"Cell"];
    [self.collectionView addGestureRecognizer:self.longPress];
    self.selectedIndexPaths = [[NSMutableArray alloc] init];
}

```

```

- (UICollectionView *)collectionView
{
    if (!_collectionView)
    {
        _collectionView = [[UICollectionView alloc] initWithFrame:CGRectZero
collectionViewLayout:[[DraggableLayout alloc]init]];
        _collectionView.backgroundColor = [UIColor whiteColor];
        _collectionView.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_collectionView];
        [_collectionView.topAnchor
constraintEqualToAnchor:self.topLayoutGuide.bottomAnchor].active = YES;
        [_collectionView.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor].active
= YES;
        [_collectionView.trailingAnchor
constraintEqualToAnchor:self.view.trailingAnchor].active = YES;
        [_collectionView.bottomAnchor
constraintEqualToAnchor:self.bottomLayoutGuide.topAnchor].active = YES;
    }
    return _collectionView;
}

- (UILongPressGestureRecognizer *)longPress
{
    if (!_longPress)
    {
        _longPress = [[UILongPressGestureRecognizer alloc] initWithTarget:self
action:@selector(handleLongPress:)];
    }
    return _longPress;
}

- (void)handleLongPress:(UILongPressGestureRecognizer *)sender
{
    DraggableLayout *draggableLayout = (DraggableLayout
*)self.collectionView.collectionViewLayout;
    CGPoint location = [sender locationInView:self.collectionView];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        [draggableLayout startDraggingWithIndexPaths:self.selectedIndexPaths
fromPoint:location];
    }
    else if(sender.state == UIGestureRecognizerStateChanged)
    {
        [draggableLayout updateDragLoactionWithPoint:location];
    }
    else if(sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled || sender.state == UIGestureRecognizerStateFailed)
    {
        [draggableLayout endDragging];
    }
}

- (NSInteger)collectionView:(UICollectionView *)collectionView
numberOfItemsInSection:(NSInteger)section
{
    return 1000;
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
cellForItemAtIndexPath:(NSIndexPath *)indexPath
{

```

```

    UICollectionViewCell *cell = [collectionView
dequeueReusableCellWithIdentifier:@"Cell" forIndexPath:indexPath];
    cell.backgroundColor = [UIColor grayColor];
    if ([self.selectedIndexPaths containsObject:indexPath])
    {
        cell.backgroundColor = [UIColor redColor];
    }
    return cell;
}

- (void)collectionView:(UICollectionView *)collectionView
didSelectItemAtIndexPath:(NSIndexPath *)indexPath
{
    BOOL isSelected = ![self.selectedIndexPaths containsObject:indexPath];
    UICollectionViewCell *cell = [collectionView cellForItemAtIndexPath:indexPath];
    if (isSelected)
    {
        cell.backgroundColor = [UIColor redColor];
        [self.selectedIndexPaths addObject:indexPath];
    }
    else
    {
        cell.backgroundColor = [UIColor grayColor];
        [self.selectedIndexPaths removeObject:indexPath];
    }
}

@end

```

Weitere Informationen zur [WWDC-Sitzung 2013 "Fortgeschrittene Techniken mit UIKit Dynamics"](#)

[UIKit Dynamics mit UICollectionView online lesen: https://riptutorial.com/de/ios/topic/10079/uikit-dynamics-mit-uicollectionview](https://riptutorial.com/de/ios/topic/10079/uikit-dynamics-mit-uicollectionview)

Kapitel 171: UILabel

Einführung

Die UILabel-Klasse implementiert eine schreibgeschützte Textansicht. Sie können diese Klasse verwenden, um eine oder mehrere Zeilen statischen Textes zu zeichnen, z. B. solche, mit denen Sie andere Teile Ihrer Benutzeroberfläche identifizieren können. Die UILabel-Basisklasse unterstützt sowohl die einfache als auch die komplexe Gestaltung des Etikettentextes. Sie können auch Aspekte des Erscheinungsbilds steuern, z. B. ob die Beschriftung einen Schatten verwendet oder mit einer Hervorhebung zeichnet. Bei Bedarf können Sie das Erscheinungsbild Ihres Textes durch Unterklassen weiter anpassen.

Syntax

- `UILabel.numberOfLines`: `Int` // Die maximale Anzahl von Zeilen für das Label ermitteln oder festlegen. 0 ist unbegrenzt
- `UILabel.text`: Zeichenfolge? // Erhalte oder setze den Text, den das Label anzeigt
- `UILabel.textColor`: `UIColor!` // Die Farbe des Texts auf dem Etikett abrufen oder festlegen
- `UILabel.tintColor`: `UIColor!` // Die Farbtonfarbe des Etiketts abrufen oder festlegen
- `UILabel.attributedText`: `NSAttributedString?` // den attributierten Text der Beschriftung abrufen oder festlegen
- `UILabel.font`: `UIFont!` // Die Schrift des Textes auf dem Etikett abrufen oder festlegen
- `UILabel.textAlignment`: `NSTextAlignment` // Ermittelt oder setzt die Ausrichtung des Textes

Bemerkungen

UILabels sind Ansichten, mit denen eine oder mehrere Textzeilen angezeigt werden können. Es enthält mehrere Möglichkeiten zum Stilisieren von Text, z. B. Schatten, Textfarben und Schriftarten.

UILabels können auch Attributed Strings anzeigen, dh Text + Inline-Markup, um Stile auf Teile des Textes anzuwenden.

UILabel entspricht nicht dem UIAppearance-Protokoll. Daher können Sie keine UIAppearance-Proxy-Methoden verwenden, um das Erscheinungsbild von UILabels anzupassen. Sehen Sie diese [Diskussion](#) für mehr.

Apple-Entwicklerreferenz [hier](#)

Examples

Text in einem vorhandenen Etikett ändern

Ändern Sie den Text eines vorhandenen `UILabel` können durch den Zugriff auf und die Änderung

der getan werden `text` - Eigenschaft des `UILabel` . Dies kann direkt mit `String` Literalen oder indirekt mit Variablen erfolgen.

Festlegen des Textes mit `String` Literalen

Schnell

```
label.text = "the new text"
```

Ziel c

```
// Dot Notation
label.text = @"the new text";

// Message Pattern
[label setText:@"the new text"];
```

Den Text mit einer Variablen einstellen

Schnell

```
let stringVar = "basic String var"
label.text = stringVar
```

Ziel c

```
NSString * stringVar = @"basic String var";

// Dot Notation
label.text = stringVar;

// Message Pattern
[label setText: stringVar];
```

Textfarbe

Sie können die Eigenschaft `textColor` der Beschriftung `textColor` , um eine `textColor` auf den gesamten Text der Beschriftung anzuwenden.

Schnell

```
label.textColor = UIColor.redColor()
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

Swift 3

```
label.textColor = UIColor.red
```

```
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

Ziel c

```
label.textColor = [UIColor redColor];  
label.textColor = [UIColor colorWithRed:64.0f/255.0f green:88.0f/255.0f blue:41.0f/255.0f  
alpha:1.0f];
```

Textfarbe auf einen Teil des Textes anwenden

Sie können auch die `NSAttributedString` (oder andere Attribute) von Textabschnitten mithilfe von `NSAttributedString`:

Ziel c

```
attributedString = [[NSMutableAttributedString alloc] initWithString:@"The grass is green; the  
sky is blue."];  
[attributedString addAttribute: NSForegroundColorAttributeName value:[UIColor greenColor]  
range:NSMakeRange(13, 5)];  
[attributedString addAttribute: NSForegroundColorAttributeName value:[UIColor blueColor]  
range:NSMakeRange(31, 4)];  
label.attributedString = attributedString;
```

Schnell

```
let attributedString = NSMutableAttributedString(string: "The grass is green; the sky is  
blue.")  
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.green(), range:  
NSRange(location: 13, length: 5))  
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue(), range:  
NSRange(location: 31, length: 4))  
label.attributedString = attributedString
```

Textausrichtung

Schnell

```
label.textAlignment = NSTextAlignment.left  
//or the shorter  
label.textAlignment = .left
```

Jeder Wert in der `NSTextAlignment` ist gültig: `.left`, `.center`, `.right`, `.justified`, `.natural`

Ziel c

```
label.textAlignment = NSTextAlignmentLeft;
```

Jeder Wert in der `NSTextAlignment` ist gültig: `NSTextAlignmentLeft`, `NSTextAlignmentCenter`, `NSTextAlignmentRight`, `NSTextAlignmentJustified`, `NSTextAlignmentNatural`

Die vertikale Ausrichtung in `UILabel` wird nicht `UILabel` unterstützt: [Text in UILabel vertikal nach oben ausrichten](#)

Erstellen Sie ein `UILabel`

Mit einem Rahmen

Wenn Sie die genauen Abmessungen kennen, die Sie für Ihr Etikett festlegen möchten, können Sie ein `UILabel` mit einem `CGRect` Rahmen initialisieren.

Schnell

```
let frame = CGRect(x: 0, y: 0, width: 200, height: 21)
let label = UILabel(frame: frame)
view.addSubview(label)
```

Ziel c

```
CGRect frame = CGRectMake(0, 0, 200, 21);
UILabel *label = [[UILabel alloc] initWithFrame:frame];
[view addSubview:label];
```

Mit automatischem Layout

Sie können Einschränkungen für ein `UILabel` wenn iOS seinen Frame zur Laufzeit dynamisch berechnen soll.

Schnell

```
let label = UILabel()
label.backgroundColor = .red
label.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(label)

NSLayoutConstraint.activate([
    //stick the top of the label to the top of its superview:
    label.topAnchor.constraint(equalTo: view.topAnchor)

    //stick the left of the label to the left of its superview
    //if the alphabet is left-to-right, or to the right of its
    //superview if the alphabet is right-to-left:
    label.leadingAnchor.constraint(equalTo: view.leadingAnchor)

    //stick the label's bottom to the bottom of its superview:
    label.bottomAnchor.constraint(equalTo: view.bottomAnchor)

    //the label's width should be equal to 100 points:
```

```
label.widthAnchor.constraint(equalToConstant: 100)
])
```

Ziel c

```
UILabel *label = [[UILabel alloc] init];
```

Mit Objective-c + Visual Format Language (VFL)

```
UILabel *label = [UILabel new];
label.translatesAutoresizingMaskIntoConstraints = NO;
[self.view addSubview label];
// add horizontal constraints with 5 left and right padding from the leading and trailing

[self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:|-5-
[labelName]-5-|"
                                options:0
                                metrics:nil

views:@{@"labelName":label}}];
// vertical constraints that will use the height of the superView with no padding on top and
bottom
[self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"H:|[labelName]|"
                                options:0
                                metrics:nil

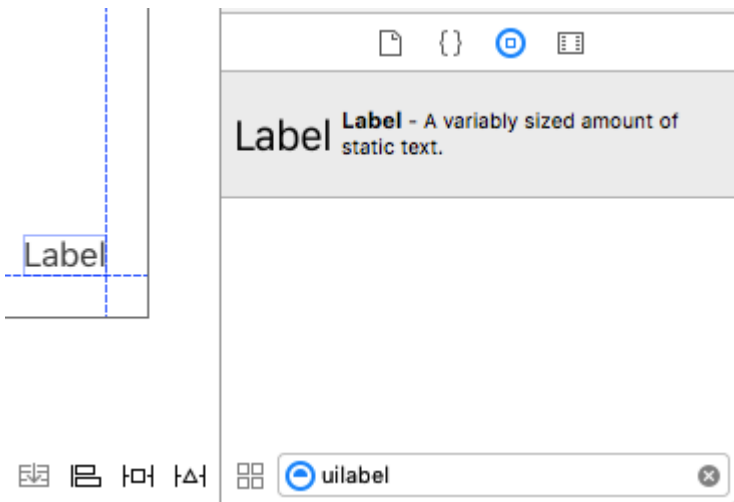
views:@{@"labelName":label}}]
```

Die VFL-Dokumentation finden Sie [hier](#)

Stellen Sie nach dem Erstellen des Etiketts sicher, dass Sie die Abmessungen über das automatische Layout festlegen. Xcode zeigt Fehler an, wenn dies nicht ordnungsgemäß durchgeführt wird.

Mit dem Interface Builder

Sie können den Interface Builder auch verwenden, um der `Storyboard` oder `.xib` Datei ein `UILabel` hinzuzufügen, indem Sie ein `Label` aus dem `.xib` ziehen und in eine Ansicht in der Leinwand ziehen:

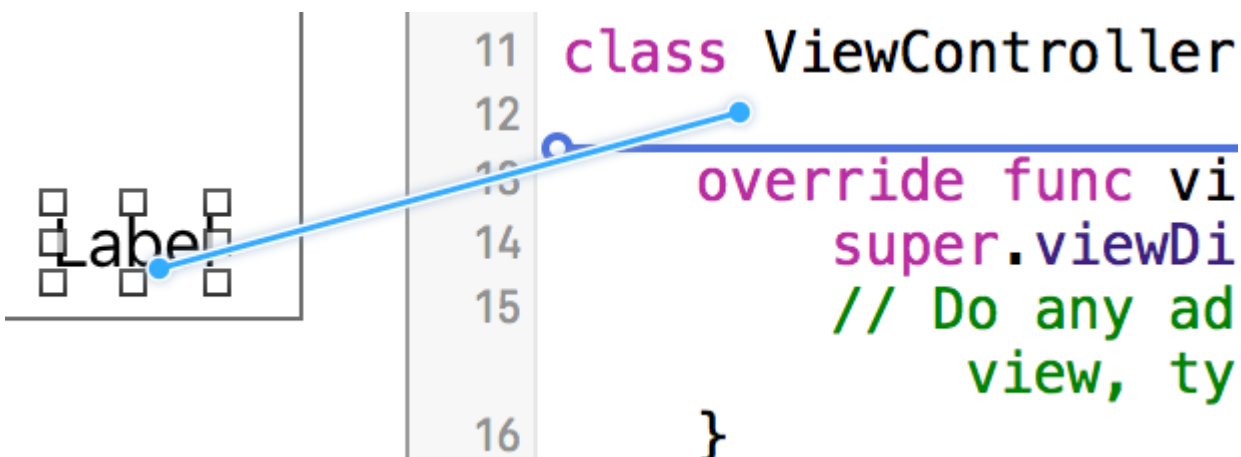


Anstatt einen Frame (Position und Größe) für ein `UILabel` programmgesteuert `.xib` können Sie mit einem `Storyboard` oder einer `.xib` mithilfe des **automatischen Layouts** Einschränkungen für das Steuerelement hinzufügen.

Um auf dieses aus `storyboard` oder `xib` erstellte Label `xib` erstellen Sie ein `IBOutlet` dieses Labels.

Verknüpfung zwischen Interface Builder und View Controller

Nachdem Sie ein `UILabel` zu Ihrem `Storyboard` oder der `.xib` Datei hinzugefügt haben, können Sie es mit Ihrem Code verknüpfen, indem Sie die `UILabel` `ViewController` `Control` ^ drücken und dann die Maus zwischen das `UILabel` zu Ihrem `ViewController` , oder Sie können in den Code ziehen, während Sie mit der rechten Maustaste darauf klicken haben die gleiche Wirkung.



Im Eigenschaftendialogfeld können Sie den Namen von `UILabel` und als `strong` oder `weak UILabel` . Weitere Informationen zu `strong` und `weak` finden Sie [hier](#) .

Die andere Möglichkeit besteht darin, den Ausgang wie folgt programmgesteuert zu machen:

Schnell

```
@IBOutlet weak var nameLabel : UILabel!
```

Ziel c

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;
```

Schriftart einstellen

Schnell

```
let label = UILabel()
```

Ziel c

```
UILabel *label = [[UILabel alloc] init];  
or  
UILabel *label = [UILabel new]; // convenience method for calling alloc-init
```

Ändern Sie die Größe der Standardschriftart

Schnell

```
label.font = UIFont.systemFontOfSize(17)
```

Swift 3

```
label.font = UIFont.systemFont(ofSize: 17)
```

Ziel c

```
label.font = [UIFont systemFontOfSize:17];
```

Verwenden Sie eine bestimmte Schriftstärke

iOS 8.2

Schnell

```
label.font = UIFont.systemFont(ofSize:17, weight: UIFontWeightBold)
```

Swift3

```
label.font = UIFont.systemFont(ofSize: 17, weight: UIFontWeightBold)
```

Ziel c

```
label.font = [UIFont systemFontOfSize:17 weight:UIFontWeightBold];
```

iOS 8.2

Schnell

```
label.font = UIFont.boldSystemFont(ofSize(17)
```

Swift3

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

Ziel c

```
label.font = [UIFont boldSystemFontOfSize:17];
```

Verwenden Sie einen dynamischen Textstil.

Die Schriftart und die Punktgröße basieren auf der bevorzugten Lesegröße des Benutzers.

Schnell

```
label.font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
```

Swift 3

```
label.font = UIFont.preferredFont(forTextStyle: .body)
```

Ziel c

```
label.font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Verwenden Sie eine andere Schriftart

Schnell

```
label.font = UIFont(name: "Avenir", size: 15)
```

Ziel c

```
label.font = [UIFont fontWithName:@"Avenir" size:15];
```

Überschreibt die Schriftgröße

Sie können die Schriftgröße ohne Kenntnis der Schriftfamilie `UILabel` die **font** `UILabel` des `UILabel` .

Schnell

```
label.font = label.font.fontWithSize(15)
```

Swift 3

```
label.font = label.font.withSize(15)
```

Ziel c

```
label.font = [label.font fontWithSize:15];
```

Verwenden Sie Custom Font Swift

[Siehe diesen Link](#)

Anzahl der Zeilen

Wenn Sie eine Beschriftung erstellen und festlegen, dass der Text mehr als eine einzelne Zeile ist, die angezeigt werden kann, wird sie abgeschnitten und es wird nur eine Textzeile angezeigt, die mit drei Punkten endet (...). Dies liegt daran, dass eine Eigenschaft namens `numberOfLines` auf 1 festgelegt ist und daher nur eine Zeile angezeigt wird. Es ist ein häufiger Fehler im Umgang mit

`UILabel`, und viele Leute halten es für einen Fehler oder sie verwenden mehr als ein Label, um mehr als nur eine Textzeile `UILabel` jedoch nur diese Eigenschaft bearbeiten, können Sie einem `UILabel` mitteilen akzeptieren Sie die angegebene Anzahl von Zeilen. Wenn diese Eigenschaft beispielsweise auf 5 eingestellt ist, kann das Etikett 1, 2, 3, 4 oder 5 Datenzeilen anzeigen.

Den Wert programmgesteuert einstellen

Um diese Eigenschaft festzulegen, weisen Sie ihr einfach eine neue Ganzzahl zu:

Schnell

```
label.numberOfLines = 2
```

Ziel c

```
label.numberOfLines = 2;
```

Hinweis

Es ist möglich, diese Eigenschaft auf 0 zu setzen. Dies bedeutet jedoch nicht, dass keine Zeilen akzeptiert werden, sondern dass die Beschriftung so viele Zeilen enthalten kann, wie sie benötigt werden (auch als "Infinity" bezeichnet):

Schnell

```
label.numberOfLines = 0
```

Ziel c

```
label.numberOfLines = 0;
```

Hinweis

Wenn das Etikett eine Höhenbeschränkung hat, wird die Einschränkung beachtet. In diesem Fall `label.numberOfLines = 0` möglicherweise nicht wie erwartet.

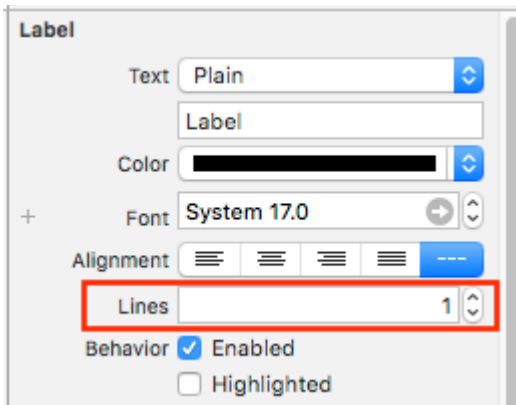
Hinweis

Für komplexeren mehrzeiligen Text ist `UITextView` möglicherweise besser geeignet. *

Wert im Interface Builder einstellen

Anstatt `numberOfLines` programmgesteuert `numberOfLines`, können Sie ein Storyboard oder eine `.xib` und die `numberOfLines` Eigenschaft `numberOfLines`. Auf diese Weise erzielen wir dieselben Ergebnisse wie der obige Code.

Wie unten:



Größe passend

Angenommen, Sie haben ein `UILabel` in Ihrem storyboard und Sie haben dafür ein `IBOutlet` in `ViewController.swift / ViewController.m` und als `labelOne`.

Um die Änderungen leicht sichtbar zu machen, ändern Sie `backgroundColor` und `textColor` von `labelOne` in der `viewDidLoad` Methode:

Die Funktion `sizeToFit` wird verwendet, wenn Sie die Größe eines Labels basierend auf dem darin gespeicherten Inhalt automatisch ändern möchten.

Schnell

```
labelOne.backgroundColor = UIColor.blueColor()
labelOne.textColor = UIColor.whiteColor()
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

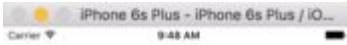
Swift 3

```
labelOne.backgroundColor = UIColor.blue
labelOne.textColor = UIColor.white
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

Ziel c

```
labelOne.backgroundColor = [UIColor blueColor];
labelOne.textColor = [UIColor whiteColor];
labelOne.text = @"Hello, World!";
[labelOne sizeToFit];
```

Die Ausgabe für den obigen Code lautet:



Wie Sie sehen, ändert sich nichts, da der Text in labelOne perfekt passt. sizeToFit ändert nur den Rahmen des Labels.

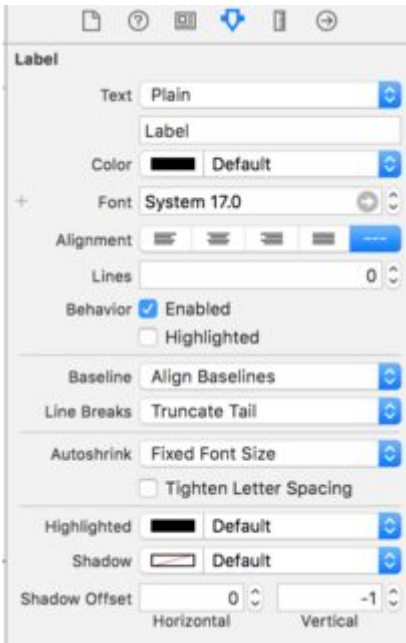
Lassen Sie uns den Text etwas länger ändern:

```
labelOne.text = "Hello, World! I'm glad to be alive!"
```

Nun sieht labelOne so aus:



Selbst `sizeToFit` aufrufen, ändert sich nichts. Dies ist darauf zurückzuführen, dass die vom UILabel angezeigten `numberOfLines` standardmäßig auf 1 gesetzt ist. Lassen Sie sie im Storyboard auf Null setzen:



Wenn Sie diesmal die App ausführen, wird `labelOne` korrekt angezeigt:



Die `numberOfLines` Eigenschaft kann auch in der `ViewController` Datei `ViewController` werden:

```
// Objective-C
labelOne.numberOfLines = 0;

// Swift
labelOne.numberOfLines = 0
```

Hintergrundfarbe

Schnell

```
label.backgroundColor = UIColor.redColor()

label.backgroundColor = .redColor()
```

Swift 3

```
label.backgroundColor = UIColor.red
```

Ziel c

```
label.backgroundColor = [UIColor redColor];
```

Fügen Sie dem Text Schatten hinzu

Schnell

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Swift 3

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Ziel c

```
label1.layer.shadowOffset = CGSizeMake(3, 3);
label1.layer.shadowOpacity = 0.7;
label1.layer.shadowRadius = 2;
```

I Like My Cat

Variable Höhe mit Einschränkungen

Sie können ein `UILabel` mit dynamischer Höhe mithilfe des automatischen Layouts `UILabel` .

Sie müssen `numberOfLines` auf Null (0) setzen und eine minimale Höhe hinzufügen, indem Sie Einschränkungen mit einer Beziehung vom Typ `.GreaterThanOrEqual` für das Attribut

`.GreaterThanOrEqual .Height`

iOS 6

Schnell

```
label.numberOfLines = 0

let heightConstraint = NSLayoutConstraint(
    item: label,
    attribute: .Height,
    relatedBy: .GreaterThanOrEqual,
    toItem: nil,
    attribute: .NotAnAttribute,
    multiplier: 0,
    constant: 20
)

label.addConstraint(heightConstraint)
```

iOS 9

Schnell

```
label.numberOfLines = 0
label.translatesAutoresizingMaskIntoConstraints = false
label.heightAnchor.constraintGreaterThanOrEqualToConstant(20).active = true
```

LineBreakMode

Code verwenden

```
UILabel.lineBreakMode: NSLineBreakMode
```

Schnell

```
label.lineBreakMode = .ByTruncatingTail
```

- .ByWordWrapping
- .ByCharWrapping
- .ByClipping
- .ByTruncatingHead
- .ByTruncatingTail
- .ByTruncatingMiddle

Swift 3

```
label.lineBreakMode = .byTruncatingTail
```

- .byWordWrapping
- .byCharWrapping
- .byClipping
- .byTruncatingHead
- .byTruncatingTail
- .byTruncatingMiddle

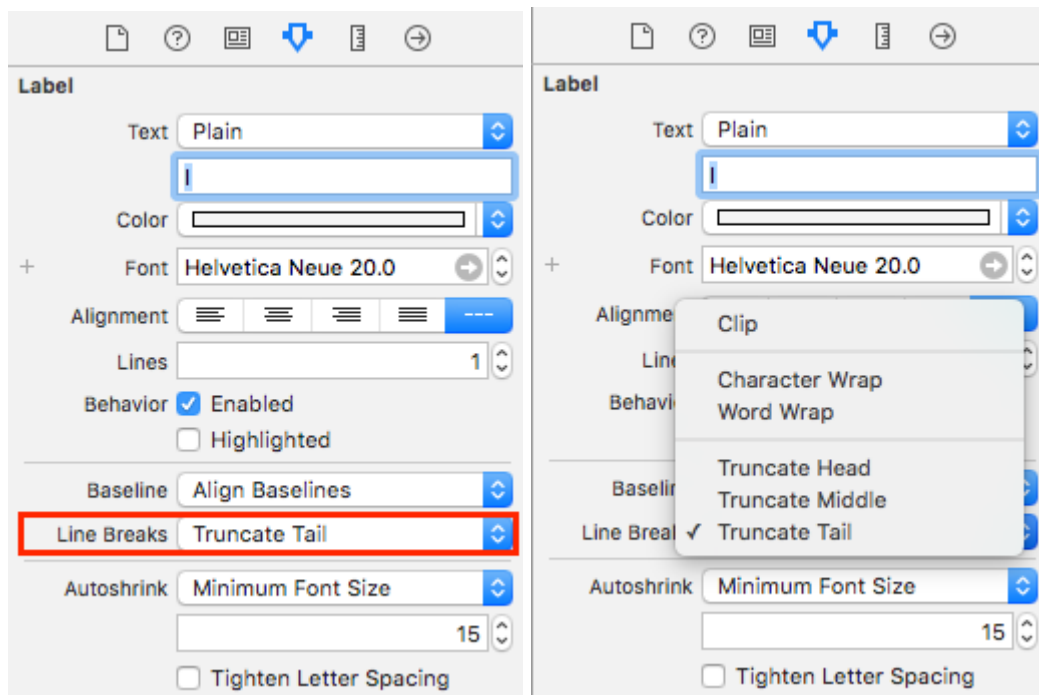
Ziel c

```
[label setLineBreakMode:NSLineBreakByTruncatingTail];
```

- NSLineBreakByWordWrapping
- NSLineBreakByCharWrapping
- NSLineBreakByClipping
- NSLineBreakByTruncatingHead
- NSLineBreakByTruncatingTail
- NSLineBreakByTruncatingMiddle

Storyboard verwenden

Dies kann auch im Attribut-Inspector eines UILabels festgelegt werden:



Konstanten

- Zeilenumbruch - Der Zeilenumbruch erfolgt an Wortgrenzen, es sei denn, das Wort selbst passt nicht in eine einzige Zeile
- Zeichenbruch - Der Umbruch erfolgt vor dem ersten Zeichen, das nicht passt

- Ausschnittlinien werden einfach nicht über den Rand des Textcontainers gezogen
- Kopf abschneiden - Die Linie wird so angezeigt, dass das Ende in den Container passt und der fehlende Text am Anfang der Zeile durch ein Ellipsenzeichen angezeigt wird
- Schwanz abschneiden - Die Linie wird so angezeigt, dass der Anfang in den Container passt und der fehlende Text am Ende der Zeile durch ein Ellipsenzeichen angezeigt wird
- Mitte abschneiden - Die Linie wird so angezeigt, dass Anfang und Ende in den Container passen und der fehlende Text in der Mitte durch ein Ellipsenzeichen angezeigt wird

Content Bounds berechnen (zB für dynamische Zellenhöhen)

Ein üblicher Anwendungsfall für die Berechnung des Rahmens, den ein Label beansprucht, ist die geeignete Größenanpassung von Tabellenzellen. Die empfohlene Methode hierfür ist die Verwendung der `NSString` Methode `boundingRectWithSize:options:attributes:context:`

`options` nimmt String-Zeichenoptionen:

- `NSStringDrawingUsesLineFragmentOrigin` sollte für Beschriftungen mit mehreren Zeilen verwendet werden
- `NSStringDrawingTruncatesLastVisibleLine` sollte mit `|` hinzugefügt werden Operator, wenn eine maximale Anzahl von Zeilen vorhanden ist

`attributes` ist ein `NSDictionary` von Attributen, die zugeordnete Zeichenfolgen `NSDictionary` (vollständige Liste: [Apple Docs](#)), aber die Faktoren, die die Höhe beeinflussen, umfassen:

- **NSFontAttributeName** : Sehr wichtig, die Größe und Schriftfamilie ist ein wichtiger Teil der angezeigten Größe des Etiketts.
- **NSParagraphStyleAttributeName** : Zum Anpassen der Anzeige des Texts. Dazu gehören Zeilenabstand, Textausrichtung, Schnittstil und einige andere Optionen. Wenn Sie keinen dieser Werte explizit geändert haben, sollten Sie sich nicht so viele Sorgen machen, aber es kann wichtig sein, wenn Sie einige Werte für IB ändern.

`context` sollte gleich `nil` da der primäre Anwendungsfall von `NSStringDrawingContext` dazu dient, die Schriftgröße an ein angegebenes `NSStringDrawingContext` , was bei der Berechnung einer dynamischen Höhe nicht der Fall sein sollte.

Ziel c

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexPath];

    NSString *labelContent = cell.textLabel.text;
    // you may choose to get the content directly from the data source if you have done
    minimal customizations to the font or are comfortable with hardcoding a few values
    //    NSString *labelContent = [self.dataSource objectAtIndex:indexPath:indexPath];

    // value may be hardcoded if retrieved from data source
    NSFont *labelFont = [cell.textLabel font];
```

```

// The NSParagraphStyle, even if you did not code any changes these values may have been
altered in IB
NSMutableParagraphStyle *paragraphStyle = [NSMutableParagraphStyle new];
paragraphStyle.lineBreakMode = NSLineBreakByWordWrapping;
paragraphStyle.alignment = NSTextAlignmentCenter;

NSDictionary *attributes = @{@"NSFontAttributeName: labelFont,
                             NSParagraphStyleAttributeName: paragraphStyle};

// The width is also important to the height
CGFloat labelWidth = CGRectGetWidth(cell.theLabel.frame);
// If you have been hardcoding up to this point you will be able to get this value by
subtracting the padding on left and right from tableView.bounds.size.width
//   CGFloat labelWidth = CGRectGetWidth(tableView.frame) - 20.0f - 20.0f;

CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, CGFLOAT_MAX)
options:NSStringDrawingUsesLineFragmentOrigin attributes:attributes context:nil];

return CGRectGetHeight(bodyBounds) + heightForObjectsOnTopOfLabel +
heightForObjectBelowLabel;
}

```

Swift 3

```

override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) ->
CGFloat {
    var cell = tableView.cellForRow(atIndexPath: indexPath)!
    var labelContent = cell.theLabel.text
    var labelFont = cell.theLabel.font
    var paragraphStyle = NSMutableParagraphStyle()

    paragraphStyle.lineBreakMode = .byWordWrapping
    paragraphStyle.alignment = .center

    var attributes = [NSFontAttributeName: labelFont, NSParagraphStyleAttributeName:
paragraphStyle]

    var labelWidth: CGFloat = cell.theLabel.frame.width

    var bodyBounds = labelContent.boundingRect(withSize: CGSize(width: width, height:
CGFLOAT_MAX), options: .usesLineFragmentOrigin, attributes: attributes, context: nil)

    return bodyBounds.height + heightForObjectsOnTopOfLabel + heightForObjectBelowLabel
}

```

Umgekehrt, wenn Sie eine festgelegte maximale Anzahl von Zeilen haben, müssen Sie zuerst die Höhe einer einzelnen Zeile berechnen, um sicherzustellen, dass wir keinen höheren Wert als die zulässige Größe erhalten:

```

// We calculate the height of a line by omitting the NSStringDrawingUsesLineFragmentOrigin
option, which will assume an infinitely wide label
CGRect singleLineRect = [labelContent boundingRectWithSize:CGSizeMake(CGFLOAT_MAX,
CGFLOAT_MAX)

options:NSStringDrawingTruncatesLastVisibleLine
context:nil];

CGFloat lineHeight = CGRectGetHeight(singleLineRect);
CGFloat maxHeight = lineHeight * cell.theLabel.numberOfLines;

```

```
// Now you can call the method appropriately
CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, maxHeight)
options:(NSStringDrawingUsesLineFragmentOrigin|NSStringDrawingTruncatesLastVisibleLine)
attributes:attributes context:nil];

return CGRectGetHeight(bodyBounds) + heightForObjectsOnTopOfLabel +
heightForObjectBelowLabel;
```

Klickbare Beschriftung

HINWEIS: In den meisten Fällen ist es besser, ein `UIButton` anstatt ein `UILabel` zu verwenden. Sie können `UILabel` verwenden, wenn Sie sicher sind, dass Sie aus irgendeinem Grund kein `UIButton` verwenden können.

1. Label erstellen
2. Benutzerinteraktion aktivieren
3. Fügen Sie `UITapGestureRecognizer` hinzu

Der Schlüssel zum Erstellen eines anklickbaren `UILabel` besteht darin, die Benutzerinteraktion zu ermöglichen.

Schnell

```
let label = UILabel()
label.userInteractionEnabled = true

let gesture = UITapGestureRecognizer(target: self, action: #selector(labelClicked(_:)))
label.addGestureRecognizer(gesture)
```

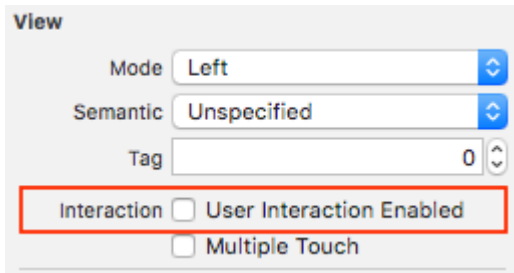
Ziel c

```
UILabel *label = [[UILabel alloc] init];
[label setUserInteractionEnabled:YES];

UITapGestureRecognizer* gesture = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(labelClicked:)];
[label addGestureRecognizer:gesture];
```

Festlegen von "userInteractionEnabled" im Attribute-Inspector des Storyboards

Anstatt Code zu verwenden, können Sie das `UILabel` im Storyboard auswählen und die Option aktivieren:



Dynamischer Etikettenrahmen aus unbekannter Textlänge

Manchmal müssen wir die Größe eines UILabels basierend auf dynamischem Inhalt ändern, bei dem die Textlänge unbekannt ist. In diesem Beispiel ist die Breite des UILabels auf 280 Punkte festgelegt, und die Höhe ist unendlich, sagen wir 9999.

Ziel c

```
UILabel * label = [[UILabel alloc] init];

NSString *message = @"Some dynamic text for label";

//set the text and style if any.
label.text = message;

label.numberOfLines = 0;

CGSize maximumLabelSize = CGSizeMake(280, 9999); //280:max width of label and 9999-max height
of label.

// use font information from the UILabel to calculate the size
CGSize expectedLabelSize = [label sizeThatFits:maximumLabelSize];

//Deprecated in iOS 7.0
//CGSize expectedLabelSize = [message sizeWithFont:label.font
constrainedToSize:maximumLabelSize lineBreakMode:NSLineBreakByWordWrapping];

// create a frame that is filled with the UILabel frame data
CGRect newFrame = label.frame;

// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height;

// put calculated frame into UILabel frame
label.frame = newFrame;
```

Schnell

```
var message: String = "Some dynamic text for label"
//set the text and style if any.
label.text = message
label.numberOfLines = 0
var maximumLabelSize: CGSize = CGSize(width: 280, height: 9999)
var expectedLabelSize: CGSize = label.sizeThatFits(maximumLabelSize)
// create a frame that is filled with the UILabel frame data
var newFrame: CGRect = label.frame
```

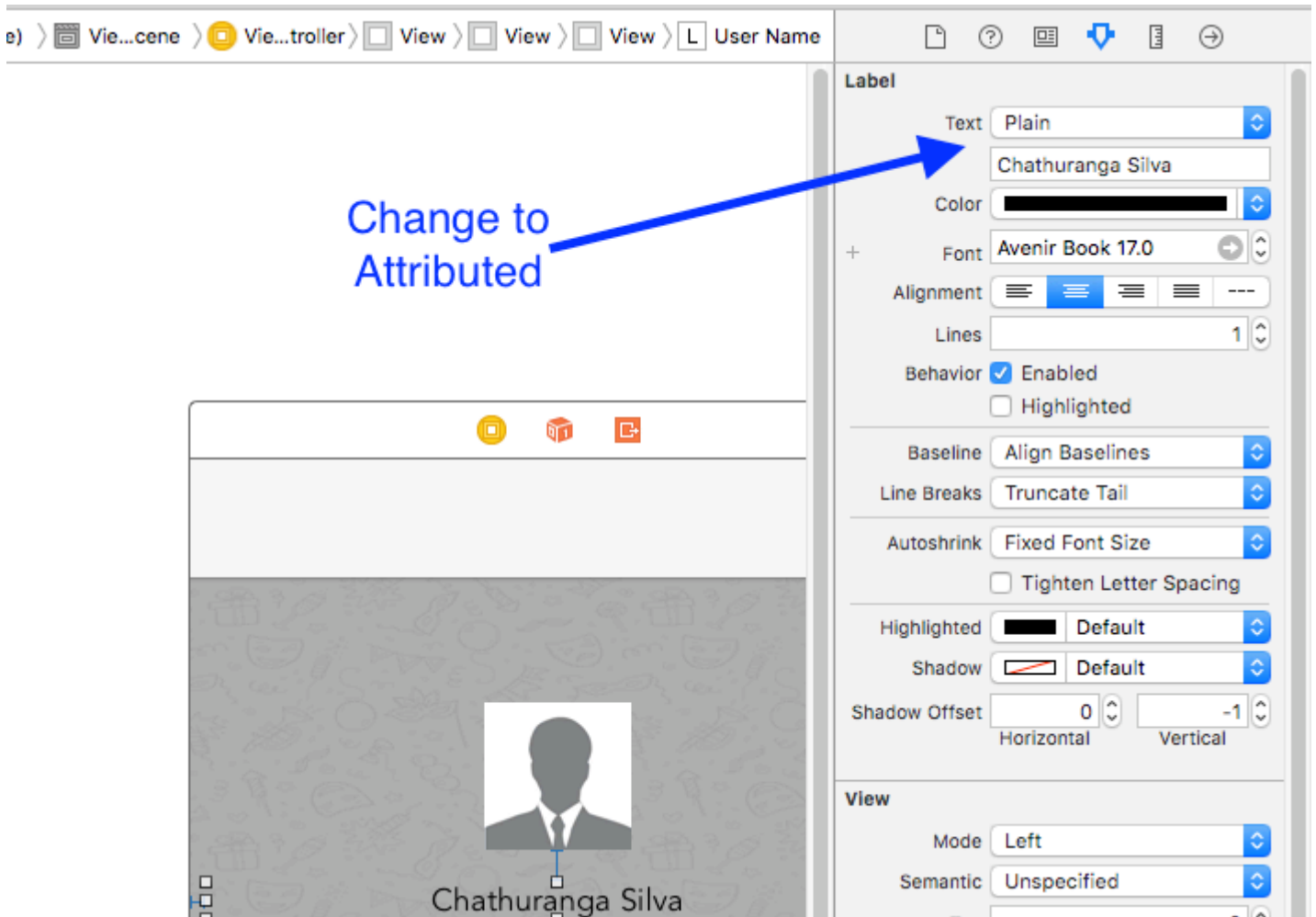
```
// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height
// put calculated frame into UILabel frame
label.frame = newFrame
```

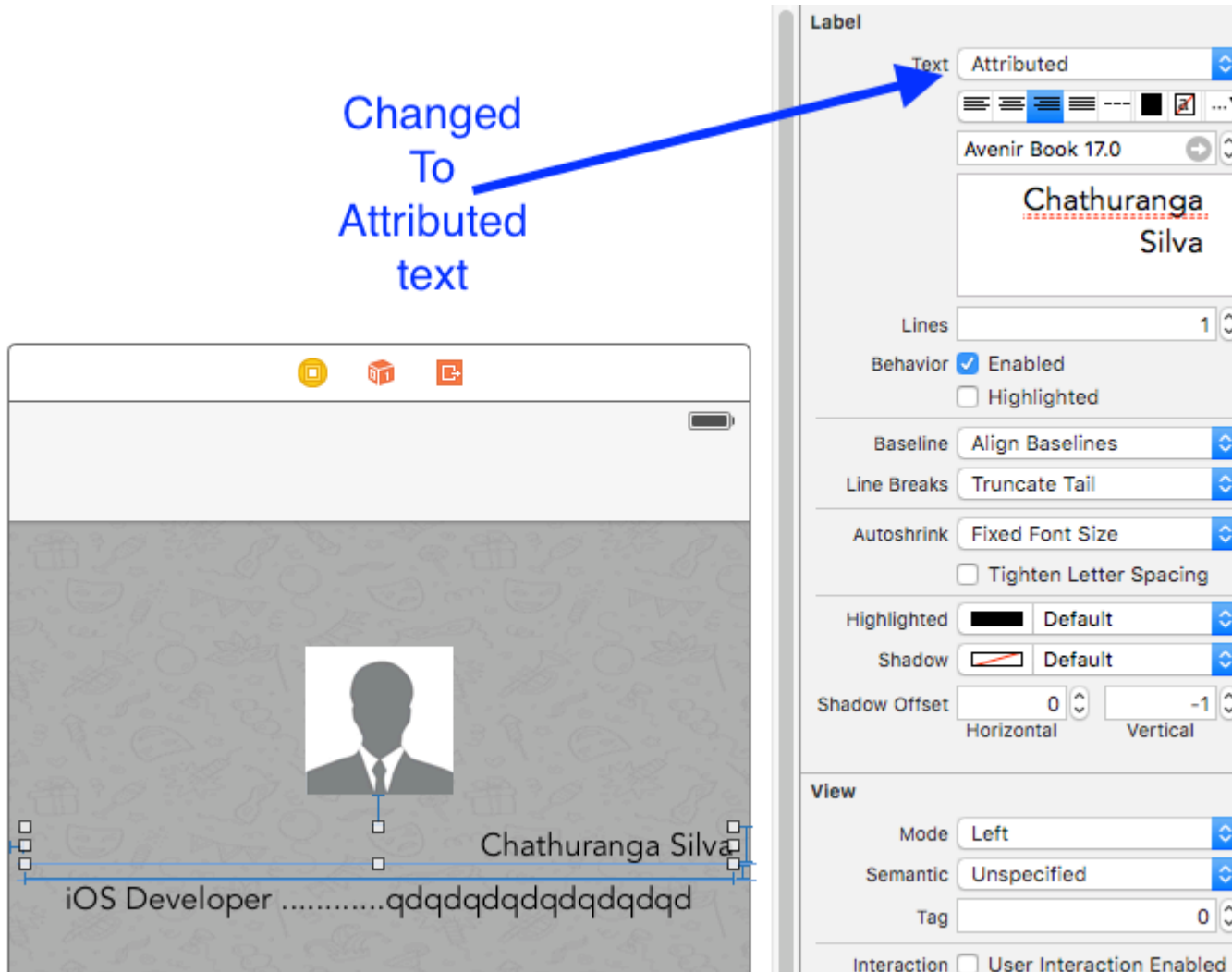
Beschriftung mit Text

01. Text unterstrichen: - Single / Double Line, durchgestrichen: - Single / Double Line

Schritt 1

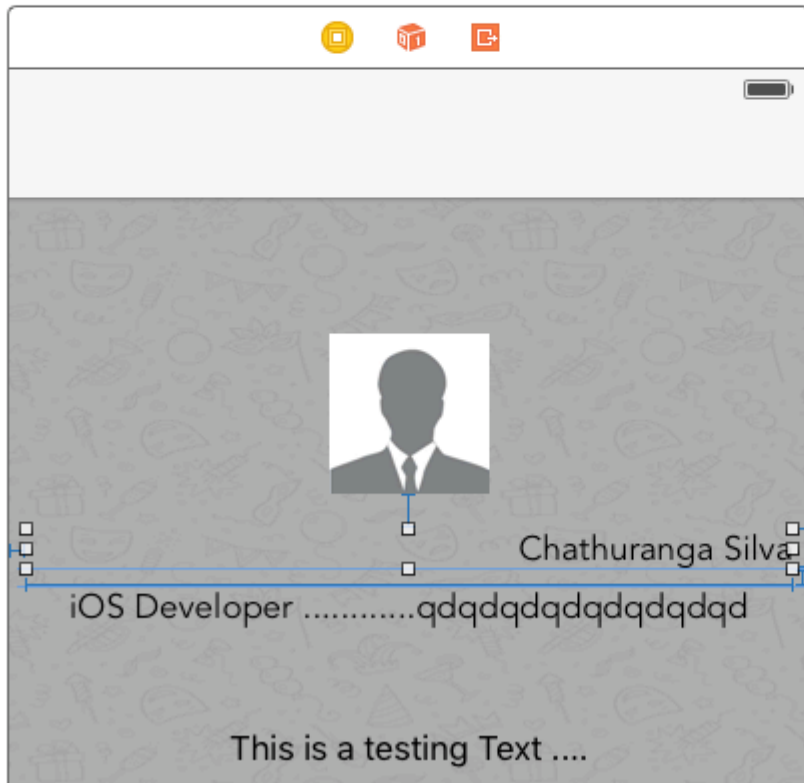
Wählen Sie das Label aus und ändern Sie den Label-Typ Plain in Attributed





Schritt 2

Klicken Sie auf den Etikettentext und klicken Sie mit der rechten Maustaste



Label

Text

Avenir Book 17.0

**Chathuranga
Silva**

Lines

Behavior Enabled
 Highlighted

Baseline

Line Breaks

Autoshrink
 Tighten Letter Spacing

Highlighted Default

Shadow Default

Shadow Offset
Horizontal Vertical

View

Mode

Semantic

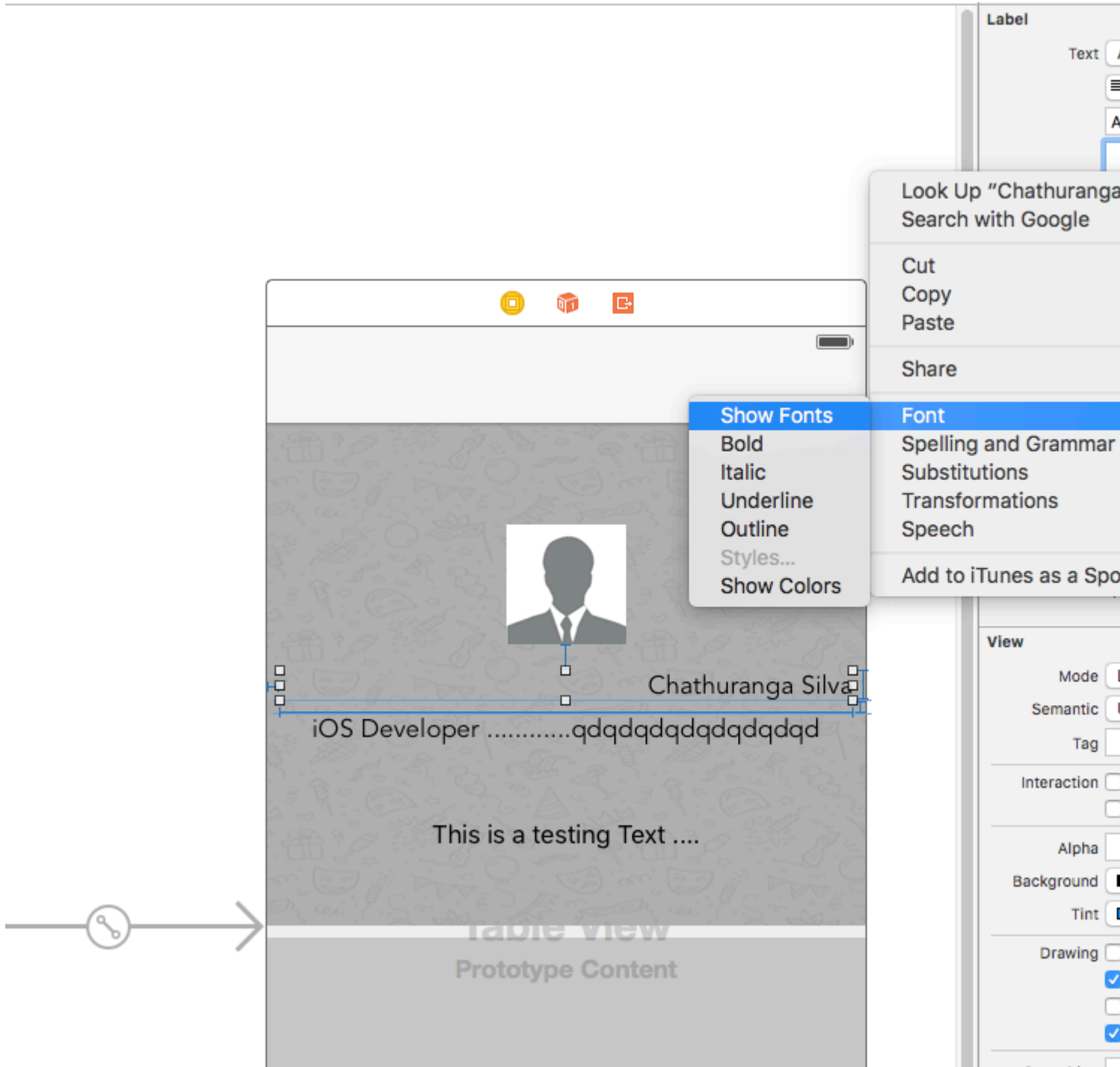
Tag

Interaction User Interaction Enabled
 Multiple Touch

Alpha

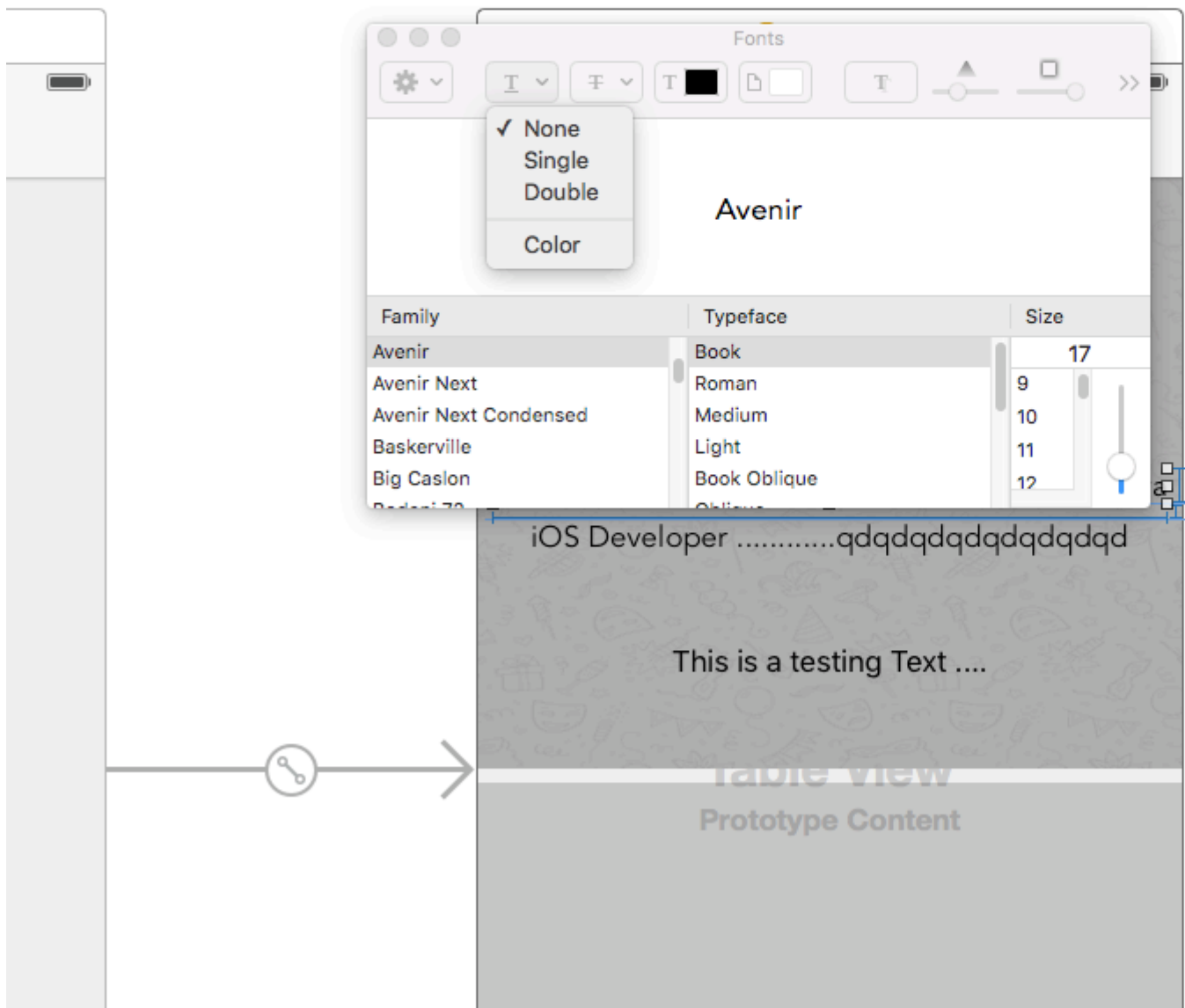
Schritt 3

Klicken Sie dann auf Font -> Show Fonts

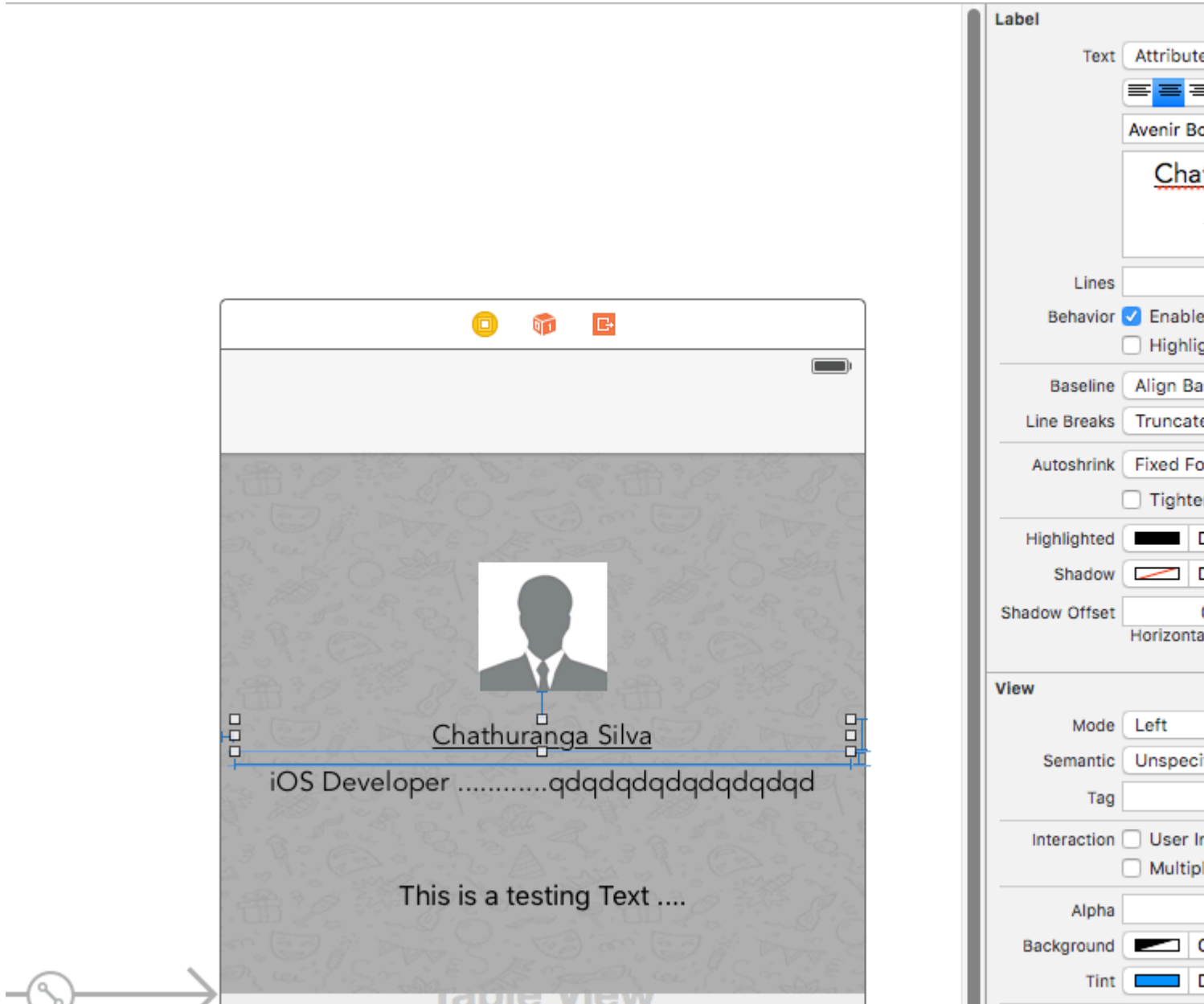


Schritt 4

Die Schriftartenansicht wird angezeigt. Klicken Sie auf die Schaltfläche "Unterstreichen", um den Text zu unterstreichen, oder klicken Sie auf die Schaltfläche "Durchgestrichen", um den Text zu markieren.

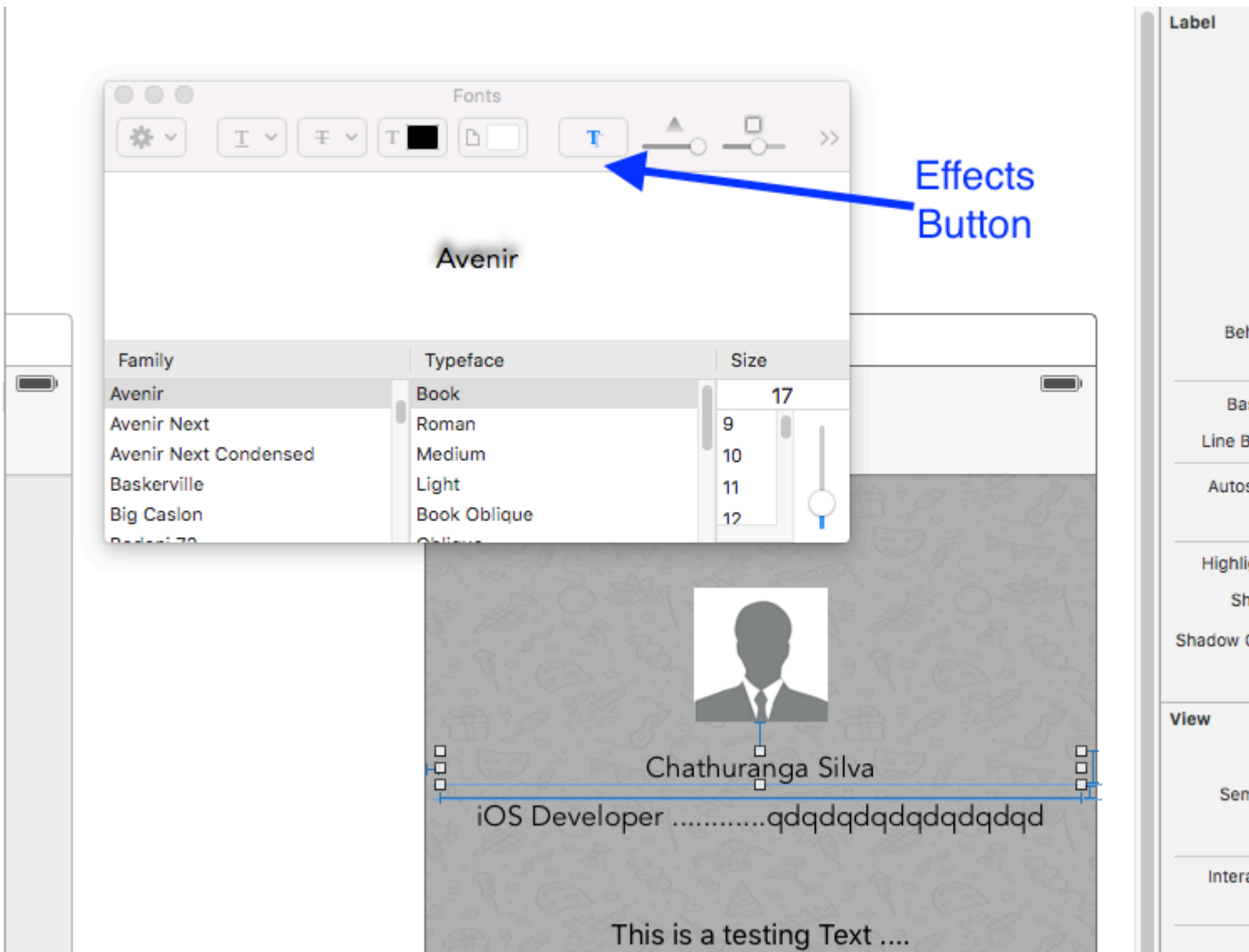


Zum Abschluss klicken Sie auf die Eingabetaste. Das Label wird entsprechend Ihrer Auswahl unterstrichen oder durchgestrichen dargestellt.

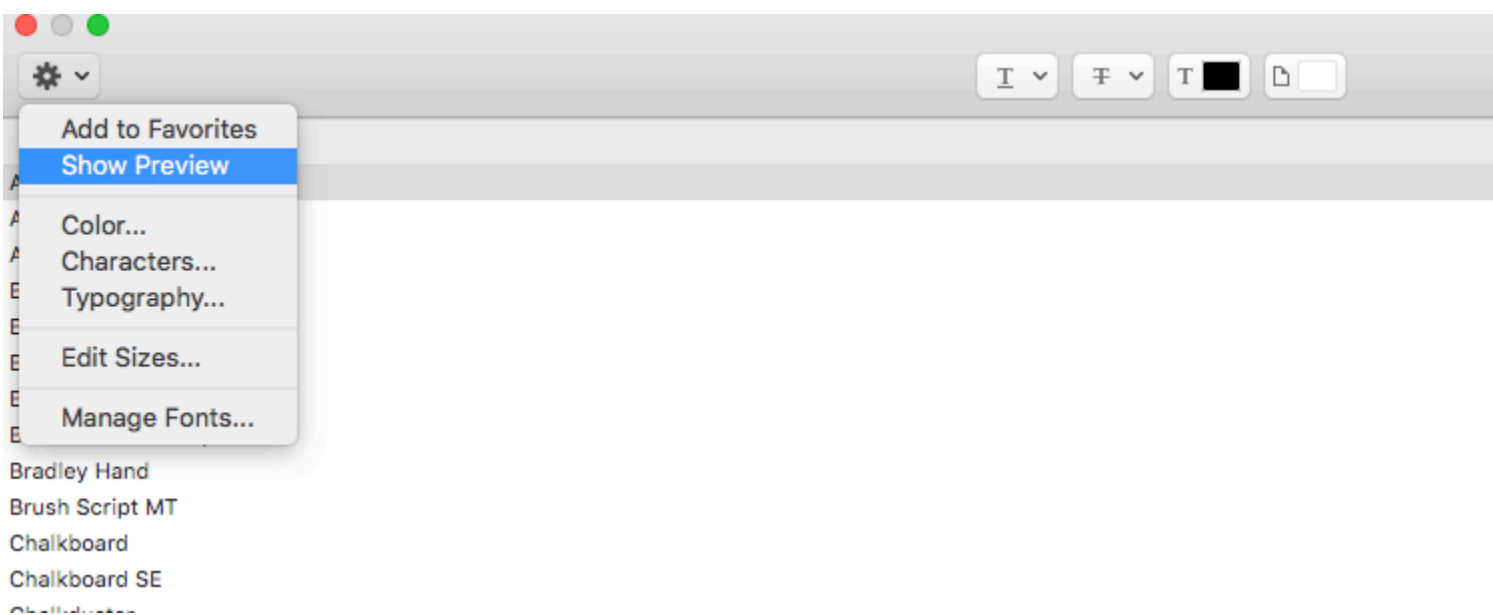


02. Text-Shadow / Hintergrundunschärfe-Effekte hinzufügen

Rufen Sie die Schriftansicht wie oben beschrieben auf und klicken Sie auf die Schaltfläche Effekte.



Wenn Sie die Vorschau nicht sehen, klicken Sie auf das Showbild in den Einstellungen



Ändern Sie schließlich Shaddow und Offset nach Ihren Wünschen.



Avenir

Begründen Sie den Text

Schnell

```
let sampleText = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum."
```

```
// Create label
let label = UILabel(frame: CGRectMake(0, 0, view.frame.size.width, 400))
label.numberOfLines = 0
label.lineBreakMode = NSLineBreakMode.ByWordWrapping
```

```
// Justify text through paragraph style
let paragraphStyle = NSMutableParagraphStyle()
paragraphStyle.alignment = NSTextAlignment.Justified
let attributes = [NSParagraphStyleAttributeName: paragraphStyle,
NSBaselineOffsetAttributeName: NSNumber(float: 0)]
let attributedString = NSAttributedString(string: sampleText, attributes: attributes)
label.attributedString = attributedString
view.addSubview(label)
```

Ziel c

```
NSString *sampleText = @"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.";
```

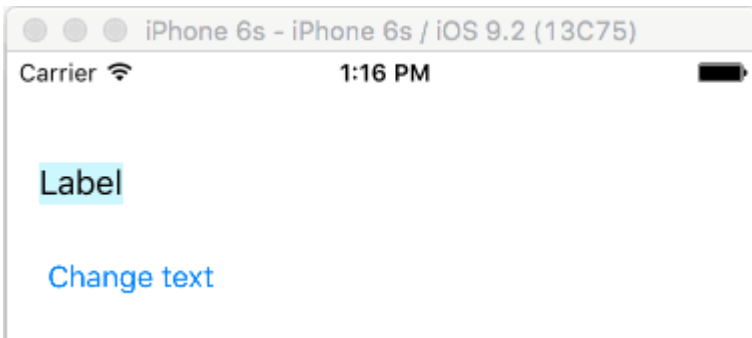
```
// Create label
UILabel *label = [[UILabel alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 400)];
label.numberOfLines = 0;
label.lineBreakMode = NSLineBreakByWordWrapping;
```

```
// Justify text through paragraph style
NSMutableParagraphStyle *paragraphStyle = [[NSMutableParagraphStyle alloc] init];
paragraphStyle.alignment = NSTextAlignmentJustified;
NSAttributedString *attributedString = [[NSAttributedString alloc]
initWithString:sampleText attributes:@{
    NSParagraphStyleAttributeName : paragraphStyle,
    NSBaselineOffsetAttributeName : [NSNumber numberWithInt:0]}
```

```
    }];  
    label.attributedString = attributedString;  
    [self.view addSubview:label];
```

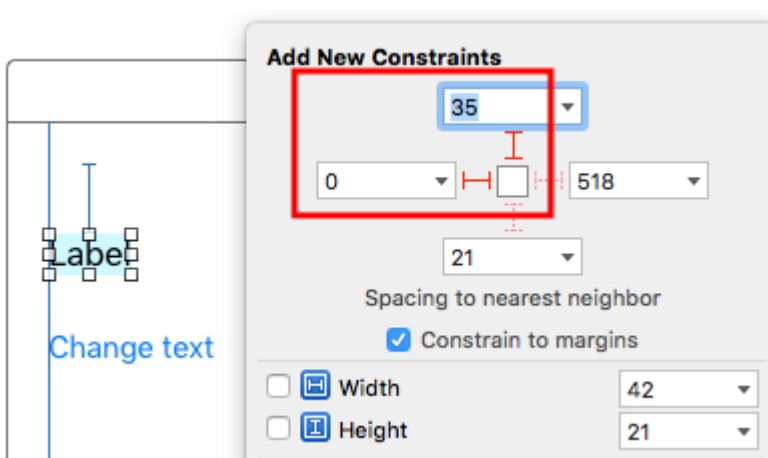
Beschriftung mit automatischer Größe für den Text

Dieses Beispiel zeigt, wie die Breite eines Etiketts automatisch angepasst werden kann, wenn sich der Textinhalt ändert.



Befestigen Sie die linke und obere Kante

Verwenden Sie das automatische Layout, um Einschränkungen hinzuzufügen, um die linke und obere Seite des Etiketts zu fixieren.



Danach wird die Größe automatisch angepasst.

Anmerkungen

- Dieses Beispiel stammt aus [dieser Stack Overflow-Antwort](#) .
- Fügen Sie keine Einschränkungen für die Breite und Höhe hinzu. Etiketten haben eine *intrinsische* Größe, die auf ihrem Textinhalt basiert.
- Bei Verwendung des automatischen Layouts muss `sizeToFit` werden. Der vollständige Code für das Beispielprojekt ist hier:


```

import UIKit
class ViewController: UIViewController {

    @IBOutlet weak var myLabel: UILabel!

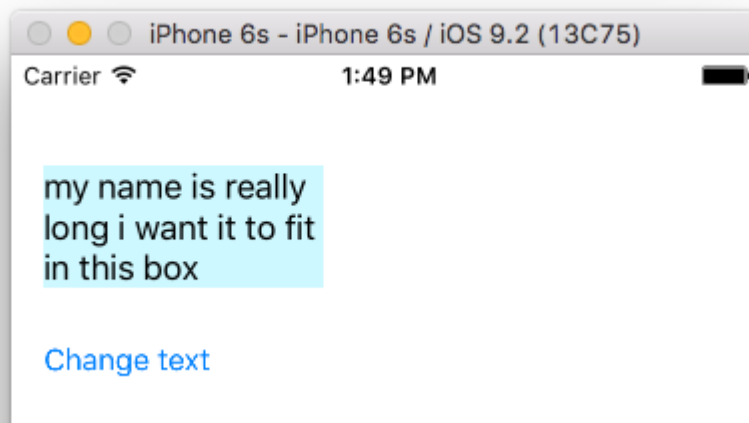
    @IBAction func changeTextButtonTapped(sender: UIButton) {
        myLabel.text = "my name is really long i want it to fit in this box"
    }
}

```

- Diese Methode kann auch verwendet werden, um mehrere Beschriftungen wie in [diesem Beispiel](#) horizontal korrekt zu platzieren.



- Wenn Sie möchten, dass Ihr Label Zeilenumbruch aufweist, setzen Sie die Anzahl der Zeilen in IB auf 0 und fügen Sie `myLabel.preferredMaxLayoutWidth = 150 // or whatever` im Code hinzu. (Die Schaltfläche ist auch an der Unterseite des Etiketts fixiert, sodass sie sich nach unten bewegt, wenn die Etikettenhöhe erhöht wird.)



Ermitteln Sie die Größe von UILabel streng nach Text und Schriftart

`NSString` stellt die Methode `boundingRectWithSize` bereit, mit der die resultierende `CGSize` eines `UILabel` basierend auf seinem Text und seiner Schriftart `UILabel` ohne dass ein `UILabel`

Ziel c

```

[[text boundingRectWithSize:maxSize options:(NSStringDrawingTruncatesLastVisibleLine |
NSStringDrawingUsesLineFragmentOrigin) attributes:@{NSFontAttributeName: fontName}
context:nil] size];

```

Schnell

```
let nsText = text as NSString?
nsText?.boundingRectWithSize(maxSize, options: [.TruncatesLastVisibleLine,
.UsesLineFragmentOrigin], attributes: [NSFontAttributeName: fontName], context: nil).size
```

Schnell

Erstellen Sie einen Auslass für Beschriftung und Beschriftung für die Höhenbeschränkung. Fügen Sie unten den Code hinzu, an dem Sie Text für die Beschriftung festlegen möchten.

```
@IBOutlet var lblDescriptionHeightConstration: NSLayoutConstraint!
@IBOutlet weak var lblDescription: UILabel!

let maxWidth = UIScreen.mainScreen().bounds.size.width - 40
let sizeOfLabel = self.lblDesc.sizeThatFits(CGSize(width: maxWidth, height: CGFloat.max))
self.lblDescriptionHeightConstration.constant = sizeOfLabel.height
```

Hinweis: "40" ist der Platz auf der linken und rechten Seite des Bildschirms.

Hervorgehobene und hervorgehobene Textfarbe

Ziel c

```
UILabel *label = [[UILabel alloc] init];
label.highlighted = YES;
label.highlightedTextColor = [UIColor redColor];
```

Schnell

```
let label = UILabel()
label.highlighted = true
label.highlightedTextColor = UIColor.redColor()
```

Swift 3

```
let label = UILabel()
label.isHighlighted = true
label.highlightedTextColor = UIColor.red
```

UILabel online lesen: <https://riptutorial.com/de/ios/topic/246/UILabel>

Kapitel 172: UILabel-Text unterstrichen

Examples

Unterstrichen eines Textes in einem UILabel mit Objective C

```
UILabel *label=[[UILabel alloc]initWithFrame:CGRectMake(0, 0, 320, 480)];
label.backgroundColor=[UIColor lightGrayColor];
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply Underlining"];
[attributedString addAttribute:NSUnderlineStyleAttributeName value:@1 range:NSMakeRange(0,
[attributedString length])];
[label setAttributedText:attributedString];
```

Einen Text in UILabel mit Swift unterstreichen

```
let label = UILabel.init(frame: CGRect(x: 0, y:0, width: 100, height: 40))
label.backgroundColor = .lightGray
let attributedString = NSMutableAttributedString.init(string: "Apply UnderLining")
attributedString.addAttribute(NSUnderlineStyleAttributeName, value: 1, range:
NSRange.init(location: 0, length: attributedString.length))
label.attributedText = attributedString
```

UILabel-Text unterstrichen online lesen: <https://riptutorial.com/de/ios/topic/7219/ui-label-text-unterstrichen>

Kapitel 173: UILocalNotification

Einführung

Lokale Benachrichtigungen ermöglichen es Ihrer App, den Benutzer über Inhalte zu informieren, für die kein Server erforderlich ist.

Im Gegensatz zu Remote-Benachrichtigungen, die von einem Server ausgelöst werden, werden lokale Benachrichtigungen in einer App geplant und ausgelöst. Benachrichtigungen zielen im Allgemeinen darauf ab, die Benutzerinteraktion mit der App zu verbessern, indem der Benutzer dazu eingeladen wird, die App zu öffnen und damit zu interagieren.

UILocalNotification wurde in iOS 10 nicht mehr unterstützt. Verwenden Sie stattdessen das Framework UserNotifications.

Bemerkungen

Verwechseln Sie UILocalNotification nicht mit Push-Benachrichtigungen. Die UILocalNotification wird von Ihrem Gerät ausgelöst und bei der Planung in das System kopiert.

Links:

- [UILocalNotification-Klassenreferenz](#)
- [UILocalNotification bei Stack Overflow](#)

Examples

Lokale Benachrichtigung planen

Stellen Sie sicher, dass Sie [für lokale Benachrichtigungen registrieren](#) sehen, damit dies funktioniert:

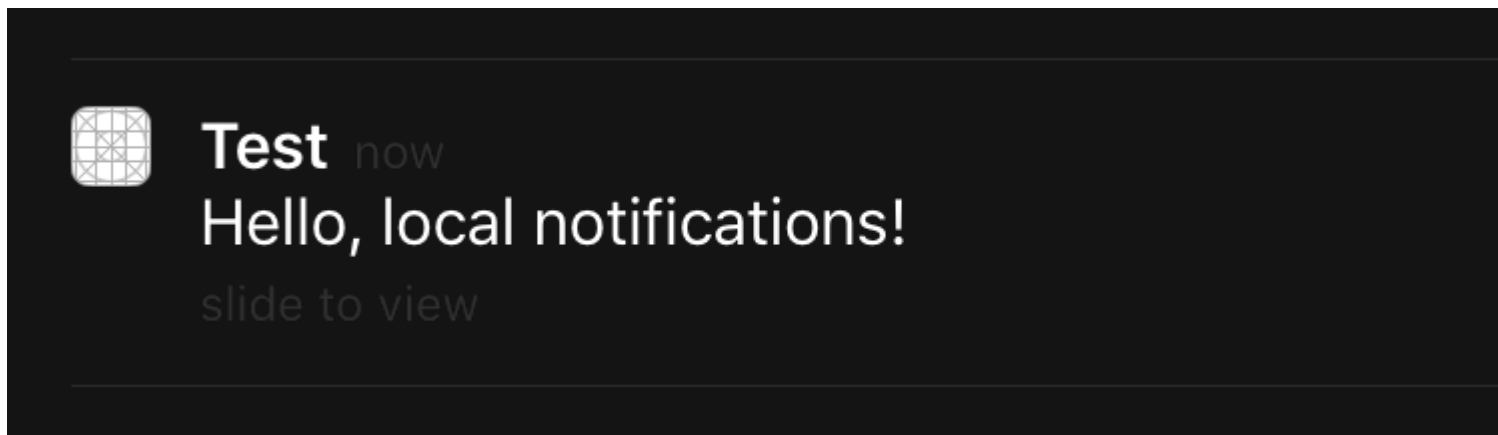
Schnell

```
let notification = UILocalNotification()
notification.alertBody = "Hello, local notifications!"
notification.fireDate = NSDate().dateByAddingTimeInterval(10) // 10 seconds after now
UIApplication.sharedApplication().scheduleLocalNotification(notification)
```

Ziel c

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.alertBody = @"Hello, local notifications!";
notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10]; // 10 seconds after now
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Um die Benachrichtigung im iOS-Simulator $\wedge \square \text{H}$, geben Sie $\wedge \square \text{H}$ (Steuerbefehl-H) ein, um nach Hause zu gelangen, und geben Sie dann $\square \text{L}$ (Befehl-L) ein, um das Gerät zu sperren. Warten Sie einige Sekunden, und die Benachrichtigung sollte erscheinen (diese Erscheinung variiert je nach Benachrichtigungstyp, der unter "Registrieren für lokale Benachrichtigungen" beschrieben wird):



Wischen Sie über die Benachrichtigung, um zur App zurückzukehren. Wenn Sie dies in `viewDidLoad` des ersten View-Controllers, `viewWillAppear` , `viewDidAppear` usw. `viewDidAppear` , wird die Benachrichtigung erneut geplant.

Registrierung für lokale Benachrichtigungen

iOS 8

Um dem Benutzer lokale Benachrichtigungen anzuzeigen, müssen Sie Ihre App beim Gerät registrieren:

Schnell

```
let settings = UIUserNotificationSettings(forTypes: [.Badge, .Sound, .Alert], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(settings)
```

Ziel c

```
UIUserNotificationSettings *settings = [UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeBadge | UIUserNotificationTypeSound |
UIUserNotificationTypeAlert) categories:nil];
[[UIApplication sharedApplication] registerUserNotificationSettings:settings];
```

Beim ersten Aufruf wird eine Warnung angezeigt:

"Test" Would Like to Send You Notifications

Notifications may include alerts, sounds, and icon badges. These can be configured in Settings.

Don't Allow

OK

Unabhängig davon, was der Benutzer auswählt, wird die Warnung nicht erneut angezeigt und Änderungen müssen vom Benutzer in den Einstellungen vorgenommen werden.

Antwort auf empfangene lokale Benachrichtigung

WICHTIG: Diese Delegatmethode wird nur im Vordergrund aufgerufen.

Schnell

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {

}
```

Ziel c

```
- (void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {

}
```

Diese Methode wird im Allgemeinen in AppDelegate überschrieben, das dem UIApplicationDelegate-Protokoll entspricht.

Lokale Benachrichtigungen mit UUID verwalten

Oft müssen Sie Ihre Benachrichtigungen verwalten können, indem Sie sie nachverfolgen und abbrechen können.

Eine Benachrichtigung verfolgen

Sie können einer Benachrichtigung eine UUID (universell eindeutiger Bezeichner) zuweisen, um sie nachverfolgen zu können:

Schnell

```
let notification = UILocalNotification()
let uuid = NSUUID().uuidString
notification.userInfo = ["UUID": uuid]
UIApplication.shared.scheduleLocalNotification(notification)
```

Ziel c

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
NSString *uuid = [[NSUUID UUID] UUIDString];
notification.userInfo = @{@"UUID": uuid };
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Eine Benachrichtigung abbrechen

Um eine Benachrichtigung abzubrechen, erhalten wir zunächst eine Liste aller Benachrichtigungen und suchen dann die Benachrichtigung mit einer entsprechenden UUID. Zum Schluss stornieren wir es.

Schnell

```
let scheduledNotifications = UIApplication.shared.scheduledLocalNotifications

guard let scheduledNotifications = scheduledNotifications else {
    return
}

for notification in scheduledNotifications where "\(notification.userInfo!["UUID"]!)" ==
UUID_TO_CANCEL {
    UIApplication.sharedApplication().cancelLocalNotification(notification)
}
```

Ziel c

```
NSArray *scheduledNotifications = [[UIApplication sharedApplication]
scheduledLocalNotifications];

for (UILocalNotification *notification in scheduledNotifications) {
    if ([[notification.userInfo objectForKey:@"UUID"] compare: UUID_TO_CANCEL]) {
        [[UIApplication sharedApplication] cancelLocalNotification:notification];
        break;
    }
}
```

Sie möchten wahrscheinlich alle diese UUIDs in Core Data oder Realm speichern.

Eine lokale Benachrichtigung wird sofort angezeigt

Wenn Sie die lokale Benachrichtigung sofort anzeigen möchten, rufen Sie an:

Swift 3

```
UIApplication.shared.presentLocalNotificationNow(notification)
```

Schnell 2

```
UIApplication.sharedApplication().presentLocalNotificationNow(notification)
```

Ziel c

```
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Dies hat den Vorteil, dass Sie die Eigenschaften `fireDate` und `timeZone` Ihres `UILocalNotification` Objekts nicht `UILocalNotification` .

Benachrichtigungston

Für Benachrichtigungen, die von Ihrer App generiert werden, können benutzerdefinierte Töne bereitgestellt werden. Wenn das System eine Benachrichtigung für eine lokale Benachrichtigung anzeigt oder ein App-Symbol kennzeichnet, wird dieses Geräusch abgespielt (solange der Benutzer die Benachrichtigungstöne nicht deaktiviert hat).

Der Standardwert ist Null, was bedeutet, dass für Ihre Benachrichtigung kein Ton abgespielt wird.

Um einen benutzerdefinierten Sound zu liefern, fügen Sie eine `.caf` , `.wav` oder `.aiff` - Datei auf Ihren App - Bundle. Sounds, die länger als 30 Sekunden dauern, werden nicht unterstützt. Wenn Sie einen Sound `UILocalNotificationDefaultSoundName` , der diese Anforderungen nicht erfüllt, wird der Standard-Sound `UILocalNotificationDefaultSoundName` (`UILocalNotificationDefaultSoundName`).

Ziel c

```
UILocalNotification *notification = [UILocalNotification new];  
notification.soundName = @"nameOfSoundInBundle.wav"; // Use  
UILocalNotificationDefaultSoundName for the default alert sound
```

Schnell

```
let notification = UILocalNotification()  
notification.soundName = "nameOfSoundInBundle.wav"
```

Registrieren und Planen der lokalen Benachrichtigung in Swift 3.0 (iOS 10)

Anmeldung

in AppDelegate


```
import UserNotifications
```

in der Methode `didFinishLaunchingWithOptions`

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) {  
    (granted, error) in  
  
    // Here you can check Request is Granted or not.  
  
}
```

Benachrichtigung erstellen und planen

```
let content = UNMutableNotificationContent()  
content.title = "10 Second Notification Demo"  
content.subtitle = "From Wolverine"  
content.body = "Notification after 10 seconds - Your pizza is Ready!!"  
content.categoryIdentifier = "myNotificationCategory"  
  
let trigger = UNTimeIntervalNotificationTrigger(  
    timeInterval: 10.0,  
    repeats: false)  
  
let request = UNNotificationRequest(  
    identifier: "10.second.message",  
    content: content,  
    trigger: trigger  
)  
UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
```

Wenn dieser Teil des Codes ausgelöst wird, erhalten Sie eine Benachrichtigung, wenn Sie die Benachrichtigungsberechtigung erteilt haben.

Um es richtig zu testen, stellen Sie sicher, dass sich Ihre Anwendung im Hintergrundmodus befindet.

Was ist neu in `UILocalNotification` mit iOS10?

Sie können `UILocalNotification`, alte APIs funktionieren auch gut mit iOS10, aber wir sollten stattdessen die APIs im Framework für Benutzerbenachrichtigungen verwenden. Es gibt auch einige neue Funktionen, die Sie nur mit dem Benutzerbenachrichtigungs-Framework von iOS10 verwenden können.

Dies geschieht auch mit der Remote-Benachrichtigung, für weitere Informationen: [Hier](#).

Neue Eigenschaften:

1. Jetzt können Sie entweder Alarm ausgeben, ein Signal ausgeben oder das Abzeichen erhöhen, während sich die App auch mit iOS 10 im Vordergrund befindet
2. Jetzt können Sie alle Ereignisse an einer Stelle abwickeln, wenn der Benutzer auf die Aktionsschaltfläche tippte (oder schob), selbst wenn die App bereits beendet wurde.
3. Unterstützen Sie 3D-Touch anstelle von gleitender Geste.

4. Jetzt können Sie spezifische lokale Benachrichtigungen nur durch einen Zeilencode entfernen.
5. Rich-Benachrichtigung mit benutzerdefinierter Benutzeroberfläche unterstützen.

Es ist sehr einfach für uns, `UINotification` APIs in iOS10-Benutzerbenachrichtigungs-Framework-APIs umzuwandeln, sie sind sich sehr ähnlich.

Ich schreibe hier eine Demo, um zu zeigen, wie neue und alte APIs gleichzeitig verwendet werden: [iOS10AdaptationTips](#) .

Zum Beispiel,

Mit Swift-Implementierung:

1. UserNotifications importieren

```
/// Notification become independent from UIKit
import UserNotifications
```

2. Berechtigung für localNotification anfordern

```
let center = UNUserNotificationCenter.current()
center.requestAuthorization(options: [.alert, .sound]) { (granted, error) in
    // Enable or disable features based on authorization.
}
```

3. localNotification einplanen

4. Anwendungssymbol-Ausweisnummer aktualisieren

```
@IBAction func triggerNotification(){
    let content = UNMutableNotificationContent()
    content.title = NSLocalizedString(localizedUserNotificationString(forKey: "Elon said:",
arguments: nil)
    content.body = NSLocalizedString(localizedUserNotificationString(forKey: "Hello Tom Get up,
let's play with Jerry!", arguments: nil)
    content.sound = UNNotificationSound.default()
    content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
    content.categoryIdentifier = "com.elonchan.localNotification"
    // Deliver the notification in five seconds.
    let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60.0, repeats:
true)
    let request = UNNotificationRequest.init(identifier: "FiveSecond", content: content,
trigger: trigger)

    // Schedule the notification.
    let center = UNUserNotificationCenter.current()
    center.add(request)
}

@IBAction func stopNotification(_ sender: AnyObject) {
    let center = UNUserNotificationCenter.current()
    center.removeAllPendingNotificationRequests()
    // or you can remove specifical notification:
```

```
// center.removePendingNotificationRequests(withIdentifiers: ["FiveSecond"])
}
```

Ziel-C-Implementierung:

1. UserNotifications importieren

```
// Notifications are independent from UIKit
#import <UserNotifications/UserNotifications.h>
```

2. Berechtigung für localNotification anfordern

```
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
[center requestAuthorizationWithOptions:(UNAuthorizationOptionBadge |
UNAuthorizationOptionSound | UNAuthorizationOptionAlert)
    completionHandler:^(BOOL granted, NSError * _Nullable error) {
    if (!error) {
        NSLog(@"request authorization succeeded!");
        [self showAlert];
    }
}];
```

3. localNotification einplanen

4. Anwendungssymbol-Ausweisnummer aktualisieren

```
UNMutableNotificationContent *content = [[UNMutableNotificationContent alloc] init];
content.title = [NSString localizedUserNotificationStringForKey:@"Elon said:"
    arguments:nil];

content.body = [NSString localizedUserNotificationStringForKey:@"Hello Tom Get up, let's
play with Jerry!"
    arguments:nil];

content.sound = [UNNotificationSound defaultSound];

// 4. update application icon badge number
content.badge = [NSNumber numberWithInt:([UIApplication
sharedApplication].applicationIconBadgeNumber + 1)];
// Deliver the notification in five seconds.
UNTimeIntervalNotificationTrigger *trigger = [UNTimeIntervalNotificationTrigger
    triggerWithTimeInterval:5.f
    repeats:NO];

UNNotificationRequest *request = [UNNotificationRequest
    requestWithIdentifier:@"FiveSecond"
    content:content
    trigger:trigger];

/// 3. schedule localNotification
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
[center addNotificationRequest:request withCompletionHandler:^(NSError * _Nullable error)
{
    if (!error) {
        NSLog(@"add NotificationRequest succeeded!");
    }
}];
}];
```

Hier finden Sie weitere Informationen: [iOS10AdaptationTips](#) .

#aktualisierte

App wird wegen nicht abgerufener Ausnahme 'NSInternalInconsistencyException' beendet, Grund: 'Zeitintervall muss mindestens 60 sein, wenn wiederholt'

```
let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60, repeats: true)
```

UILocalNotification online lesen: <https://riptutorial.com/de/ios/topic/635/uilocalnotification>

Kapitel 174: UINavigationController

Bemerkungen

Aus der [Dokumentation](#) :

Die UINavigationController-Klasse implementiert einen speziellen View-Controller, der die Navigation hierarchischer Inhalte verwaltet. Diese Navigationsschnittstelle ermöglicht eine effiziente Darstellung Ihrer Daten und erleichtert dem Benutzer das Navigieren in diesen Inhalten. Sie verwenden diese Klasse im Allgemeinen wie sie ist, aber Sie können auch eine Unterklasse verwenden, um das Klassenverhalten anzupassen.

Examples

In einem Navigations-Controller einblenden

Zum vorherigen Ansichtcontroller

Um zur vorherigen Seite zurückzukehren, können Sie Folgendes tun:

Schnell

```
navigationController?.popViewControllerAnimated(true)
```

Ziel c

```
[self.navigationController popViewControllerAnimated:YES];
```

Zum Root-View-Controller

Um zum Stamm des Navigationsstapels zu gelangen, können Sie Folgendes tun:

Schnell

```
navigationController?.popToRootViewControllerAnimated(true)
```

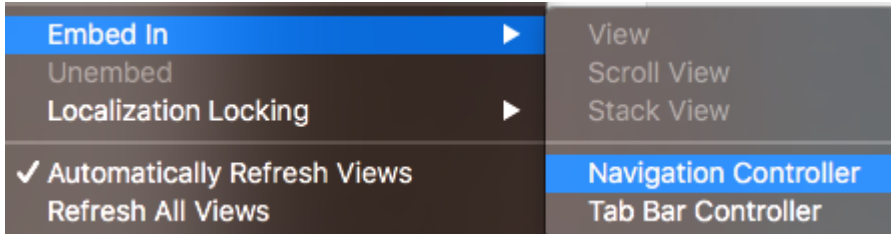
Ziel c

```
[self.navigationController popToRootViewControllerAnimated:YES];
```

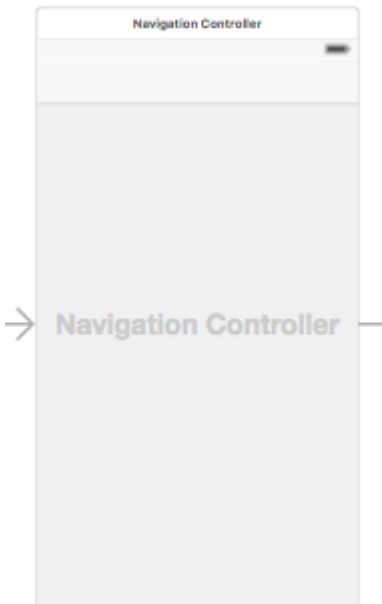
Einen Navigationscontroller erstellen

Wählen Sie in Ihrem Storyboard den ViewController aus, den Sie in einen Navigationscontroller einbetten möchten.

Navigieren Sie dann zu Editor > Einbetten in > Navigationscontroller



Und damit wird Ihr Navigationscontroller erstellt



Betten Sie einen View-Controller programmgesteuert in einen Navigationscontroller ein

Schnell

```
//Swift
let viewController = UIViewController()
let navigationController = UINavigationController(rootViewController: viewController)

//Objective-C
UIViewController *viewController = [[UIViewController alloc] init];
UINavigationController *navigationController = [[UINavigationController alloc]
initWithRootViewController:viewController];
```

Einen View Controller auf den Navigationsstapel schieben

```
//Swift
let fooViewController = UIViewController()
navigationController?.pushViewController(fooViewController, animated: true)

//Objective-C
UIViewController *fooViewController = [[UIViewController alloc] init];
[navigationController pushViewController:fooViewController animated:YES];
```

Zweck

`UINavigationController` wird verwendet, um eine baumartige Hierarchie von View-Controllern zu bilden, die als `navigation stack`.

Aus der Sicht der Entwickler:

Sie können einen unabhängigen Controller anschließen und alle Vorteile eines kostenlosen Hierarchie-Managers und eines allgemeinen UI-Presenters kostenlos nutzen.

`UINavigationController` animiert den Übergang zu neuen Controllern und stellt die Back-Funktionalität automatisch für Sie bereit. `UINavigationController` auch den Zugriff auf alle anderen Controller im `navigation stack` die den Zugriff auf bestimmte Funktionen oder Daten

`UINavigationController`.

Aus Benutzersicht:

`UINavigationController` hilft, sich zu erinnern, wo sich der Benutzer gerade befindet (Navigationsleiste) und wie er zurückkehren kann (eingebetteter Zurück-Button) zu einem der vorherigen Bildschirme.

`UINavigationController` online lesen: <https://riptutorial.com/de/ios/topic/1079/uINavigationController>

Kapitel 175: UIPageViewController

Einführung

Mit UIPageViewController können Benutzer mithilfe einer Wischgeste problemlos zwischen verschiedenen Ansichten wechseln. Um einen UIPageViewController erstellen zu können, müssen Sie die Methoden UIPageViewControllerDataSource implementieren. Dazu gehören Methoden, um sowohl den UIPageViewController als auch vor dem aktuellen UIPageViewController zusammen mit den Methoden presentationCount und presentationIndex zurückzugeben.

Syntax

1. UIPageViewControllerTransitionStyle
2. UIPageViewControllerNavigationOrientation
3. UIPageViewControllerSpineLocation
4. UIPageViewControllerNavigationDirection

Bemerkungen

Apple-Entwicklerreferenz [hier](#)

Examples

Erstellen Sie programmgesteuert einen horizontalen Paging-UIPageViewController

1. Init Array von View-Controllern, die von UIPageViewController verwaltet werden. Hinzufügen eine Basis View - Controller - Klasse , die Eigenschaft hat `identifier` , die verwendet wird , um zu identifizieren , wenn Ansicht - Controller mit UIPageViewController Datenquelle Methoden arbeiten. Lassen Sie die View-Controller von dieser Basisklasse erben.

```
UIViewController *firstVC = [[UIViewController alloc] init];
firstVC.identifier = 0
UIViewController *secondVC = [[UIViewController alloc] init];
secondVC.identifier = 1
NSArray *viewControllers = [[NSArray alloc] initWithObjects: firstVC, secondVC, nil];
```

2. Erstellen Sie eine UIPageViewController-Instanz.

```
UIPageViewController *pageViewController = [[UIPageViewController alloc]
initWithTransitionStyle:UIPageViewControllerTransitionStyleScroll

navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal

options:nil];
```


3. Datenquelle ist die aktuelle Klasse, die das `UIPageViewControllerDataSource` Protokoll implementieren `UIPageViewControllerDataSource` .

```
pageViewController.dataSource = self;
```

4. `setViewControllers` fügt nur den ersten View-Controller hinzu, der nächste wird mithilfe von Datenquellenmethoden zum Stack hinzugefügt

```
if (viewControllers.count) {
    [pageViewController setViewControllers:@[[viewControllers objectAtIndex:0]]
                        direction:UIPageViewControllerNavigationDirectionForward
                        animated:NO
                        completion:nil];
}
```

5. Fügen Sie `UIPageViewController` als untergeordneter Ansichts-Controller hinzu, damit er von seinem übergeordneten Ansichts-Controller- `appearance` und `rotation` empfangen wird.

```
[self addChildViewController:pageViewController];
pageViewController.view.frame = self.view.frame;
[self.view addSubview:pageViewController.view];
[pageViewController didMoveToParentViewController:self];
```

6. Implementieren von `UIPageViewControllerDataSource`-Methoden

```
- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController
viewControllerBeforeViewController:(UIViewController *)viewController
{
    index = [(Your View Controller Base Class *)viewController identifier];
    index--;
    return [self childViewControllerAtIndex:index];
}

- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController
viewControllerAfterViewController:(UIViewController *)viewController
{
    index = [(Your View Controller Base Class *)viewController identifier];
    index++;
    return [self childViewControllerAtIndex:index];
}

- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController
{
    return [viewControllers count];
}

- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController
{
    return index;
}
```

7. Dienstprogrammmethode, die einen View-Controller mit einem Index zurückgibt, wenn der Index außerhalb der Grenzen liegt, gibt er null zurück.

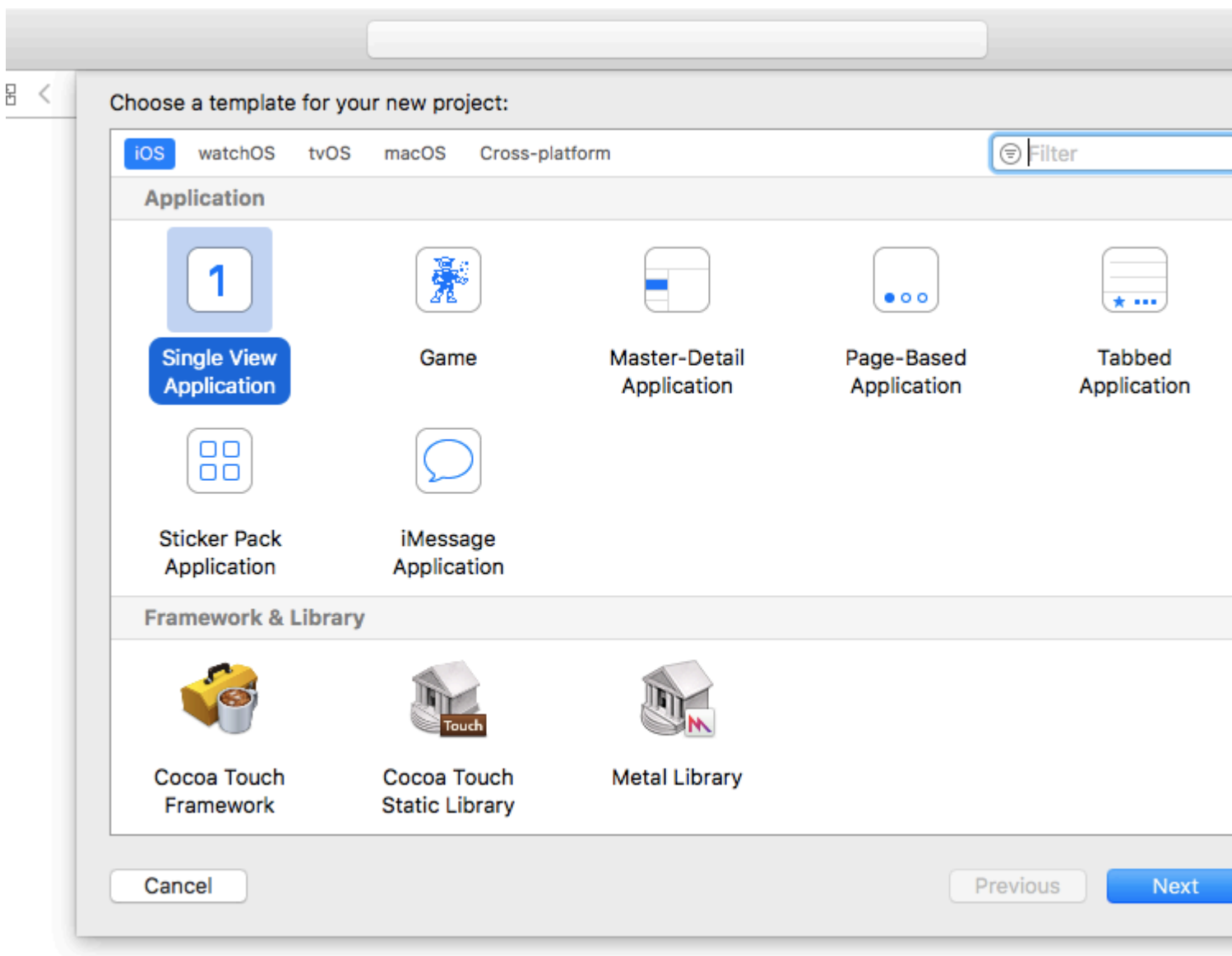
```

- (UIViewController *)childViewControllerAtIndex:(NSInteger) index
{
    if (index <= ([viewControllers count] - 1)) {
        return [viewControllers objectAtIndex:index];
    } else {
        return nil;
    }
}

```

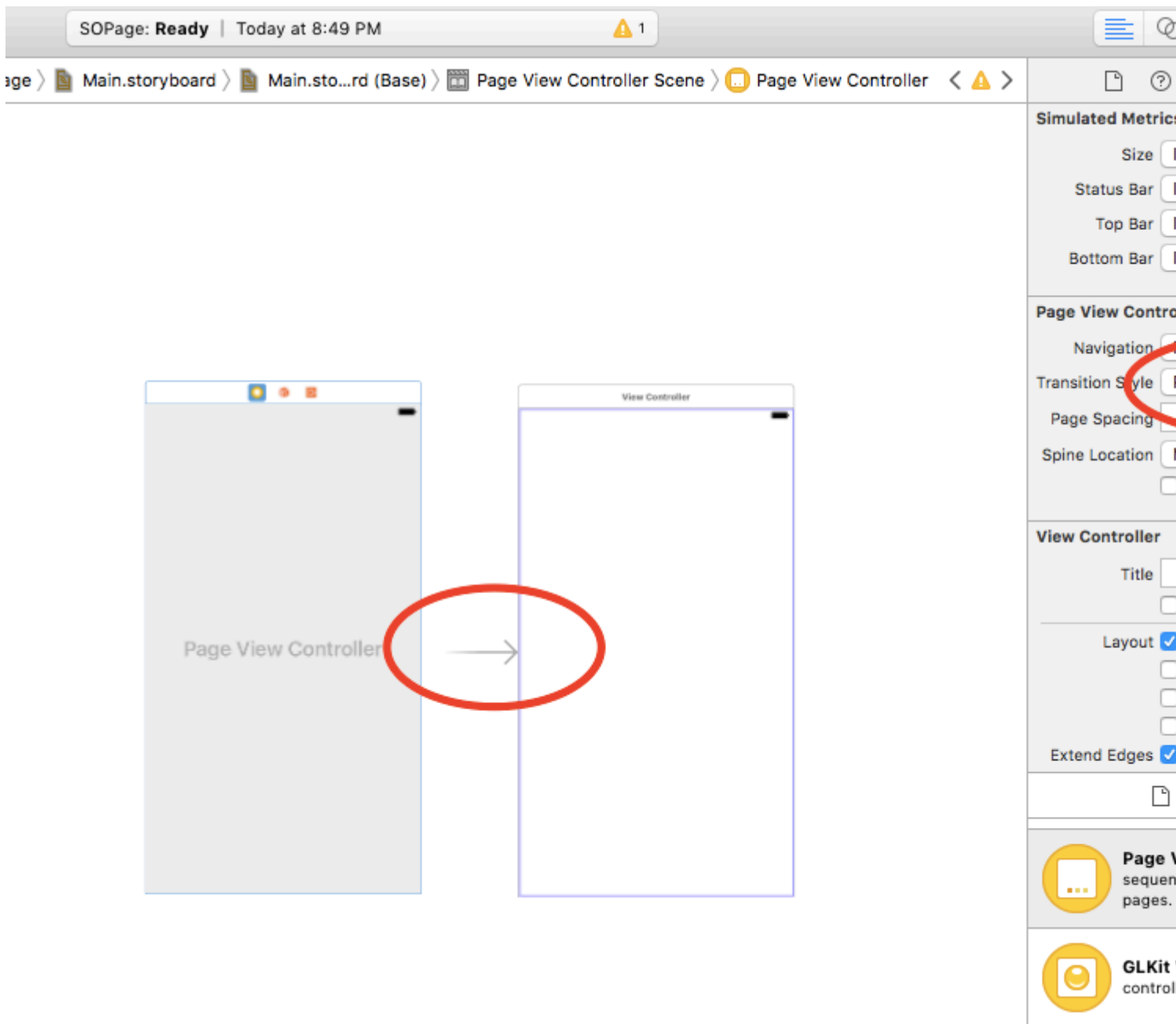
Eine einfache Möglichkeit zum Erstellen horizontaler Seitenansichts-Controller (unendliche Seiten)

1. Lassen Sie uns ein neues Projekt erstellen, wähle ich Single View Application zur besseren Demonstration



2. Ziehen Sie einen Controller für die Seitenansicht auf das Storyboard. Es gibt zwei Dinge, die Sie danach ändern sollten:
 1. Legen Sie den Seitensichtcontroller als anfänglichen Viewcontroller fest

2. Ändern Sie den Übergangsstil in einen Bildlauf



3. Sie müssen eine `UIPageViewController`-Klasse erstellen und diese dann als benutzerdefinierte Klasse des Seitensichtcontrollers im Storyboard festlegen
4. Fügen Sie diesen Code in Ihre `UIPageViewController`-Klasse ein. Sie sollten eine farbenfrohe, unendliche Seitenanwendung erhalten :)

```
class PageViewController: UIPageViewController, UIPageViewControllerDataSource {  
  
    override func viewDidLoad() {  
        self.dataSource = self  
        let controller = createViewController()  
        self.setViewControllers([controller], direction: .forward, animated: false,  
completion: nil)  
    }  
}
```

```

func pageViewController(_ pageViewController: UIPageViewController,
viewControllerBefore viewController: UIViewController) -> UIViewController? {
    let controller = createViewController()
    return controller
}

func pageViewController(_ pageViewController: UIPageViewController,
viewControllerAfter viewController: UIViewController) -> UIViewController? {
    let controller = createViewController()
    return controller
}

func createViewController() -> UIViewController {
    var randomColor: UIColor {
        return UIColor(hue: CGFloat(arc4random_uniform(360))/360, saturation: 0.5,
brightness: 0.8, alpha: 1)
    }
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    let controller = storyboard.instantiateViewController(withIdentifier: "View
Controller")
    controller.view.backgroundColor = randomColor
    return controller
}
}

```

So sieht das endgültige Projekt aus: Mit jedem Bildlauf erhalten Sie einen View-Controller mit unterschiedlichen Farben:

Kapitel 176: UIPheonix - einfaches, flexibles, dynamisches und hoch skalierbares UI-Framework

Einführung

Inspiziert von der Spieleentwicklung ist UIPheonix ein sehr einfaches, flexibles, dynamisches und hochskalierbares UI-Framework + -Konzept zum Erstellen wiederverwendbarer Komponenten- / Steuerungsanwendungen für Mac OS, iOS und TVOS. Die gleiche API gilt für die plattformübergreifende Entwicklung! Stellen Sie sich vor, Sie verwenden Lego-Blöcke. Sie können ähnliche Blöcke verwenden und sie einfach umherziehen.

<https://github.com/MKGitHub/UIPheonix>

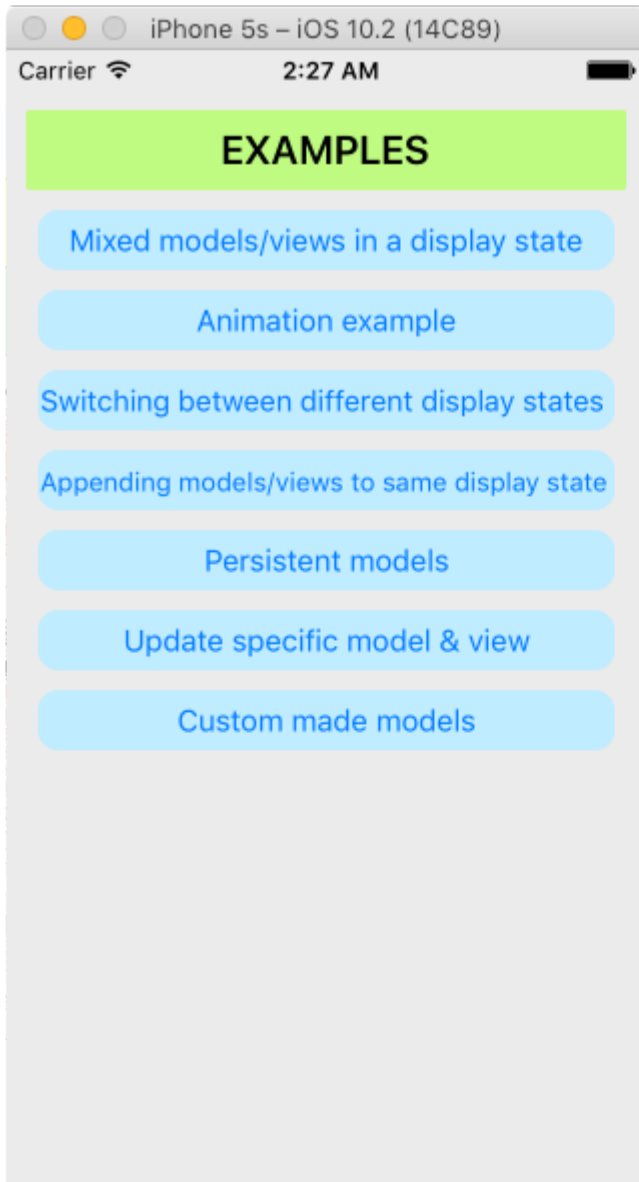
Bemerkungen

- Vergessen Sie statische Layouts, Einschränkungen und Warnmeldungen in der Konsole.
- Vergessen Sie den gesamten Klebercode, den gesamten Boilerplate-Code und all die übermäßig konstruierten unnötigen Haufen Müllcode in Ihren Apps.
- Erstellen und ändern Sie Ihre Benutzeroberfläche in kürzester Zeit.
- Machen Sie Ihre Benutzeroberfläche wieder verwendbar.
- Konzentrieren Sie sich auf die Erstellung Ihrer App, und vermeiden Sie Layoutprobleme.
- Minimales Setup, minimale Auswirkungen auf Ihre App, geringes Gewicht, keine Abhängigkeiten, keine Schmerzen, aber so viel Gewinn!
- Bauen Sie auf Sammlungsansichten und Tabellenansichten auf, sodass Sie sie leicht mischen und anpassen können.
- Ersetzt Apple-Technologien nicht durch benutzerdefinierte Implementierungen, sodass Sie immer sicher und auf dem neuesten Stand sind und jederzeit problemlos zurückkehren können.
- Demo-Apps für macOS, iOS und tvOS (Kung Fu!)

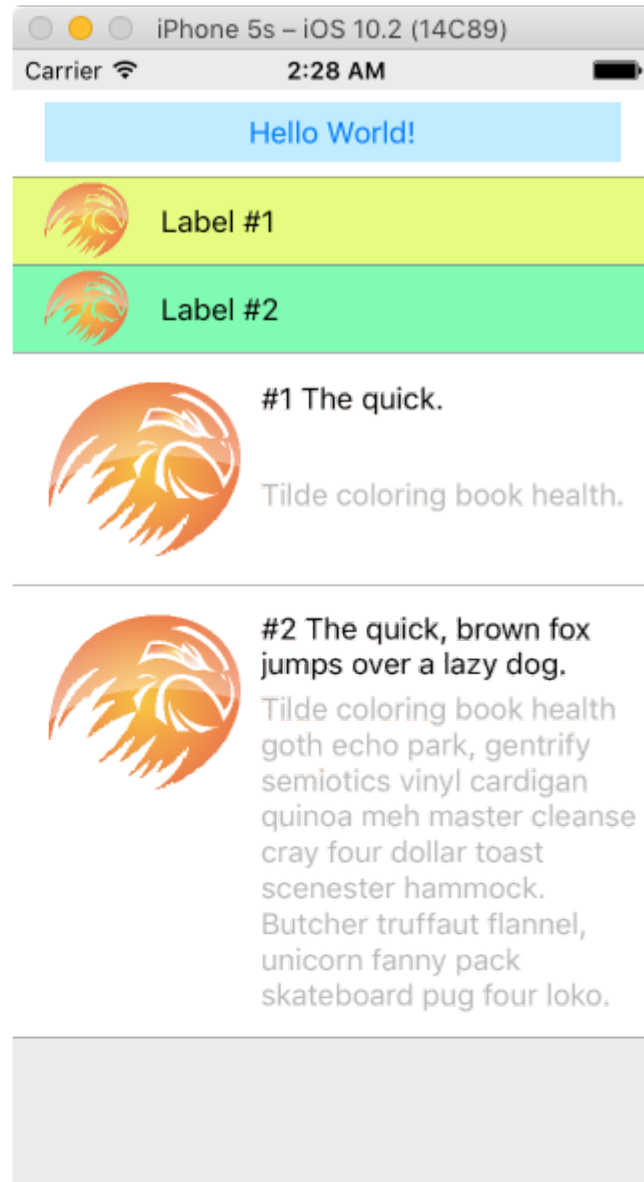
Examples

Beispiel-UI-Komponenten

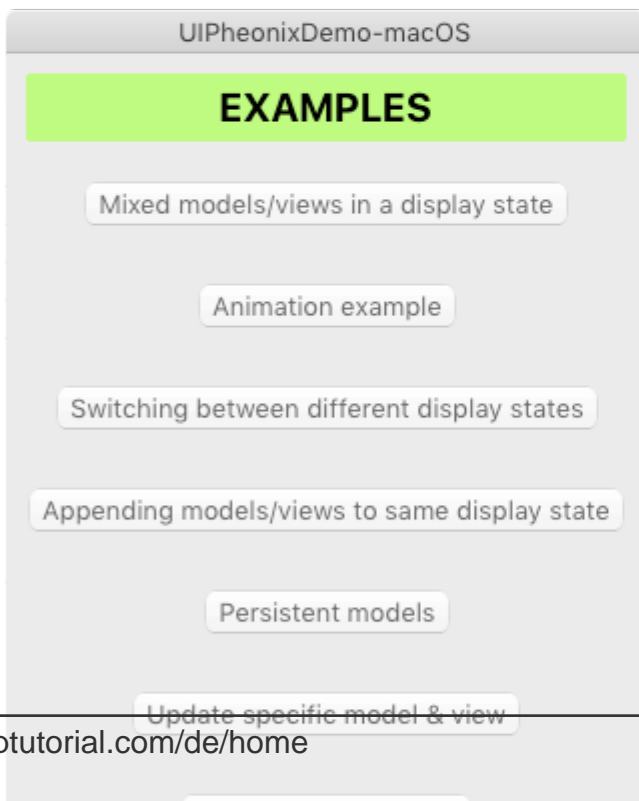
iOS - Collection View



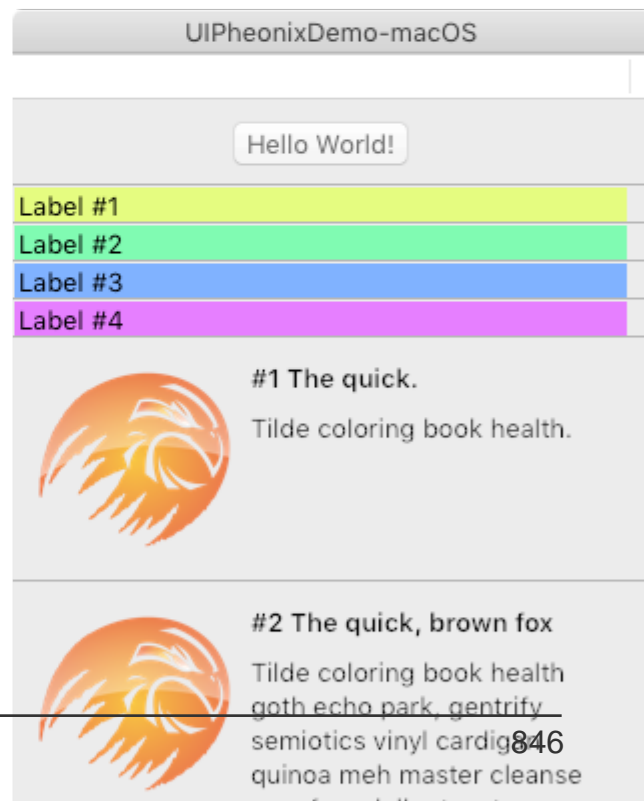
iOS - Table View



macOS - Collection View



macOS - Table View



<https://riptutorial.com/de/ios/topic/9120/uipeonix---einfaches--flexibles--dynamisches-und-hochskalierbares-ui-framework>

Kapitel 177: UIPickerView

Examples

Grundlegendes Beispiel

Schnell

```
class PickerViewExampleViewController : UIViewController, UIPickerViewDelegate,
UIPickerViewDataSource {
    @IBOutlet weak var btnFolder: UIButton!
    let pickerView = UIPickerView()
    let pickerViewRows = ["First row,", "Secound row,", "Third row,", "Fourth row"]

    override func viewDidLoad() {
        super.viewDidLoad()
        self.btnFolder.addTarget(self, action: #selector(CreateListVC.btnFolderPress),
forControlEvents: UIControlEvents.TouchUpInside)
    }

    @objc private func btnFolderPress() {
        self.pickerView.delegate = self
        self.pickerView.dataSource = self
        self.view.addSubview(self.pickerView)
    }

    //MARK: UIPickerViewDelegate

    func pickerView(pickerView: UIPickerView, titleForRow row: Int, forComponent component:
Int) -> String? {
        return self.pickerViewRows[row]
    }

    //MARK: UIPickerViewDataSource

    func numberOfComponentsInPickerView(pickerView: UIPickerView) -> Int {
        return 1
    }

    func pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
        return self.pickerViewRows.count
    }
}
```

Ziel c

```
@property (nonatomic, strong) UIPickerView *countryPicker;
@property (nonatomic, strong) NSArray *countryNames;

- (void)viewDidLoad {
```

```

    [super viewDidLoad];
    _countryNames = @[@"Australia (AUD)", @"China (CNY)",
                    @"France (EUR)", @"Great Britain (GBP)", @"Japan (JPY)", @"INDIA
(IN)", @"AUSTRALIA (AUS)", @"NEW YORK (NW)"];

    [self pickcountry];
}

-(void)pickcountry {
    _countryPicker = [[UIPickerView alloc] init];

    _countryPicker.delegate = self;
    _countryPicker.dataSource = self;

    [[UIPickerView appearance] setBackgroundColor:[UIColor colorWithRed:21/255.0
green:17/255.0 blue:50/255.0 alpha:1.0]];
}

#pragma mark- pickerView Delegates And datasource

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView {
    return 1;
}

- (NSInteger)pickerView:(UIPickerView *)pickerView
numberOfRowsInComponent:(NSInteger)component {
    return _countryNames.count;
}

- (NSString *)pickerView:(UIPickerView *)pickerView
titleForRow:(NSInteger)row
forComponent:(NSInteger)component {
    return _countryNames[row];
}

- (void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row
inComponent:(NSInteger)component {
    NSString *pickedCountryName = _countryNames[row];
}

```

PickerView-Hintergrundfarbe und Textfarbe ändern

Ziel c

```

//Displays the country pickerView with black background and white text
[self.countryPicker setValue:[UIColor whiteColor] forKey:@"textColor"];
[self.countryPicker setValue:[UIColor blackColor] forKey:@"backgroundColor"];

```

Schnell

```

let color1 = UIColor(colorLiteralRed: 1, green: 1, blue: 1, alpha: 1)
let color2 = UIColor(colorLiteralRed: 0, green: 0, blue: 0, alpha: 1)
pickerView2.setValue(color1, forKey: "textColor")
pickerView2.setValue(color2, forKey: "backgroundColor")

```

UIPickerView online lesen: <https://riptutorial.com/de/ios/topic/4242/uipickerview>

Kapitel 178: UIRefreshControl TableView

Einführung

Ein UIRefreshControl-Objekt stellt ein Standardsteuerelement bereit, mit dem die Aktualisierung des Inhalts einer Tabellenansicht initiiert werden kann. Sie verknüpfen ein Aktualisierungssteuerelement mit einer Tabelle über ein zugeordnetes Tabellensicht-Controller-Objekt. Der Table View-Controller übernimmt die Aufgabe, das Steuerelement dem visuellen Erscheinungsbild der Tabelle hinzuzufügen und die Anzeige dieses Steuerelements als Reaktion auf entsprechende Benutzergesten zu verwalten.

Examples

Ziel-C-Beispiel

Deklarieren Sie zunächst eine solche Eigenschaft im ViewController

```
@property (nonatomic) UIRefreshControl *refreshControl;
```

Später in `viewDidLoad()` richten Sie das `refreshControl` wie folgt ein:

```
self.refreshControl = [[UIRefreshControl alloc] init];
[self.tableView addSubview:self.refreshControl];
[self.refreshControl addTarget:self action:@selector(refreshTable)
forControlEvents:UIControlEventValueChanged];
//Setting the tint Color of the Activity Animation
self.refreshControl.tintColor = [UIColor redColor];
//Setting the attributed String to the text
NSMutableAttributedString * string = [[NSMutableAttributedString alloc]
initWithString:@"firstsecondthird"];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor redColor]
range:NSMakeRange(0, 5)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(5, 6)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor blueColor]
range:NSMakeRange(11, 5)];
self.refreshControl.attributedString = string;
```

Nun ist die Funktion `refreshTable` definiert als:

```
- (void)refreshTable {
    //TODO: refresh your data
    [self.refreshControl endRefreshing];
    [self.refreshControl beginRefreshing];
    [self.tableView reloadData];
    [self.refreshControl endRefreshing];
}
```



RefreshControl für tableView einrichten:

```
UIRefreshControl *refreshControl = [[UIRefreshControl alloc] init];
[refreshControl addTarget:self action:@selector(pullToRefresh:)
forControlEvents:UIControlEventValueChanged];
self.scrollView.alwaysBounceVertical = YES;
[self.scrollView addSubview:refreshControl];

- (void)pullToRefresh:(UIRefreshControl*) sender{
//Do work off the main thread
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
// Simulate network traffic (sleep for 2 seconds)
[NSThread sleepForTimeInterval:2];
//Update data
//Call complete on the main thread
dispatch_sync(dispatch_get_main_queue(), ^{
//Update network activity UI
NSLog(@"COMPLETE");
[sender endRefreshing];
});
});
}
```

UIRefreshControl TableView online lesen: <https://riptutorial.com/de/ios/topic/8278/uirefreshcontrol-tableview>

Kapitel 179: UIScrollView

Examples

Erstellen Sie eine UIScrollView

Erstellen Sie eine Instanz von `UIScrollView` mit einem `CGRect` als Frame.

Schnell

```
let scrollView = UIScrollView.init(frame: CGRect(x: 0, y: 0, width: 320, height: 400))
```

Ziel c

```
UIScrollView *scrollView = [[UIScrollView alloc] initWithFrame:CGRectMake(0, 0, 320, 400)];
```

Blättern Sie in die Inhaltsgröße

Die `contentSize` Eigenschaft muss auf die Größe des scrollbaren Inhalts festgelegt werden. Dies gibt die Größe des scrollbaren Bereichs an. Bildlauf ist sichtbar, wenn der bildlauffähige Bereich, dh `contentSize` größer ist als die `UIScrollView`.

Mit Autolayout:

Wenn der Inhalt der Bildlaufansicht mit Autolayout eingerichtet wird, muss er explizit sowohl vertikal als auch horizontal dimensioniert sein und alle 4 Kanten müssen an die Bildlaufansicht angeheftet sein. Auf diese Weise wird `contentSize` automatisch basierend auf dem Inhalt der `contentSize` berechnet und auch aktualisiert, wenn das Layout des Inhalts geändert wird.

Manuell:

Schnell

```
scrollView.contentSize = CGSize(width: 640, height: 800)
```

Ziel c

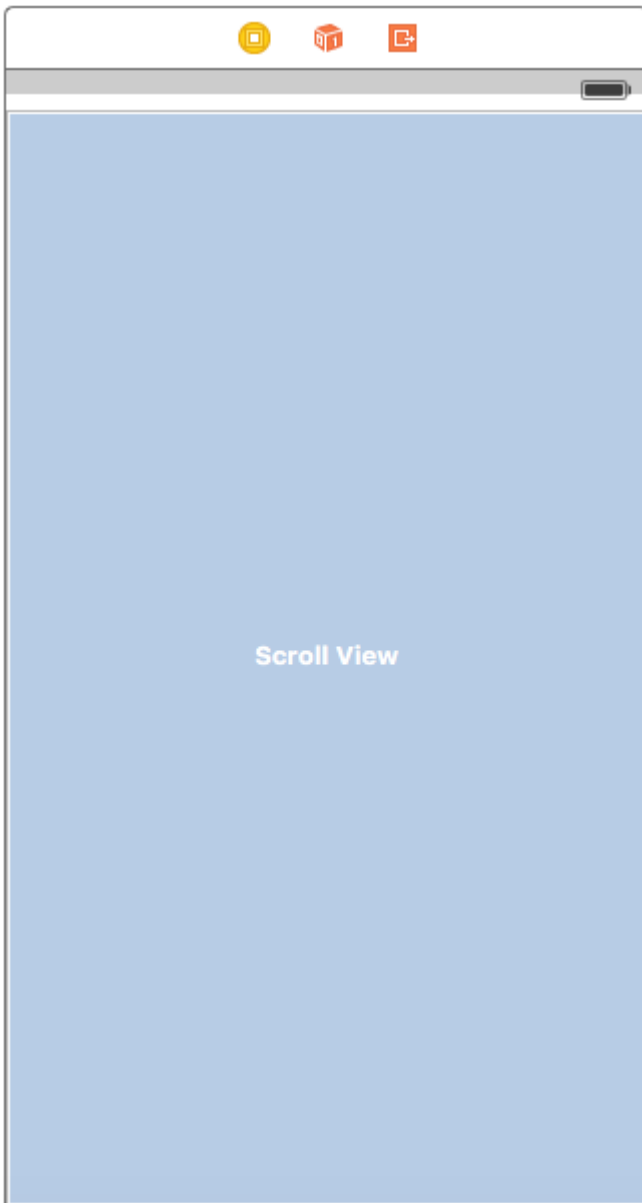
```
scrollView.contentSize = CGSizeMake(640, 800);
```

ScrollView mit AutoLayout

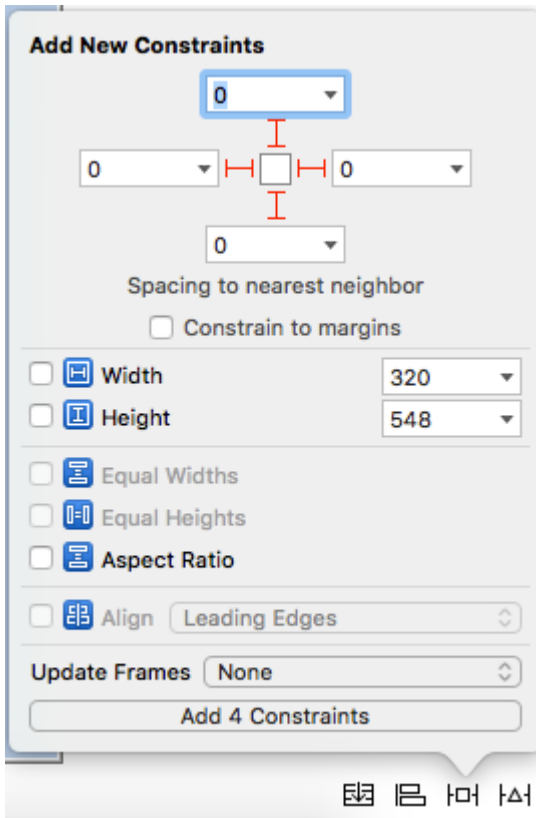
Einfache Schritte zur Verwendung von ScrollView mit Autolayout.

- Erstellen Sie ein neues Projekt mit Einzelansicht
- Wählen Sie den Standard-Viewcontroller aus, und ändern Sie seine Bildschirmgröße vom Attribute-Inspector in iPhone-4inch.

- Fügen Sie der Ansicht Ihres Viewcontrollers wie folgt eine Bildlaufansicht hinzu und setzen Sie die Hintergrundfarbe auf Blau



- Fügen Sie Einschränkungen hinzu, wie im Bild unten gezeigt



Dazu müssen Sie einfach jede Kante des Bildlaufs in die Ansicht des Viewcontrollers stecken

Szenario 1:

Nun sagen wir, unser Inhalt ist riesig und wir möchten, dass er sowohl horizontal als auch vertikal scrollt.

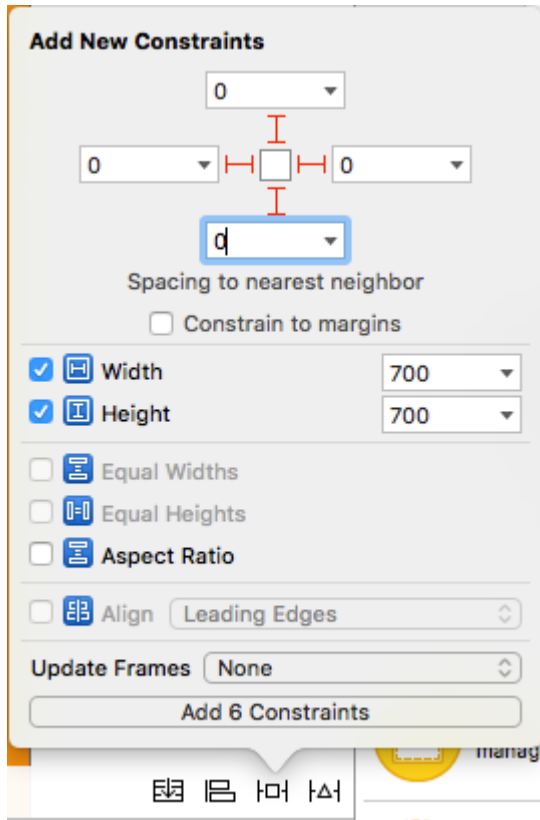
Dafür,

- Fügen Sie der Bildlaufansicht des Rahmens (0,0,700,700) eine UIView hinzu. Erlaubt es orange Hintergrundfarbe, um ihn anders zu identifizieren.



Als nächstes kommt der wichtige Teil, wir müssen ihn horizontal und vertikal scrollen.

- Wählen Sie die orange Ansicht aus und fügen Sie die folgenden Einschränkungen hinzu



Lassen Sie mich erklären, was wir oben gemacht haben.

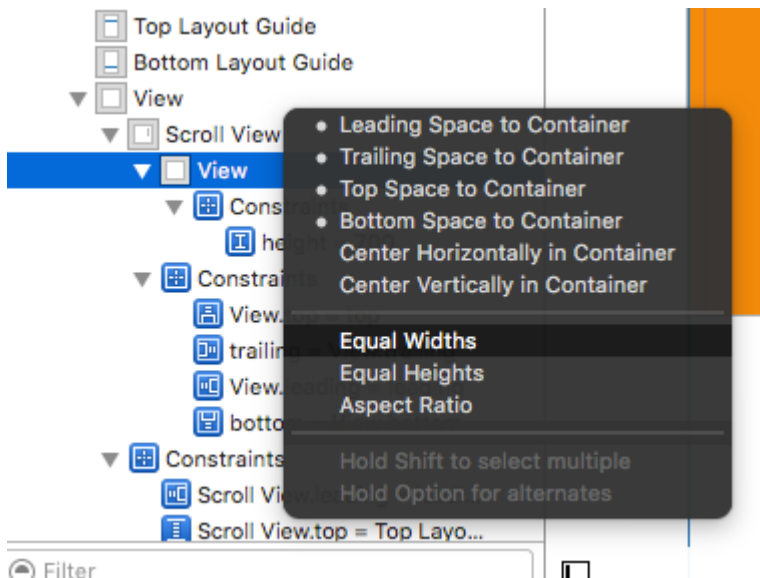
- Wir haben die Höhe und Breite auf 700 festgelegt.
- Der nachstehende Speicherplatz wird auf scrollview = 0 gesetzt, wodurch dem ScrollView mitgeteilt wird, dass der Inhalt horizontal scrollbar ist.
- Der untere Bereich wird auf scrollview = 0 gesetzt, wodurch dem ScrollView mitgeteilt wird, dass der Inhalt vertikal scrollbar ist.

Führen Sie nun das Projekt aus und überprüfen Sie es.

Szenario 2: Betrachten wir ein Szenario, in dem wir wissen, dass die Breite des Inhalts der Bildlaufbreite entspricht, aber die Höhe größer ist als die Bildlaufansicht.

Folgen Sie den Schritten, um den Inhalt vertikal zu scrollen.

- Löschen Sie die Breitenbeschränkung im obigen Fall.
- Ändern Sie die Breite der orangefarbenen Ansicht, um sie an die Breite der Bildlaufansicht anzupassen.
- Ziehen Sie bei gedrückter Strg-Taste aus der orangefarbenen Ansicht, um eine Bildlaufansicht zu erstellen und die Einschränkung für die **gleiche Breite** hinzuzufügen.



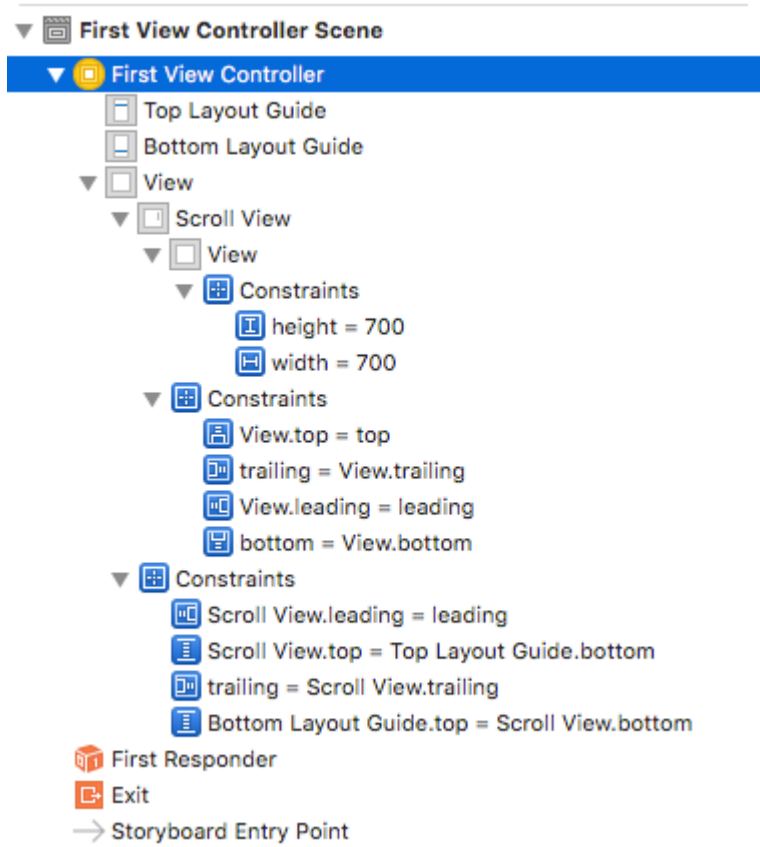
- Und fertig!!! Führen Sie einfach aus und prüfen Sie, ob es vertikal scrollt

Szenario 3:

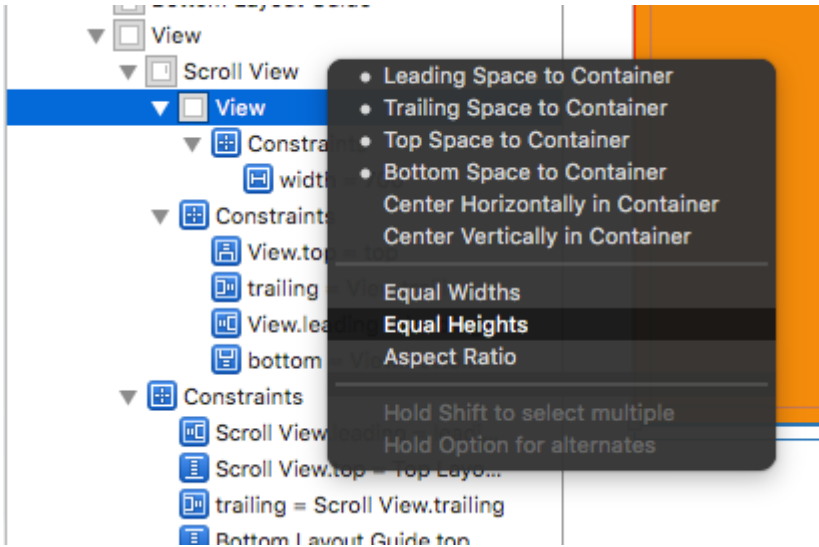
Jetzt wollen wir nur horizontal und nicht vertikal scrollen.

Folgen Sie den Schritten, um den Inhalt horizontal zu scrollen.

- Machen Sie alle Änderungen rückgängig, um die folgenden Einschränkungen zu erreichen (dh **stellen Sie die ursprünglichen Einschränkungen wieder her, die einen vertikalen und horizontalen Bildlauf ergeben**).



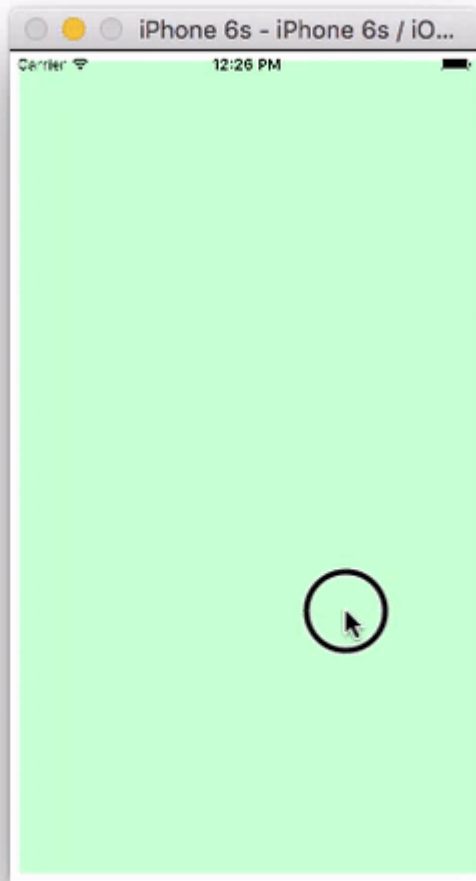
- Überprüfen Sie den orangefarbenen Sichtrahmen, der (0,0,700,700) sein sollte.
- Löschen Sie die Höhenbeschränkung der orange Ansicht.
- Ändern Sie die Höhe der orangefarbenen Ansicht entsprechend der Höhe der Bildlaufansicht.
- Ziehen Sie bei gedrückter Strg-Taste aus der orangefarbenen Ansicht, um die Ansicht zu scrollen und die Beschränkung für **gleiche Höhen** hinzuzufügen.



- Und fertig!!! Führen Sie einfach aus und prüfen Sie, ob es vertikal scrollt

Bildlauf mit aktiviertem automatischen Layout

Dieses Projekt ist ein eigenständiges Beispiel, das vollständig im Interface Builder erstellt wurde. Sie sollten es in 10 Minuten oder weniger durcharbeiten können. Dann können Sie die erlernten Konzepte auf Ihr eigenes Projekt anwenden.



Hier verwende ich nur `UIView`s, aber sie können jede beliebige Ansicht darstellen (z. B. Schaltfläche, Beschriftung usw.). Ich habe mich auch für das horizontale Scrollen entschieden, da die Storyboard-Screenshots für dieses Format kompakter sind. Die Prinzipien sind jedoch beim vertikalen Scrollen gleich.

Schlüssel Konzepte

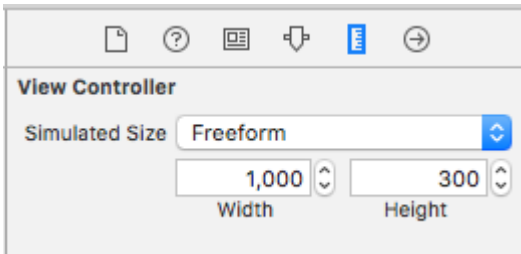
- Die `UIScrollView` sollte nur eine Unteransicht verwenden. Dies ist eine `UIView`, die als Inhaltsansicht für alles dient, was Sie scrollen möchten.
- Stellen Sie sicher, dass die Inhaltsansicht und die *übergeordnete* Ansicht der Bildlaufansicht für den horizontalen Bildlauf gleich hoch sind. (Gleiche Breiten für vertikales Scrollen)
- Stellen Sie sicher, dass der gesamte scrollbare Inhalt eine festgelegte Breite hat und an allen Seiten gepinnt ist.

Starten Sie ein neues Projekt

Es kann nur eine einzige Ansichtsanwendung sein.

Storyboard

In diesem Beispiel erstellen wir eine horizontale Bildlaufansicht. Wählen Sie den Ansichts-Controller aus und wählen Sie dann im Größeninspektor die Option Freiform. Machen Sie die Breite 1,000 und die Höhe 300 . Dies gibt uns nur den Raum auf dem Storyboard, um Inhalte hinzuzufügen, die durchlaufen werden.



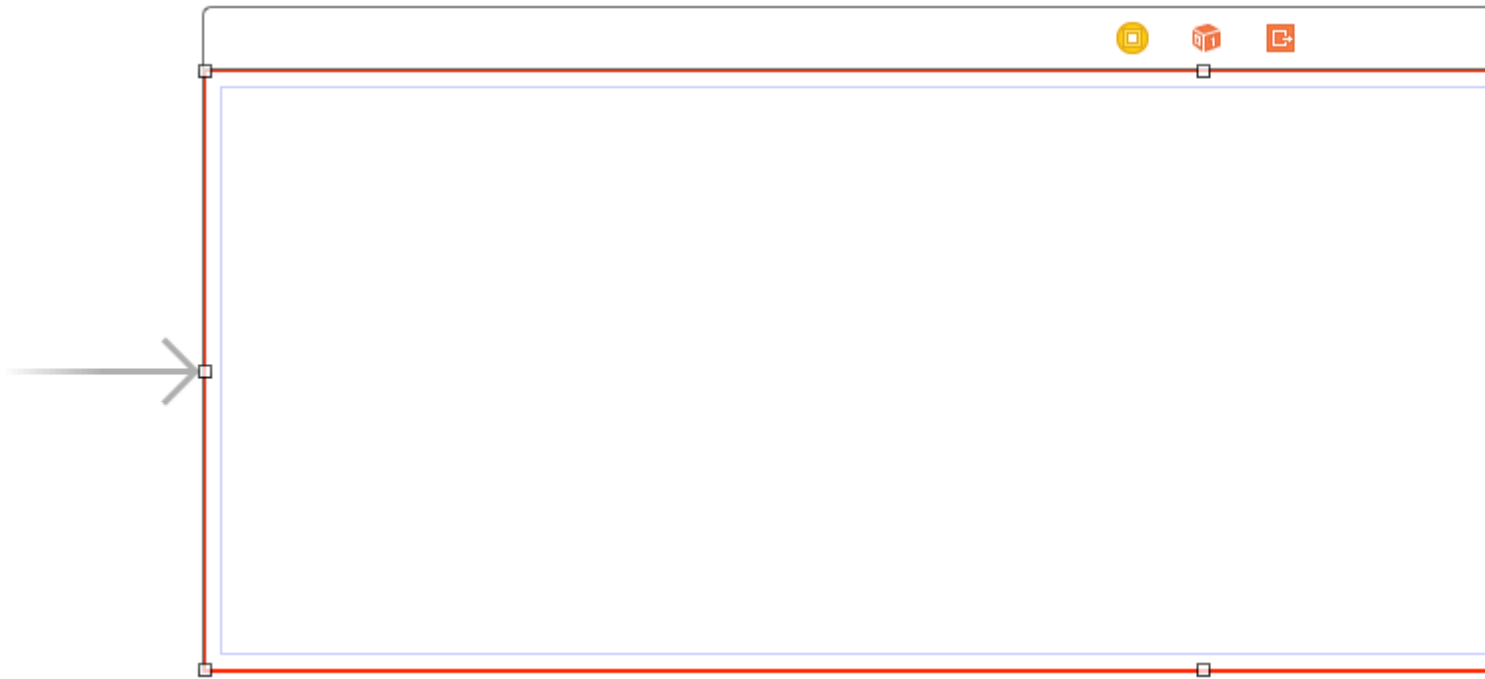
Fügen Sie eine Bildlaufansicht hinzu

Fügen Sie eine `UIScrollView` und verbinden Sie alle vier Seiten mit der `UIScrollView` des View-Controllers.



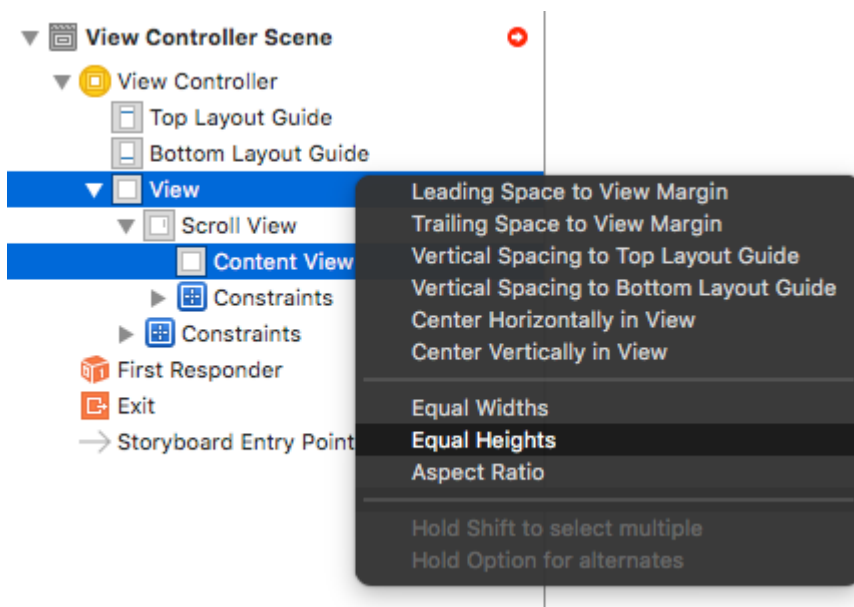
Fügen Sie eine Inhaltsansicht hinzu

Fügen Sie der `UIView` eine `UIView` als Unteransicht hinzu. *Das ist der Schlüssel.* Versuchen Sie nicht, viele Unteransichten zur Bildlaufansicht hinzuzufügen. `UIView` einfach eine einzelne `UIView` . Dies ist Ihre Inhaltsansicht für die anderen Ansichten, die Sie scrollen möchten. Verbinden Sie die Inhaltsansicht auf allen vier Seiten mit der Bildlaufansicht.



Gleiche Höhen

Jetzt in der Dokumentgliederung, klicken Sie auf `Befehl` sowohl den Inhalt Ansicht und der übergeordneten Ansicht der Ansicht blättern, um sie beide zu wählen. Stellen Sie dann die Höhen auf gleich ein (ziehen Sie `<Control </code> von der Inhaltsansicht in die Bildlaufansicht>). Dies ist auch der Schlüssel. Da wir horizontal scrollen, weiß die Inhaltsansicht der Bildlaufansicht nicht, wie hoch sie sein soll, wenn wir sie nicht auf diese Weise festlegen.`

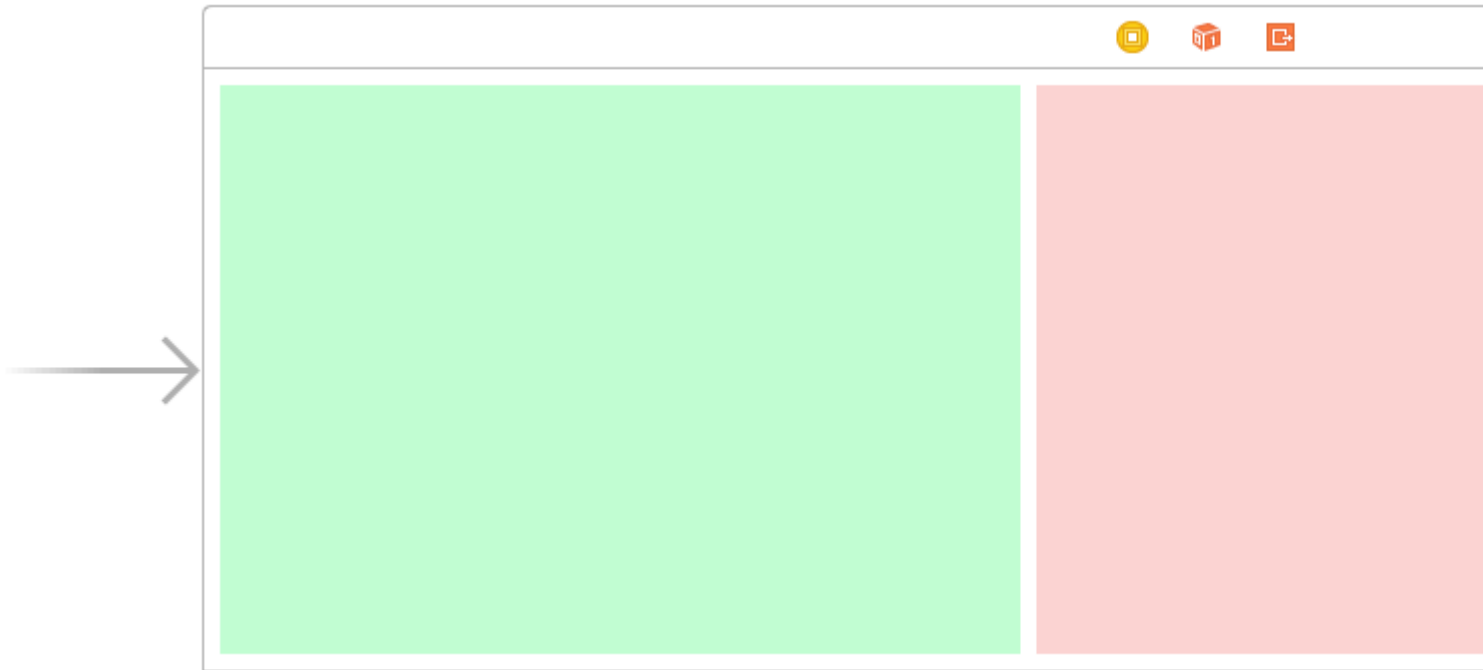


Hinweis:

- Wenn wir den Inhalt vertikal scrollen lassen, würden wir die Breite der Inhaltsansicht so einstellen, dass sie der Breite der übergeordneten Ansicht der Bildlaufansicht entspricht.

Inhalt hinzufügen

Fügen Sie drei `UIView` und geben Sie ihnen alle Einschränkungen. Ich habe 8 Punkte für alles verwendet.



Einschränkungen:

- Grüne Ansicht: Pin oben, links und unten. Machen Sie die Breite 400.
- Rote Ansicht: Pin oben, links und unten. Machen Sie die Breite 300.
- Lila Ansicht: Alle vier Kanten anheften. Stellen Sie die Breite so ein, wie der verbleibende Platz ist (in diesem Fall 268).

Das Festlegen der Breitenbeschränkungen ist auch wichtig, damit die Bildlaufansicht weiß, wie breit die Inhaltsansicht ist.

Fertig

Das ist alles. Sie können Ihr Projekt jetzt ausführen. Es sollte sich wie das Bildlaufbild oben in dieser Antwort verhalten.

Weitere Studie

- [iOS: So machen Sie AutoLayout für ein UIScrollView](#)
- [So konfigurieren Sie eine UIScrollView mit automatischem Layout im Interface Builder](#)
- [YouTube-Video-Tutorial: UIScrollView - So behalten Sie Ihre Ansichten auf dem Bildschirm](#)

Bildlauf aktivieren / deaktivieren

Die Eigenschaft `scrollEnabled` speichert einen `Boolean` Wert, der bestimmt, ob der Bildlauf aktiviert

ist oder nicht.

Wenn der Wert dieser Eigenschaft `true / YES` ist, ist der Bildlauf aktiviert, andernfalls nicht. Der Standardwert ist `true`

Schnell

```
scrollView.isEnabled = true
```

Ziel c

```
scrollView.scrollEnabled = YES;
```

Vergrößern / Verkleinern von UIImageView

Erstellen Sie eine UIScrollView-Instanz

```
let scrollView = UIScrollView.init(frame: self.view.bounds)
```

Und dann diese Eigenschaften einstellen:

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
scrollView.delegate = self as? UIScrollViewDelegate
```

Um das Bild zu vergrößern oder zu verkleinern, müssen Sie den Betrag angeben, um den der Benutzer zoomen kann. `minimumZoomScale` `maximumZoomScale` sind Werte für die Eigenschaften `minimumZoomScale` und `maximumZoomScale` der `UIScrollView`. Beide sind standardmäßig auf 1,0 eingestellt.

Und `zoomScale` auf 1,0, die den Zoomfaktor für den minimalen und maximalen Zoom angeben.

Um das Zoomen zu unterstützen, müssen wir einen Delegaten für Ihre Bildlaufansicht festlegen. Das Delegate-Objekt muss dem `UIScrollViewDelegate` Protokoll entsprechen. Diese `viewForZoomingInScrollView()` muss die `viewForZoomingInScrollView()` Methode implementieren und die Ansicht zum Zoomen zurückgeben.

Ändern Sie Ihren ViewController wie gezeigt

```
class ViewController: UIViewController, UIScrollViewDelegate
```

Fügen Sie dann der Klasse die folgende Delegatfunktion hinzu.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```


Erstellen Sie nun die UIImageView-Instanz

Machen Sie diese Variable als Klassenvariable

```
var imageView:UIImageView = UIImageView.init(image: UIImage.init(named: "someImage.jpg"))
```

Und fügen Sie es dann zum Bildlauf hinzu

```
scrollView?.addSubview(imageView)
```

Referenz

- [Scroll View Programmierhandbuch für iOS](#)
- [UIScrollView-Tutorial](#)

Ermitteln, wann der Bildlauf von UIScrollView mit Delegat-Methoden abgeschlossen ist

scrollViewDidEndDecelerating: Dies teilt dem Delegierten mit, dass die Bildlaufansicht beendet ist und die Bildlaufbewegung verlangsamt hat.

Ziel c:

```
- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {
    [self stoppedScrolling];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate {
    if (!decelerate) {
        [self stoppedScrolling];
    }
}

- (void)stoppedScrolling {
    // done, do whatever
}
```

Schnell:

```
func scrollViewDidEndDragging(scrollView: UIScrollView, willDecelerate decelerate: Bool) {
    if !decelerate {
        stoppedScrolling()
    }
}

func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
    stoppedScrolling()
}
```

```
func stoppedScrolling() {
    // done, do whatever
}
```

Bildlaufrichtung einschränken

Sie können die Richtungen einschränken, zu denen der Benutzer scrollen kann, indem Sie folgenden Code verwenden:

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if scrollView.contentOffset.x != 0 {
        scrollView.contentOffset.x = 0
    }
}
```

Bei jedem Bildlauf des Benutzers auf der x-Achse wird der Inhaltsversatz der scrollView auf 0 zurückgesetzt.

Sie können natürlich die Änderung x s y s und die Richtung dafür sperren Horizontal nur zu sein.

Sie müssen auch sicherstellen, dass Sie diesen Code in die Delegat-Methode

`scrollViewDidScroll(_ scrollView: UIScrollView)` . Andernfalls wird es nicht funktionieren.

`UIScrollViewDelegate` Sie außerdem sicher, dass Sie `UIScrollViewDelegate` wie `UIScrollViewDelegate` in Ihre Klassendeklaration importiert haben:

```
class ViewController: UIViewController, UIScrollViewDelegate
```

... und setzen Sie den Delegaten der scrollView in einer Methode wie `viewDidLoad(_:)`

```
scrollView.delegate = self
```

UIScrollView online lesen: <https://riptutorial.com/de/ios/topic/1575/uiscrollview>

Kapitel 180: UIScrollView AutoLayout

Examples

ScrollableController

Wenn Sie Autolayout mit einer `UIScrollView`, wird die Größe je nach Größe des Inhalts oder der Unteransichten NICHT korrekt angepasst.

Damit ein `UIScrollView` automatisch scrollen kann, wenn sein Inhalt zu groß wird, um in den sichtbaren Bereich zu passen, müssen ein `ContentView` und einige Einschränkungen `UIScrollView`, die es dem `UIScrollView` ermöglichen, die Größe des Inhalts UND dessen Breite und Höhe im übergeordneten `UIScrollView` zu bestimmen Aussicht.

```
import Foundation
import UIKit

class ScrollableController : UIViewController {

    private var scrollView: UIScrollView!
    private var contentView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        //Setup
        self.initControls()
        self.setTheme()
        self.layoutScrollView()
        self.layoutContentView()

        //Add child views
        self.addChildViews()
    }

    func initControls() {
        self.scrollView = UIScrollView()
        self.contentView = UIView()
    }

    func setTheme() {
        self.scrollView.backgroundColor = UIColor.blue()
        self.contentView.backgroundColor = UIColor.orange()
    }

    func layoutScrollView() {
        self.view.addSubview(self.scrollView)

        let views: NSDictionary = ["scrollView": self.scrollView]
        var constraints = Array<String>()

        //Constrain the scrollView to our controller's self.view.
        constraints.append("H:|-0-[scrollView]-0-|")
        constraints.append("V:|-0-[scrollView]-0-|")
    }
}
```

```

        for constraint in constraints {
            self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
        }

        self.scrollView.translatesAutoresizingMaskIntoConstraints = false
    }

    func layoutContentView() {
        self.scrollView.addSubview(self.contentView)

        let views: NSDictionary = ["contentView": self.contentView, "view": self.view]
        var constraints = Array<String>()

        //Constrain the contentView to the scrollView.
        constraints.append("H:|-0-[contentView]-0-|")
        constraints.append("V:|-0-[contentView]-0-|")

        for constraint in constraints {
            self.scrollView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
        }

        //Disable Horizontal Scrolling by making the contentView EqualWidth with our
controller's self.view (ScrollView's parentView).
        self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
"H:[contentView(==view)]", options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views:
views as! [String : AnyObject]))

        self.contentView.translatesAutoresizingMaskIntoConstraints = false
    }

    func addChildViews() {
        //Init
        let greenView = UIView()
        let whiteView = UIView()

        //Theme
        greenView.backgroundColor = UIColor.green()
        whiteView.backgroundColor = UIColor.orange()

        //Layout -- Child views are added to the 'ContentView'
        self.contentView.addSubview(greenView)
        self.contentView.addSubview(whiteView)

        let views: NSDictionary = ["greenView": greenView, "whiteView": whiteView];
        var constraints = Array<String>()

        //Constrain the greenView to the contentView with a height of 400 and 15 spacing all
around.
        constraints.append("H:|-15-[greenView]-15-|")
        constraints.append("V:|-15-[greenView(400)]")

        //Constrain the whiteView below the greenView with 15 spacing all around and a height
of 500.
        constraints.append("H:|-15-[whiteView]-15-|")
        constraints.append("V:[greenView]-15-[whiteView(500)]-15-|")

        for constraint in constraints {

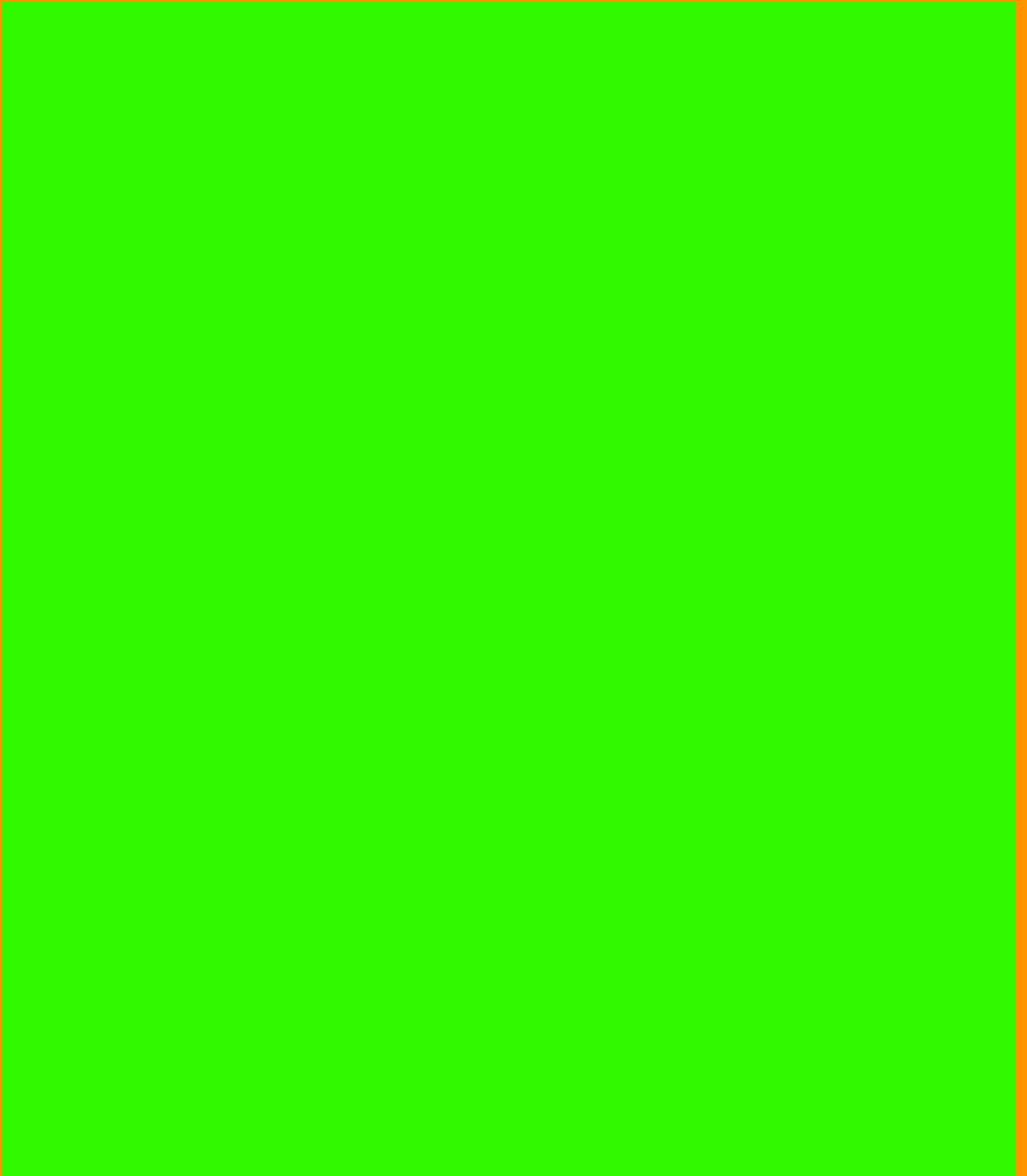
```

```
        self.contentView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
    }

    greenView.translatesAutoresizingMaskIntoConstraints = false
    whiteView.translatesAutoresizingMaskIntoConstraints = false
}
}
```

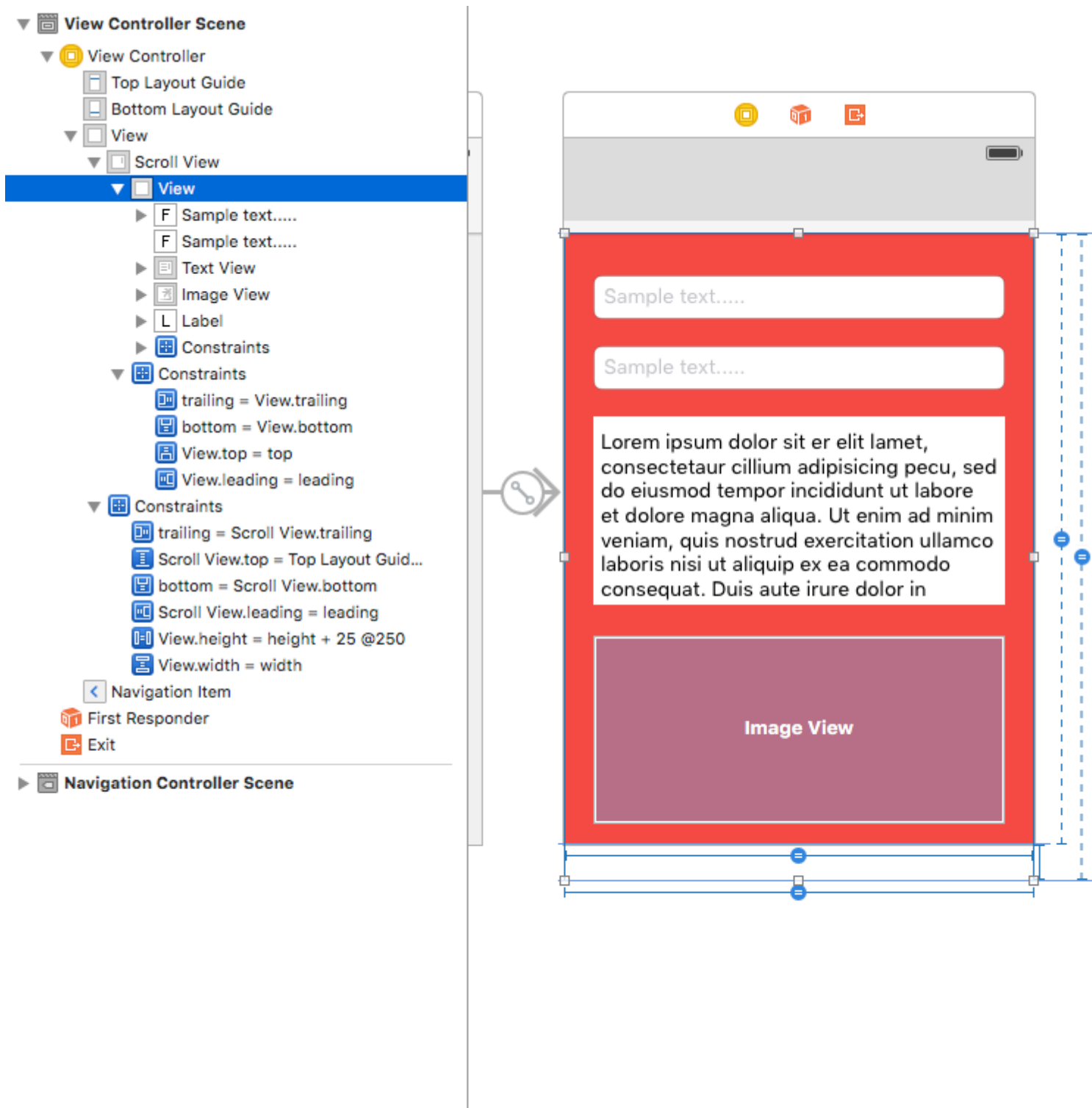
Jetzt können wir sehen, dass das `greenView` (Höhe 400) + das `WhiteView` (Höhe 500) größer ist als unser Bildschirm. Dadurch wird die `contentSize` von `ScrollView` auf **BEIDE** Ansichten angepasst, sodass sie vertikal scrollen kann.

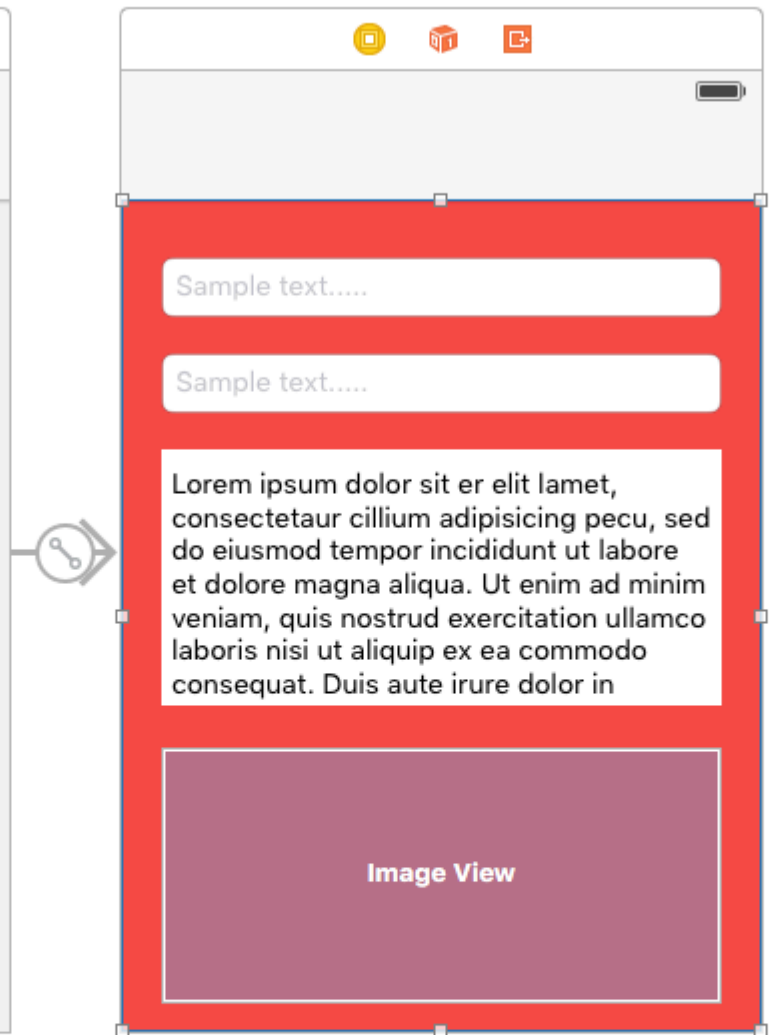
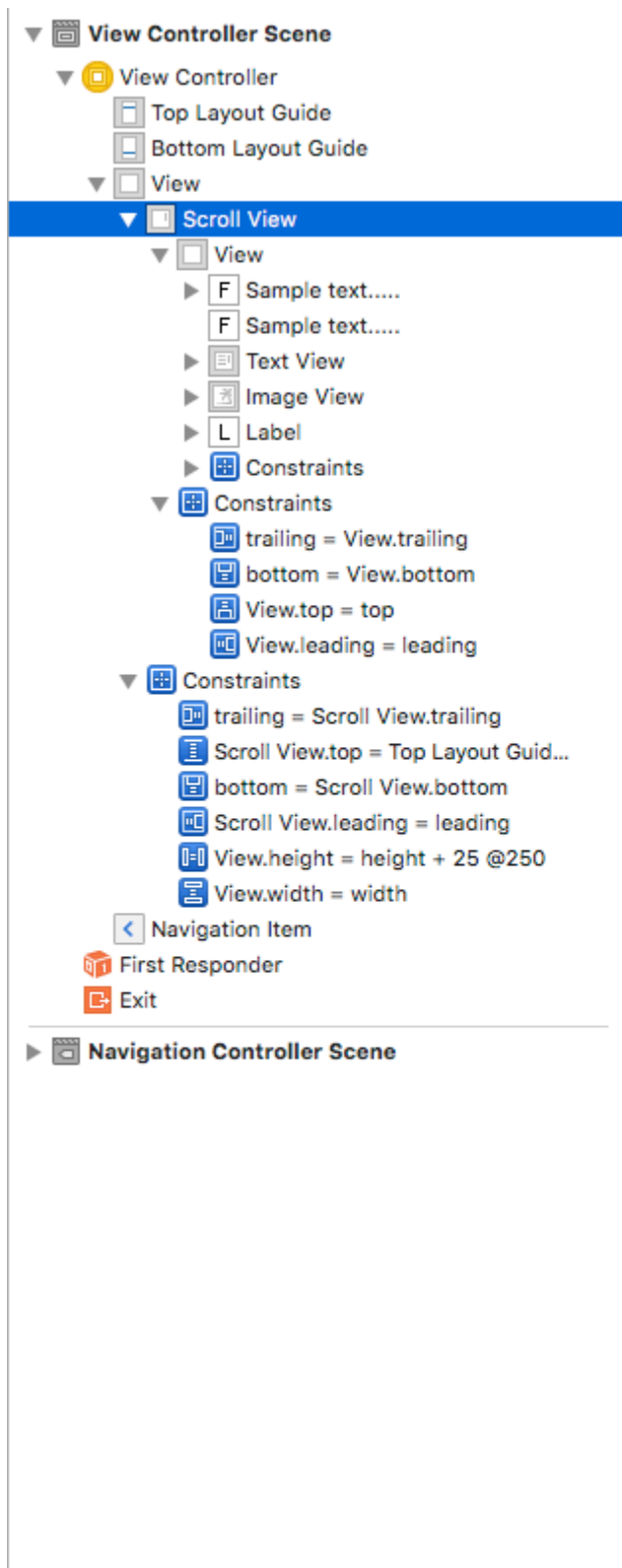
Wir haben den horizontalen Bildlauf mithilfe der `EqualWidth` Einschränkung für `contentView` und `self.view`



2. Fügen Sie der Hauptansicht die gleiche Höhe und die gleiche Breite hinzu (dh, die die Bildlaufansicht enthält). Für gleiche Höhe die Priorität auf niedrig setzen. (Dies ist der wichtige Schritt zum Einstellen der Inhaltsgröße).
3. Die Höhe dieser Inhaltsansicht richtet sich nach der Anzahl der Ansichten, die der Ansicht hinzugefügt wurden. Angenommen, Sie haben die letzte Ansicht hinzugefügt, ist eine Beschriftung und seine Y-Position ist 420 und die Höhe ist 20, dann beträgt Ihre Inhaltsansicht 440.

Schritt 3: Fügen Sie Einschränkungen für alle Ansichten hinzu, die Sie je nach Anforderung in der Inhaltsansicht hinzugefügt haben.





UIScrollView AutoLayout online lesen: <https://riptutorial.com/de/ios/topic/4671/uiscrollview-autolayout>

Kapitel 181: UIScrollView mit StackView-Kind

Examples

Eine komplexe StackView in Scrollview Example

Es folgt ein Beispiel dafür, was mit verschachtelten StackViews möglich ist, und vermittelt dem Benutzer den Eindruck eines kontinuierlichen Bildlaufs, bei dem komplexe Benutzeroberflächenelemente oder -ausrichtungen verwendet werden.



Mehrdeutiges Layout vermeiden

Eine häufige Frage zu StackViews innerhalb von Scrollviews stammt aus mehrdeutigen / heigh-
Warnungen im Interface Builder. Wie [diese Antwort](#) erklärt hat, ist es notwendig:

1. Fügen Sie in der UIScrollView eine UIView (die contentScrollView) hinzu.
2. Setzen Sie in dieser contentScrollView den oberen, unteren, linken und rechten Rand auf 0
3. Richten Sie das Zentrum auch horizontal und vertikal aus.

In verschachtelten StackViews zum Inhalt blättern

Das große Problem beim Scrollen besteht darin, den Versatz zu bestimmen, der erforderlich ist, um ein **Textfeld in einer StackView darzustellen, das sich in der ScrollView befindet** .

Wenn Sie versuchen, **die Position von `TextField.frame.minY` kann dies 0 sein** , da der minY-Frame nur den Abstand zwischen dem Element und der Oberseite der StackView berücksichtigt. Sie müssen also **alle anderen übergeordneten Stackviews / Views betrachten**.

Eine gute Lösung für dieses Problem ist:

1 - Implementieren Sie die ScrollView-Erweiterung

```
extension UIScrollView {  
  
    func scrollToShowView(view: UIView){  
        var offset = view.frame.minY  
        var superview = view.superview  
        while((superview != nil)){  
            offset += (superview?.frame.minY)!  
            superview = superview?.superview  
        }  
  
        offset -= 100 //optional margin added on offset  
  
        self.contentOffset = CGPoint.init(x: 0, y: offset)  
    }  
  
}
```

Dies berücksichtigt alle übergeordneten Ansichten und summiert den erforderlichen Versatz für die Bildlaufansicht, um die erforderliche Ansicht auf dem Bildschirm darzustellen (z. B. ein Textfeld, das nicht hinter der Benutzertastatur bleiben kann).

Anwendungsbeispiel:

```
func textViewDidBeginEditing(_ textView: UITextView) {  
    self.contentScrollView.scrollToShowView(view: textView)  
}
```

UIScrollView mit StackView-Kind online lesen: <https://riptutorial.com/de/ios/topic/9404/uiscrollview-mit-stackview-kind>

Kapitel 182: UISearchController

Syntax

- `UISearchController (searchResultsController: UIViewController?)` // Geben Sie nil als Parameter an, wenn der Suchaktualisierungscontroller auch den durchsuchbaren Inhalt anzeigt.
- `func updateSearchResults (für searchController: UISearchController)` // Erforderliche Methode zur Implementierung bei der Übernahme des `UISearchResultsUpdating`-Protokolls

Parameter

Parameter	Einzelheiten
<code>UISearchController.searchBar</code>	Die Suchleiste, die in Ihrer Benutzeroberfläche installiert werden soll. (<i>schreibgeschützt</i>)
<code>UISearchController.searchResultsUpdater</code>	Das Objekt, das für die Aktualisierung des Inhalts des Suchergebniscontrollers verantwortlich ist.
<code>UISearchController.isActive</code>	Der dargestellte Status der Suchoberfläche.
<code>UISearchController.obscuresBackgroundDuringPresentation</code>	Ein boolescher Wert, der angibt, ob der zugrunde liegende Inhalt während einer Suche verdeckt wird.
<code>UISearchController.dimsBackgroundDuringPresentation</code>	Ein boolescher Wert, der angibt, ob der zugrunde liegende Inhalt während einer Suche abgeblendet ist.
<code>UISearchController.hidesNavigationBarDuringPresentation</code>	Ein boolescher Wert, der angibt, ob die Navigationsleiste bei der Suche ausgeblendet werden soll.
<code>UIViewController.definesPresentationContext</code>	Ein boolescher Wert, der angibt, ob die Ansicht dieses View-Controllers abgedeckt ist, wenn der View-Controller oder einer seiner Nachkommen einen View-

Parameter	Einzelheiten
	Controller darstellt.
<code>UINavigationController.navigationItem.titleView</code>	Eine benutzerdefinierte Ansicht, die in der Mitte der Navigationsleiste angezeigt wird, wenn der Empfänger das oberste Element ist, in dem eine Suchleiste platziert werden kann.
<code>UITableViewController.tableView.tableHeaderView</code>	Gibt eine Zubehöransicht zurück, die über der Tabelle angezeigt wird, in der eine Suchleiste platziert werden kann.

Bemerkungen

UIKit Framework-Referenz:

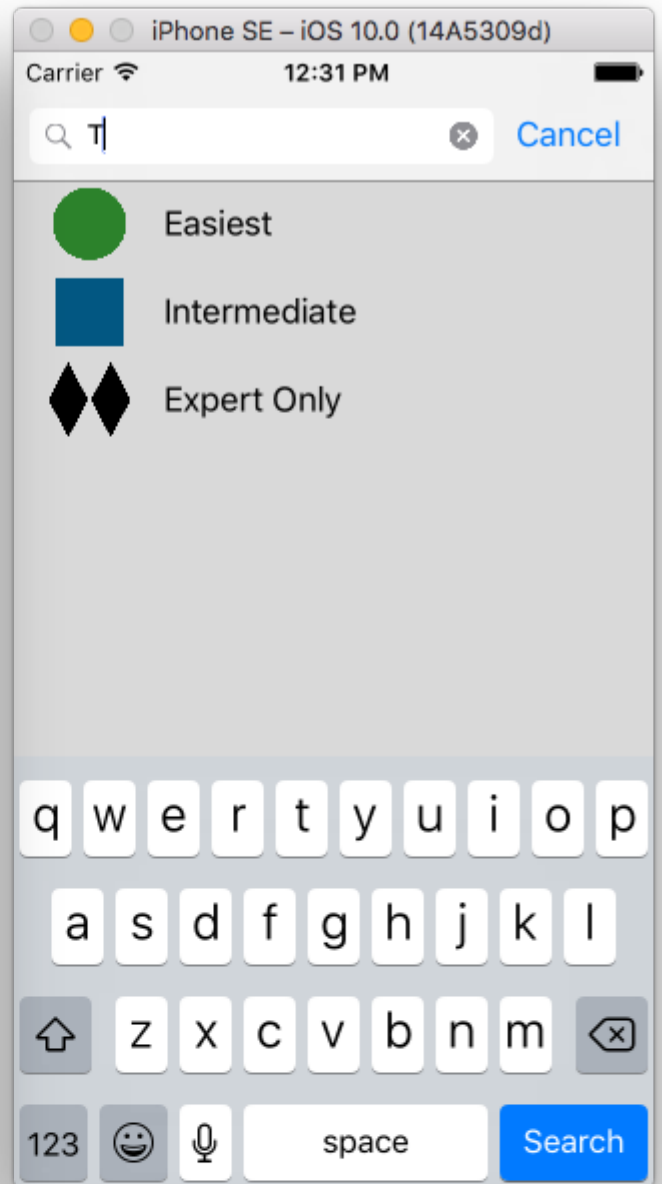
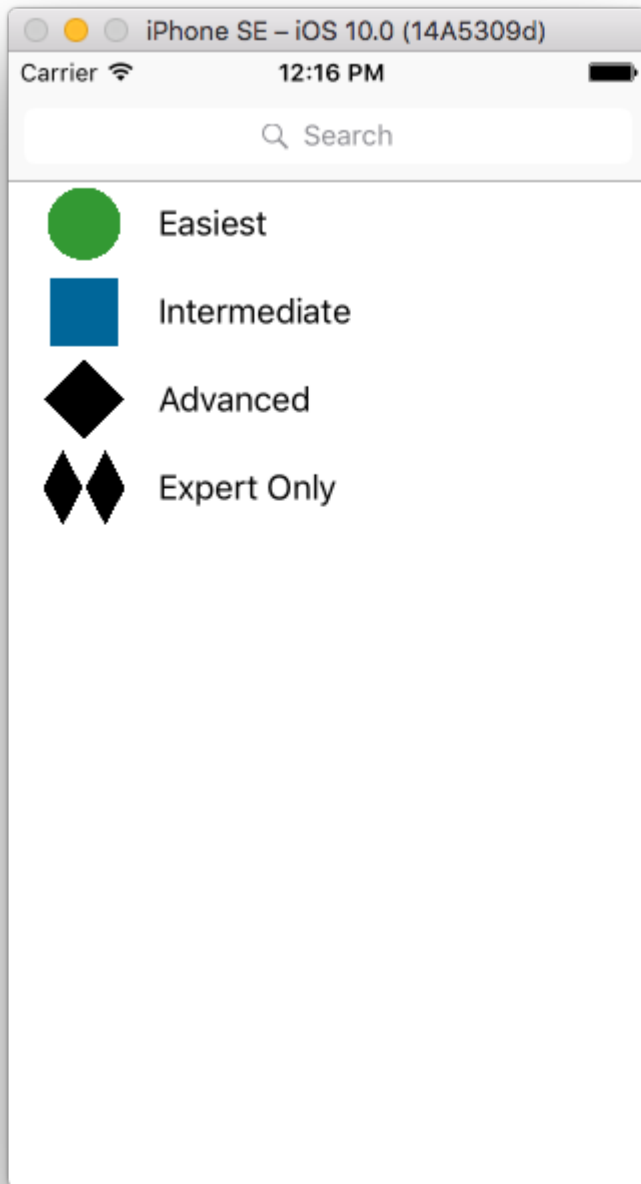
[UISearchController](#)

[UISearchErgebnisseAktualisierung](#)

Examples

Suchleiste im Titel der Navigationsleiste

In diesem Beispiel wird ein Suchcontroller verwendet, um die Daten in einem Tabellensichtcontroller zu filtern. Die Suchleiste befindet sich in der Navigationsleiste, in die die Tabellenansicht eingebettet ist.



(das die Navigationsleiste enthält) zu erhalten. Stellen Sie dann unsere benutzerdefinierte ViewController-Klasse so ein, dass sie von UITableViewController erbt und das UISearchResultsUpdating Protokoll UISearchResultsUpdating .

```
class ViewController: UITableViewController, UISearchResultsUpdating {

    let entries = [(title: "Easiest", image: "green_circle"),
                  (title: "Intermediate", image: "blue_square"),
                  (title: "Advanced", image: "black_diamond"),
                  (title: "Expert Only", image: "double_black_diamond")]

    // An empty tuple that will be updated with search results.
    var searchResults : [(title: String, image: String)] = []

    let searchController = UISearchController(searchResultsController: nil)

    override func viewDidLoad() {
        super.viewDidLoad()

        searchController.searchResultsUpdater = self
        self.definesPresentationContext = true

        // Place the search bar in the navigation item's title view.
        self.navigationItem.titleView = searchController.searchBar

        // Don't hide the navigation bar because the search bar is in it.
        searchController.hidesNavigationBarDuringPresentation = false
    }

    func filterContent(for searchText: String) {
        // Update the searchResults array with matches
        // in our entries based on the title value.
        searchResults = entries.filter({ (title: String, image: String) -> Bool in
            let match = title.range(of: searchText, options: .caseInsensitive)
            // Return the tuple if the range contains a match.
            return match != nil
        })
    }

    // MARK: - UISearchResultsUpdating method

    func updateSearchResults(for searchController: UISearchController) {
        // If the search bar contains text, filter our data with the string
        if let searchText = searchController.searchBar.text {
            filterContent(for: searchText)
            // Reload the table view with the search result data.
            tableView.reloadData()
        }
    }

    // MARK: - UITableViewController methods

    override func numberOfSections(in tableView: UITableView) -> Int { return 1 }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        // If the search bar is active, use the searchResults data.
        return searchController.isActive ? searchResults.count : entries.count
    }
}
```

```

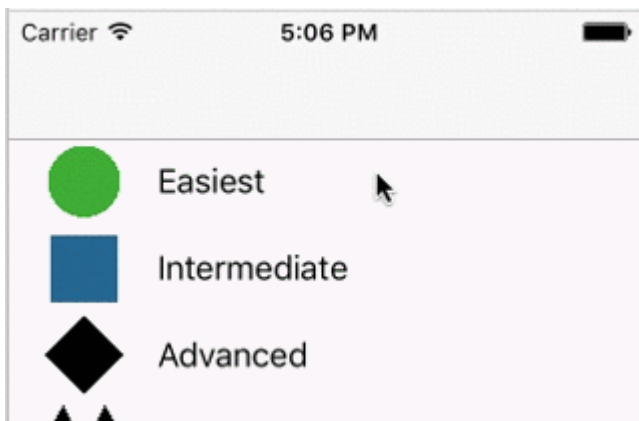
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
    // If the search bar is active, use the searchResults data.
    let entry = searchController.isActive ?
        searchResults[indexPath.row] : entries[indexPath.row]

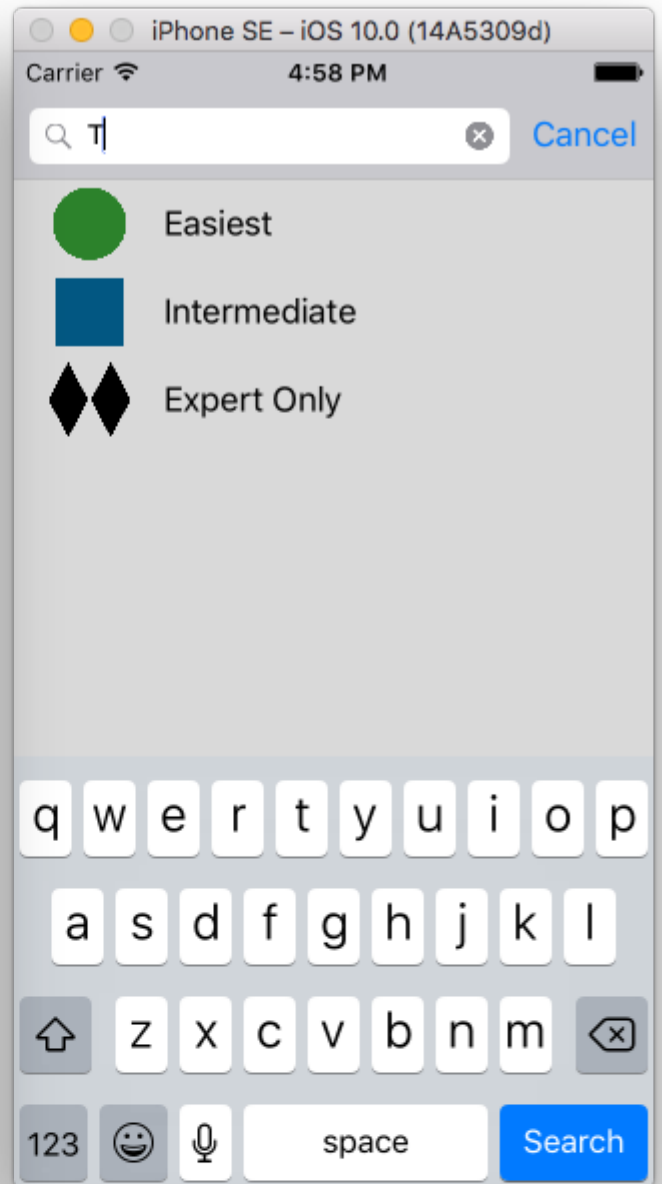
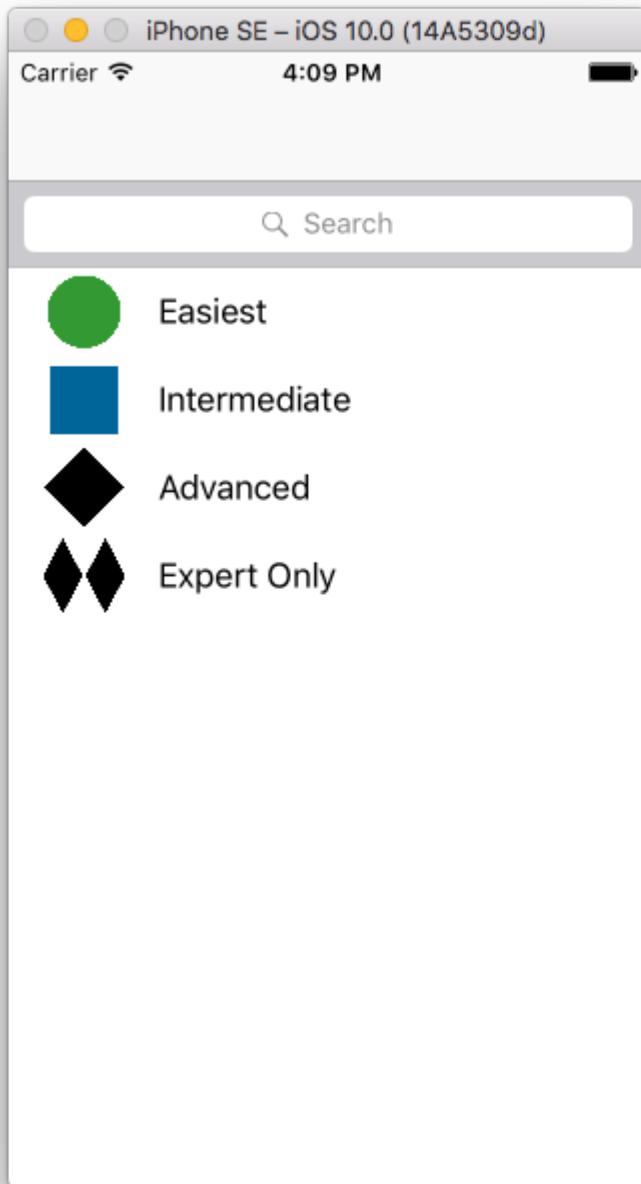
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
    cell.textLabel?.text = entry.title
    cell.imageView?.image = UIImage(named: entry.image)
    return cell
}
}

```

Suchleiste im Header der Tabellenansicht

In diesem Beispiel wird ein Suchcontroller verwendet, um die Zellen in einem Tabellensichtcontroller zu filtern. Die Suchleiste befindet sich in der Kopfansicht der Tabellenansicht. Der Inhalt der Tabellendarstellung ist um die gleiche Höhe wie die Suchleiste versetzt, sodass die Suchleiste zunächst ausgeblendet wird. Wenn Sie am oberen Rand der Tabellenansicht nach oben scrollen, wird die Suchleiste eingeblendet. Wenn dann die Suchleiste aktiv wird, blendet sie die Navigationsleiste aus.





updateSearchResultsController Methode, die aus dem UISearchResultsUpdating Protokoll stammt:

```
func updateSearchResultsController(searchController: UISearchController) {  
  
}
```

UISerachController in Objective-C

```
Delegate: UISearchBarDelegate, UISearchControllerDelegate, UISearchBarDelegate  
  
@property (strong, nonatomic) UISearchController *searchController;  
  
- (void)searchBarConfiguration  
{  
    self.searchController = [[UISearchController alloc] initWithSearchResultsController:nil];  
    self.searchController.searchBar.delegate = self;  
    self.searchController.hidesNavigationBarDuringPresentation = NO;  
  
    // Hides search bar initially. When the user pulls down on the list, the search bar is  
    revealed.  
    [self.tableView setContentOffset:CGPointMake(0,  
self.searchController.searchBar.frame.size.height)];  
  
    self.searchController.searchBar.backgroundColor = [UIColor DarkBlue];  
    self.searchController.searchBar.tintColor = [UIColor DarkBlue];  
  
    self.tableView.contentOffset = CGPointMake(0,  
CGRectGetHeight(_searchController.searchBar.frame));  
    self.tableView.tableHeaderView = _searchController.searchBar;  
    _searchController.searchBar.delegate = self;  
    _searchController.searchBar.showsCancelButton = YES;  
    self.tapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self  
action:@selector(resetSearchbarAndTableView)];  
    [self.view addGestureRecognizer:self.tapGestureRecognizer];  
}  
  
- (void)resetSearchbarAndTableView{  
    // Reload your tableview and resign keyboard.  
}  
  
- (void)searchBarCancelButtonClicked:(UISearchBar *)searchBar{  
    // Search cancelled  
}  
- (void)searchBarSearchButtonClicked:(UISearchBar *)searchBar{  
    // Implement filtration of your data as per your need using NSPredicate or else.  
    // then reload your data control like Tableview.  
}
```

UISearchController online lesen: <https://riptutorial.com/de/ios/topic/2813/uisearchcontroller>

Kapitel 183: UISegmentedControl

Einführung

Ein UISegmentedControl-Objekt ist ein horizontales Steuerelement, das aus mehreren Segmenten besteht, wobei jedes Segment als diskrete Schaltfläche fungiert. Eine segmentierte Steuerung bietet eine kompakte Möglichkeit, eine Reihe von Steuerungen zu gruppieren.

Examples

UISegmentedControl über Code erstellen

1. Erstellen Sie eine neue Instanz von UISegmentedControl, die mit 3 Elementen (Segmenten) gefüllt ist:

```
let mySegmentedControl = UISegmentedControl (items: ["One", "Two", "Three"])
```

2. Setup-Frame;

```
mySegmentedControl.frame = CGRect(x: 0.0, y: 0.0, width: 300, height: 50)
```

3. Standardauswahl treffen (nicht, dass Segmente mit 0 indiziert werden):

```
mySegmentedControl.selectedSegmentIndex = 0
```

4. Ziel konfigurieren:

```
mySegmentedControl.addTarget(self, action: #selector(segmentedValueChanged(_:)), for: .valueChanged)
```

- 5 Handle-Wert geändert:

```
func segmentedValueChanged(_ sender:UISegmentedControl!) {  
    print("Selected Segment Index is : \(sender.selectedSegmentIndex)")  
}
```

6. Fügen Sie UISegmentedControl zur Ansichtshierarchie hinzu

```
yourView.addSubview(mySegmentedControl)
```

UISegmentedControl online lesen: <https://riptutorial.com/de/ios/topic/9963/uisegmentedcontrol>

Kapitel 184: UISlider

Examples

UISlider

Ziel c

Deklarieren Sie eine Slider-Eigenschaft in `ViewController.h` oder in der Schnittstelle von `ViewController.m`

```
@property (strong, nonatomic)UISlider *slider;

//Define frame of slider and add to view
CGRect frame = CGRectMake(0.0, 100.0, 320.0, 10.0);
UISlider *slider = [[UISlider alloc] initWithFrame:frame];
[slider addTarget:self action:@selector(sliderAction:)
forControlEvents:UIControlEventValueChanged];
[self.slider setBackgroundColor:[UIColor clearColor]];
self.slider.minimumValue = 0.0;
self.slider.maximumValue = 50.0;
//sending a NO/False would update the value of slider only when the user is no longer touching
the screen. Hence sending only the final value
self.slider.continuous = YES;
self.slider.value = 25.0;
[self.view addSubview slider];
```

Behandeln Sie den Schiebereglerwechsel

```
- (IBAction)sliderAction:(id)sender {
    NSLog(@"Slider Value %f", sender.value);
}
```

SWIFT-Beispiel

```
let frame = CGRect(x: 0, y: 100, width: 320, height: 10)
let slider = UISlider(frame: frame)
slider.addTarget(self, action: #selector(sliderAction), for: .valueChanged)
slider.backgroundColor = .clear
slider.minimumValue = 0.0
slider.maximumValue = 50.0
//sending a NO/False would update the value of slider only when the user is no longer
touching the screen. Hence sending only the final value
slider.isContinuous = true
slider.value = 25.0
view.addSubview(slider)
```

Umgang mit dem Slider-Change-Ereignis

```
func sliderAction(sender:UISlider!)
{
```

```
print ("value--\ (sender.value) ")  
}
```

Hinzufügen eines benutzerdefinierten Daumens

Um ein benutzerdefiniertes Bild für den Daumen des Schiebereglers hinzuzufügen, rufen Sie einfach die [setThumbImage](#)- Methode mit Ihrem benutzerdefinierten Bild auf:

Schnell 3,1:

```
let slider = UISlider()  
let thumbImage = UIImage  
slider.setThumbImage(thumbImage, for: .normal)
```

UISlider online lesen: <https://riptutorial.com/de/ios/topic/7402/uislider>

Kapitel 185: UISplitViewController

Bemerkungen

`UISplitViewController` ist eine Containerklasse wie `UITabViewController`, `UINavigationController`. Es teilt die Hauptansicht in zwei View Controller (`MasterViewController`) und `DetailViewController` (`SecondaryViewController`) auf. Wir können ein Array mit zwei View-Controllern senden, das Apple als Rootview-Controller für Ihre Anwendung an `UISplitViewController` empfiehlt. Für die Interaktion zwischen den Viewcontrollern verwende ich `NSNotificationCenter`.

Examples

Master- und Detailansicht-Interaktion mit Delegaten in Ziel C

`UISplitViewController` muss der `rootViewController` Ihrer Anwendung sein.

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
    self.window.backgroundColor = [UIColor blackColor];
    [self.window makeKeyAndVisible];
    self.window.clipsToBounds = YES;
    SplitViewController *spView = [[SplitViewController alloc]init];
    self.window.rootViewController = spView;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Erstellen Sie einfach ein Objekt für den `UISplitViewController` und legen Sie diesen Viewcontroller als Rootviewcontroller für Ihre Anwendung fest.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
    DetailViewController *detailVC;
    MasterViewController *masterVC;
    NSMutableArray *array;
}
@end
```

`MasterViewController` befindet sich immer auf der linken Seite des Geräts. Sie können die Breite in `UISplitViewController` Delegierungsmethoden von `DetailViewController` `UISplitViewController`

`DetailViewController` befindet sich auf der rechten Seite der Anwendung

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    masterVC = [[MasterViewController alloc] init];
    detailVC = [[DetailViewController alloc] init];
    [masterVC setDetailDelegate:(id)detailVC];
    NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
    self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
    self.viewControllers = vcArray;
    self.delegate = (id)self;
    self.presentsWithGesture = YES;
}
}
```

Erstellte Master- und Detail-ViewControllers werden zu einem Array hinzugefügt, das in `self.viewControllers` auf `UISplitViewController`. `self.preferredDisplayMode` ist der Modus, der für die Anzeige der Master- und `DetailViewController` [Apple-Dokumentation für DisplayMode eingestellt ist](#). `self.presentsWithGesture` die `self.presentsWithGesture` zur Anzeige des `MasterViewController`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void)sendSelectedNavController:(UIViewController *)viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Erstellen Sie einen `DetailViewDelegate` Delegaten mit `sendSelectedNavController:(UIViewController *)viewController` Methode zum Senden des `UIViewController` an den `DetailViewController`. In `MasterViewController` die `mainTableView` die `MasterViewController` der linken Seite. Der `viewControllerArray` enthält alle `UITableViewController`, die in `DetailViewController` angezeigt werden `DetailViewController`

MasterViewController.m

```
#import "MasterViewController.h"
```

```

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
[super viewDidLoad];

UIViewController *dashBoardVC = [[UIViewController alloc]init];
[dashBoardVC.view setBackgroundColor:[UIColor redColor]];
UIViewController *inventVC = [[UIViewController alloc]init];
[inventVC.view setBackgroundColor:[UIColor whiteColor]];
UIViewController *alarmVC = [[UIViewController alloc]init];
[alarmVC.view setBackgroundColor: [UIColor purpleColor]];
UIViewController *scanDeviceVC = [[UIViewController alloc]init];
[scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
UIViewController *serverDetailVC = [[UIViewController alloc]init];
[serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
viewControllerArray = [[NSMutableArray
alloc]initWithObjects:dashBoardVC,inventVC,alarmVC,scanDeviceVC,serverDetailVC,nil];
mainTableView = [[UITableView alloc]initWithFrame:CGRectMake(0, 50,self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
[mainTableView setDelegate:(id)self];
[mainTableView setDataSource:(id)self];
[mainTableView setSeparatorStyle:UITableViewCellStyleNone];
[mainTableView setScrollsToTop:NO];
[self.view addSubview:mainTableView];
}

-(CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
return 100;
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection: (NSInteger)section
{
return [viewControllerArray count];
}

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return 1; //count of section
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
NSString *cellId = [NSString
stringWithFormat:@"Cell%li%ld", (long)indexPath.section, (long)indexPath.row];
UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:cellId];

if (cell == nil)
{
cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellId];
}
[cell.contentView setBackgroundColor:[UIColor redColor]];
cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld", (long)indexPath.row];
return cell;
}

```



```

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end

```

Erstellen Sie einige `UIViewController`s und `UIViewController`s Sie sie einem Array hinzu. Die Tabellenansicht wird dann auf initialisierten `didSelectRowAtIndexPath` Methode I einen senden `UIViewController` zum `DetailViewController` Verwendung `detailDelegate` mit dem entsprechenden `UIViewController` in Array als Parameter

DetailViewController.h

```

#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UIViewController *tempNav;
}

@end

```

DetailViewController.m

```

#import "DetailViewController.h"

@implementation DetailViewController
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
-(void)sendSelectedNavController:(UIViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}

@end

```

`sendSelectedNavController` wird hier deklariert, indem alle Ansichten im `DetailViewController` und der übergebene `UIViewController` aus dem `MasterViewController`

Einige Screenshots der Anwendung hinzufügen



`preferredDisplayMode` als automatisch festgelegt haben. Beim Wischen des Bildschirms erhalten Sie den `MasterViewController` wie in der folgenden Abbildung angehängt. Im Landscape-Modus erhalten Sie jedoch den `MasterViewController` und den `DetailViewController`

Carrier 

8:26 PM

My VC at index 0

My VC at index 1

My VC at index 2

My VC at index 3

My VC at index 4

My VC at index 0

My VC at index 1

My VC at index 2

My VC at index 3

My VC at index 4

Kapitel 186: UISplitViewController

Bemerkungen

In iOS 8 und höher können Sie die `UISplitViewController` Klasse auf allen iOS-Geräten verwenden. In früheren Versionen von iOS ist die Klasse nur auf dem iPad verfügbar.

`UISplitViewController` ist eine Containerklasse wie `UITabViewController`, `UINavigationController`.

Sie `UISplitViewController` die Hauptansicht in zwei `UIViewController`s `masterViewController` (`PrimaryViewController`) und `detailViewController` (`SecondaryViewController`). Wir können einen

`NSArray` mit zwei `UIViewController`s und Apple empfiehlt `UISplitViewController` als

RootviewController für Ihre Anwendung. Für die Interaktion zwischen den `UIViewController`s ich

`NSNotificationCenter`.

Examples

Interaktion zwischen Master- und Detailansicht mithilfe von Delegierten in Ziel C

`UISplitViewController` muss den Root-View-Controller des App-Fensters verwenden

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
    self.window.backgroundColor = [UIColor blackColor];
    [self.window makeKeyAndVisible];
    self.window.clipsToBounds = YES;
    SplitViewController *spView = [[SplitViewController alloc]init];
    self.window.rootViewController = spView;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Erstellen Sie `UISplitViewController` ein Objekt für Ihren `UISplitViewController` und legen Sie es als `rootViewController` für Ihre Anwendung fest.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
    DetailViewController *detailVC;
    MasterViewController *masterVC;
    NSMutableArray *array;
}
```

```
@end
```

`MasterViewController` ist ein `UIViewController`, der auf der linken Seite des Geräts eingestellt wird. Sie können die Breite in `UISplitViewController` mithilfe von `maximumPrimaryColumnWidth` und `DetailViewController` auf der rechten Seite `DetailViewController`

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    masterVC = [[MasterViewController alloc] init];
    detailVC = [[DetailViewController alloc] init];
    [masterVC setDetailDelegate:(id)detailVC];
    NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
    self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
    self.viewControllers = vcArray;
    self.delegate = (id)self;
    self.presentsWithGesture = YES;
}
}
```

Die Master - und Detail - `UIViewController` 's werden zu einem `NSArray` hinzugefügt, der auf `self.viewControllers`. `self.preferredDisplayMode` ist der Modus zur Anzeige von `MasterViewController` und `DetailViewController`. `self.presentsWithGesture` die `self.presentsWithGesture` zur Anzeige von `MasterViewController`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void)sendSelectedNavController:(UIViewController *)viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Erstellen Sie einen `DetailViewDelegate` Delegaten mit der `sendSelectedNavController` Methode, um die `UITableViewController`s an den `DetailViewController` zu `DetailViewController`. In `MasterViewController` eine `UITableView` erstellt. Der `viewControllerArray` enthält alle `UITableViewController`s, die in `DetailViewController` angezeigt werden `DetailViewController`

MasterViewController.m

```
#import "MasterViewController.h"

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
    [super viewDidLoad];

    UIViewController *dashBoardVC = [[UIViewController alloc]init];
    [dashBoardVC.view setBackgroundColor:[UIColor redColor]];
    UIViewController *inventVC = [[UIViewController alloc]init];
    [inventVC.view setBackgroundColor:[UIColor whiteColor]];
    UIViewController *alarmVC = [[UIViewController alloc]init];
    [alarmVC.view setBackgroundColor: [UIColor purpleColor]];
    UIViewController *scanDeviceVC = [[UIViewController alloc]init];
    [scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
    UIViewController *serverDetailVC = [[UIViewController alloc]init];
    [serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
    viewControllerArray = [[NSMutableArray
alloc] initWithObjects:dashBoardVC, inventVC, alarmVC, scanDeviceVC, serverDetailVC, nil];
    mainTableView = [[UITableView alloc] initWithFrame:CGRectMake(0, 50, self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
    [mainTableView setDelegate:(id)self];
    [mainTableView setDataSource:(id)self];
    [mainTableView setSeparatorStyle:UITableViewCellStyleNone];
    [mainTableView setScrollsToTop:NO];
    [self.view addSubview:mainTableView];
}

-(CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 100;
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection: (NSInteger)section
{
    return [viewControllerArray count];
}

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1; //count of section
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    NSString *cellId = [NSString
stringWithFormat:@"Cell%li%ld", (long) indexPath.section, (long) indexPath.row];
    UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:cellId];

    if (cell == nil)
    {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellId];
    }
    [cell.contentView setBackgroundColor:[UIColor redColor]];
    cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld", (long) indexPath.row];
}
```

```

return cell;
}

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end

```

Erstellt einige `UIViewController` und fügt sie einem `NSMutableArray`. Die `UITableView` initialisiert dann auf `didselectrowatindexPath` Methode I einen senden `UIViewController` zum `DetailViewController` Verwendung `detailDelegate` Delegierten mit dem entsprechenden `UIViewController` im `NSMutableArray` als Parameter

DetailViewController.h

```

#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UIViewController *tempNav;
}
@end

```

DetailViewController.m

```

#import "DetailViewController.h"

@implementation DetailViewController
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
-(void)sendSelectedNavController:(UIViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}
@end

```

Der `sendSelectedNavController` wird hier deklariert, indem alle `UIView` im `DetailViewController` und der übergebene `UIViewController` aus dem `MasterViewController`.

UISplitViewController online lesen: <https://riptutorial.com/de/ios/topic/4844/uisplitviewController>

Kapitel 187: UIStackView

Examples

Erstellen Sie eine horizontale Stapelansicht programmgesteuert

Swift 3

```
let stackView = UIStackView()
stackView.axis = .horizontal
stackView.alignment = .fill // .leading .firstBaseline .center .trailing .lastBaseline
stackView.distribution = .fill // .fillEqually .fillProportionally .equalSpacing
    .equalCentering

let label = UILabel()
label.text = "Text"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

Schnell

```
let stackView = UIStackView()
stackView.axis = .Horizontal
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
    .EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

Ziel c

```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisHorizontal;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For horizontal stack view, you might want to add a width constraint to your label or
whatever view you are adding.
```

Erstellen Sie eine vertikale Stapelansicht programmgesteuert

Schnell

```
let stackView = UIStackView()
stackView.axis = .Vertical
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
.EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for vertical stack view, you might want to add height constraint to label or whatever view
you're adding.
```

Ziel c

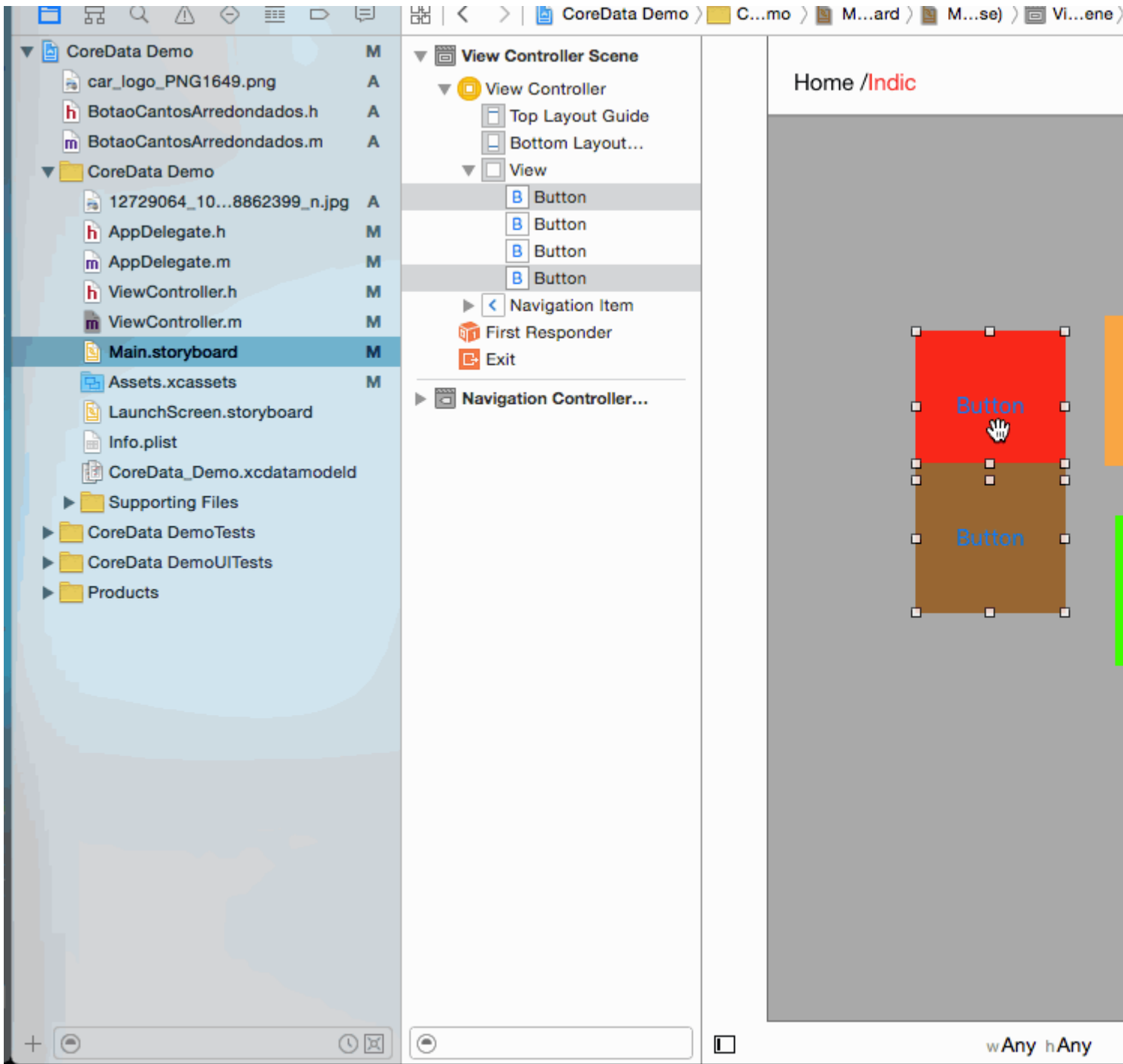
```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisVertical;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For vertical stack view, you might want to add a height constraint to your label or whatever
view you are adding.
```

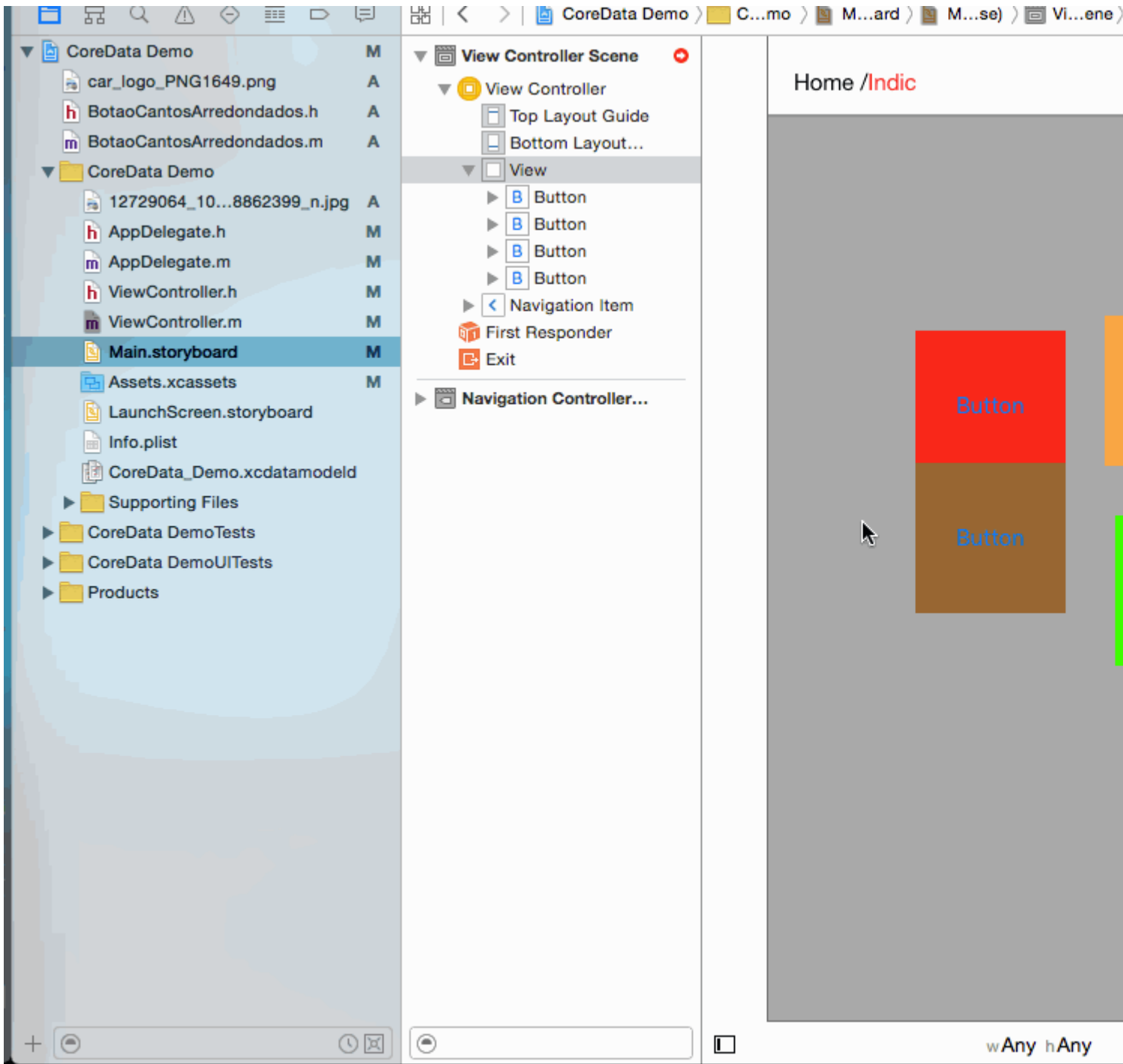
Zentriertasten mit UIStackview

Schritt 1: - Nehmen Sie die 4 Taste in Ihrem Storyboard. Button1, Button2, Button 3, Button4

Schritt 2: - Allen Schaltflächen feste Höhe und Breite zuweisen.

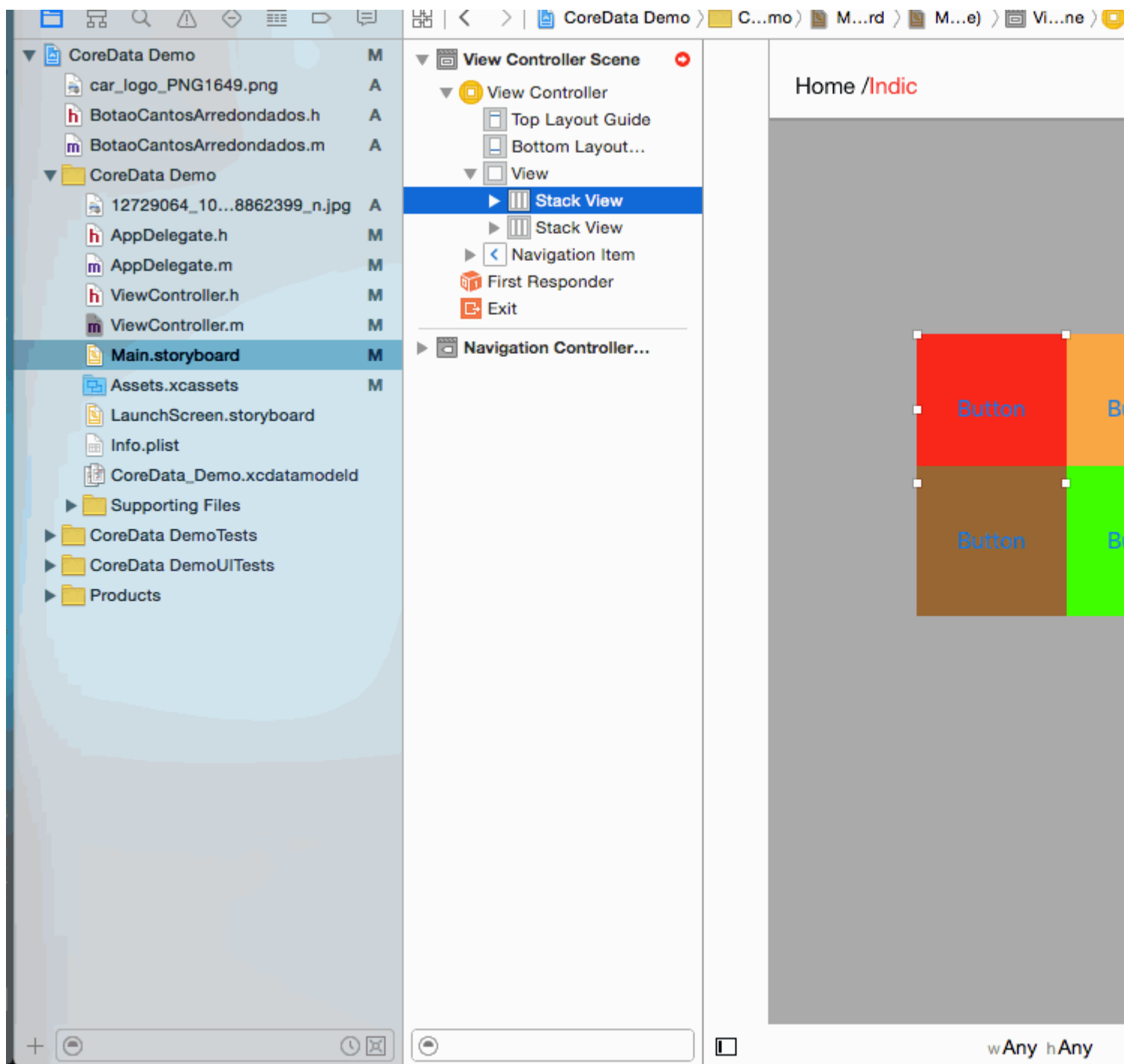
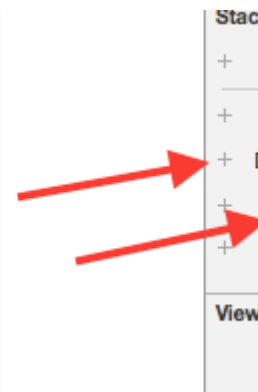
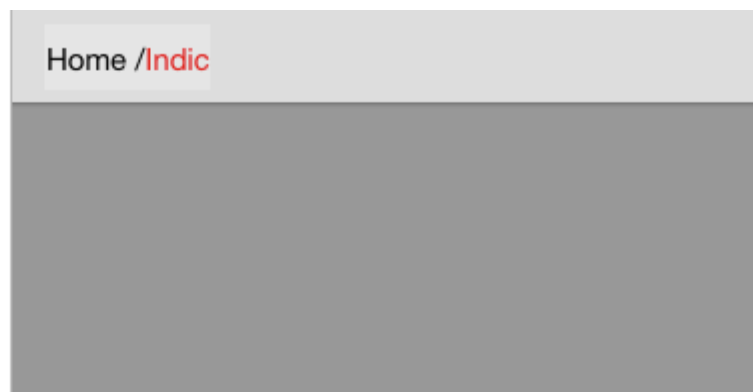
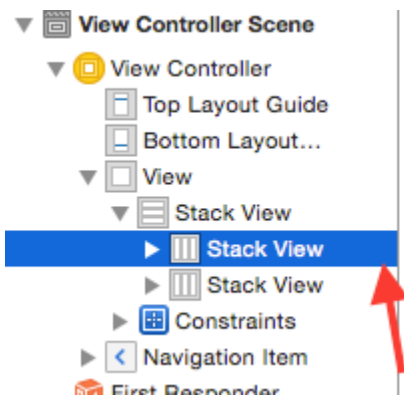


Schritt 3: - Alle 2 - 2 Tastenpaare in 2-Stack-Ansicht.

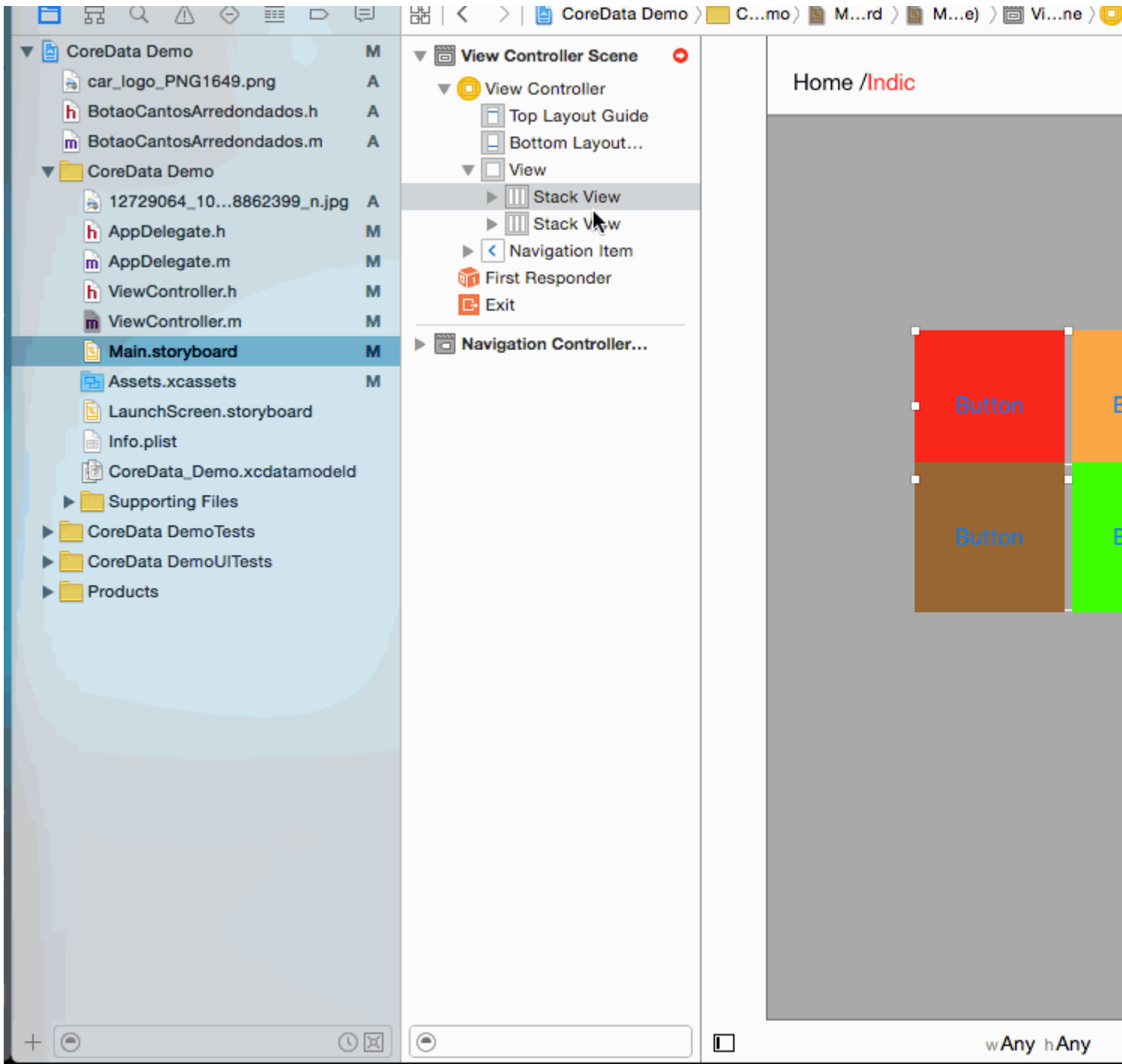


Schritt 4: - Legen Sie die UIStackview-Eigenschaft für beide fest.

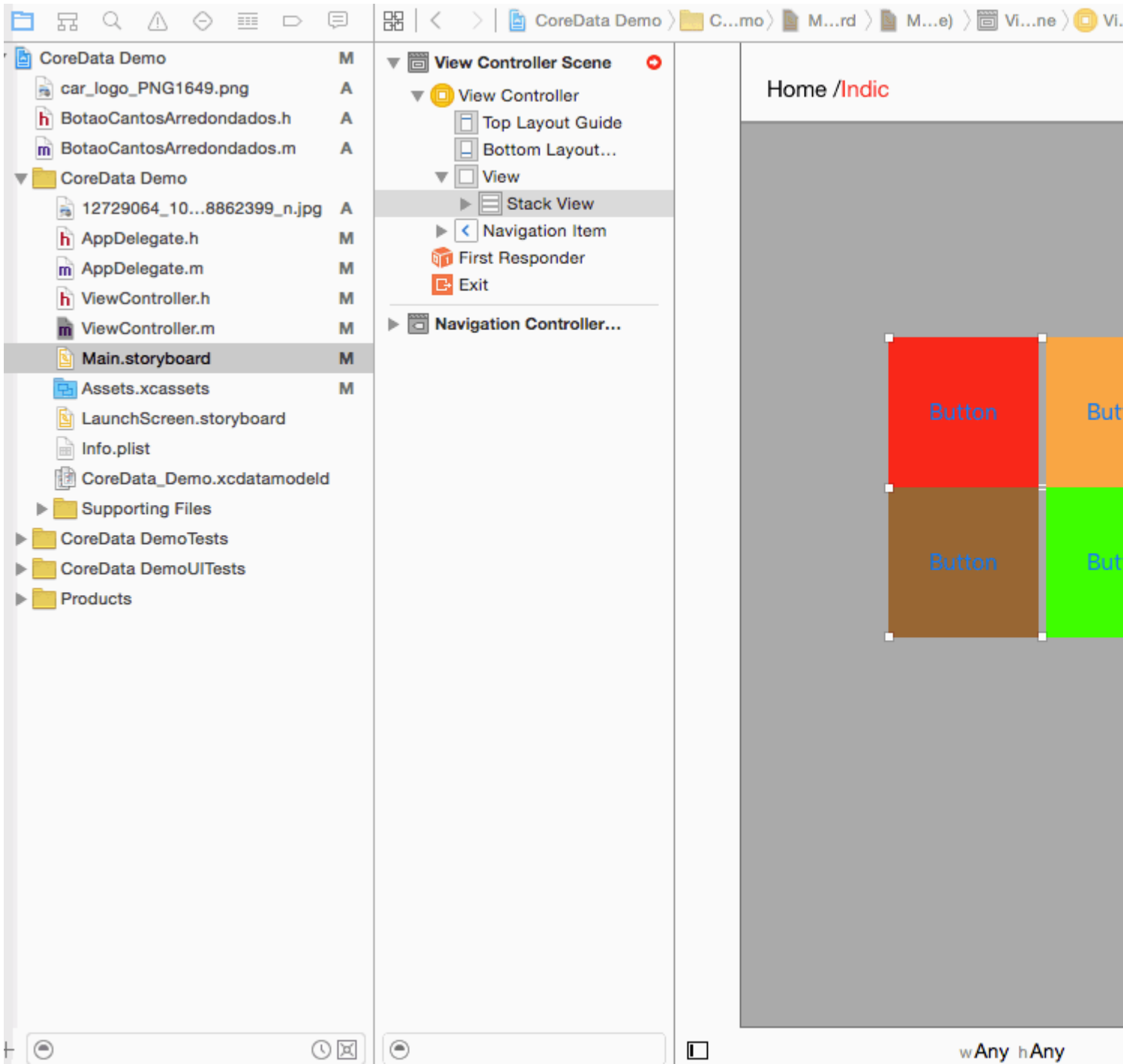
```
Distribution -> Fill Equally  
Spacing -> 5 (as per your requirement)
```



Schritt 5: - Fügen Sie beide Stackview in einem Stackview hinzu

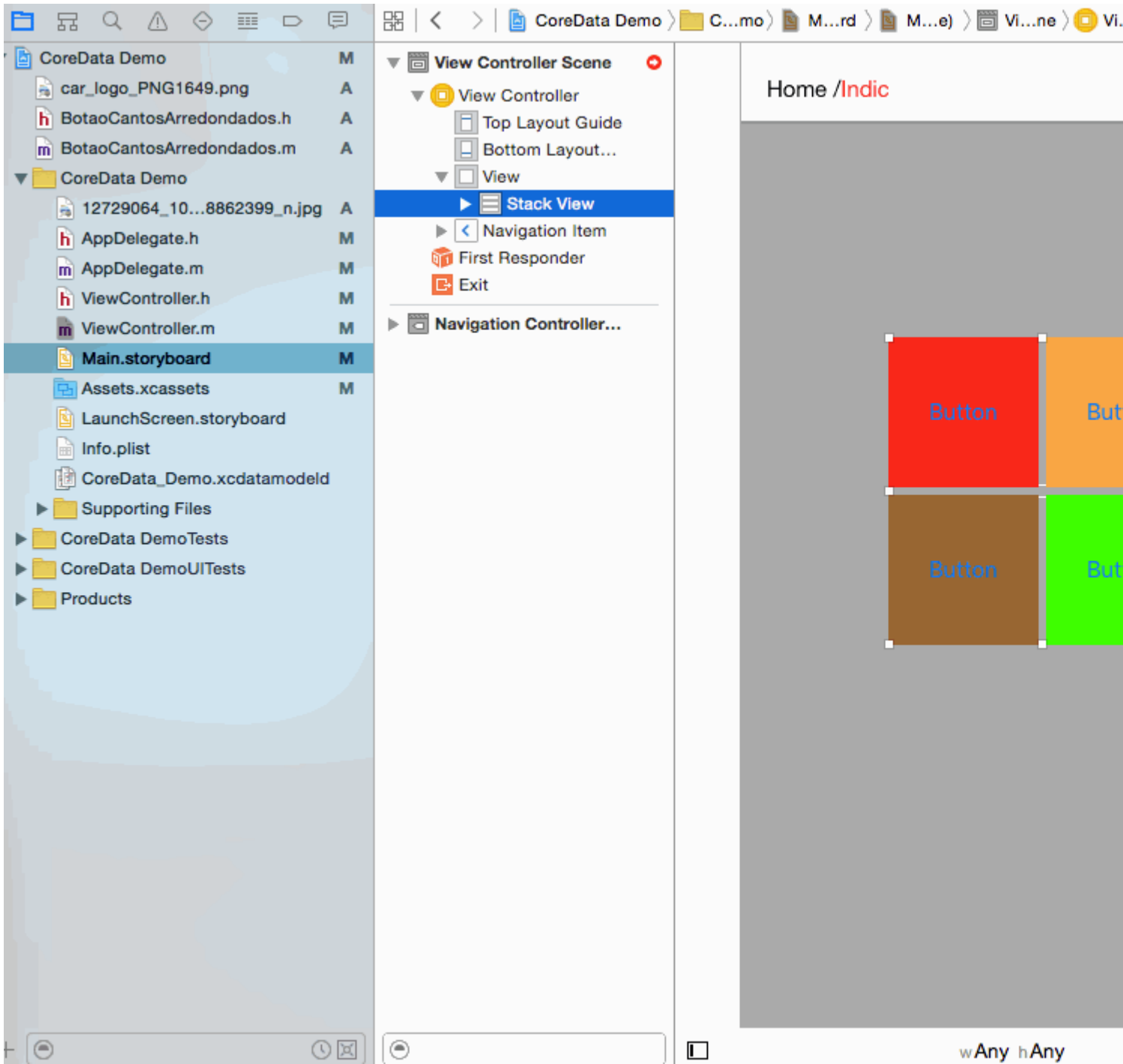


Schritt 6: - Distribution = Fill equally Spacing =5 einstellen Distribution = Fill equally Spacing =5 Gleiche Fläche Distribution = Fill equally Spacing =5 in der Hauptstapelansicht

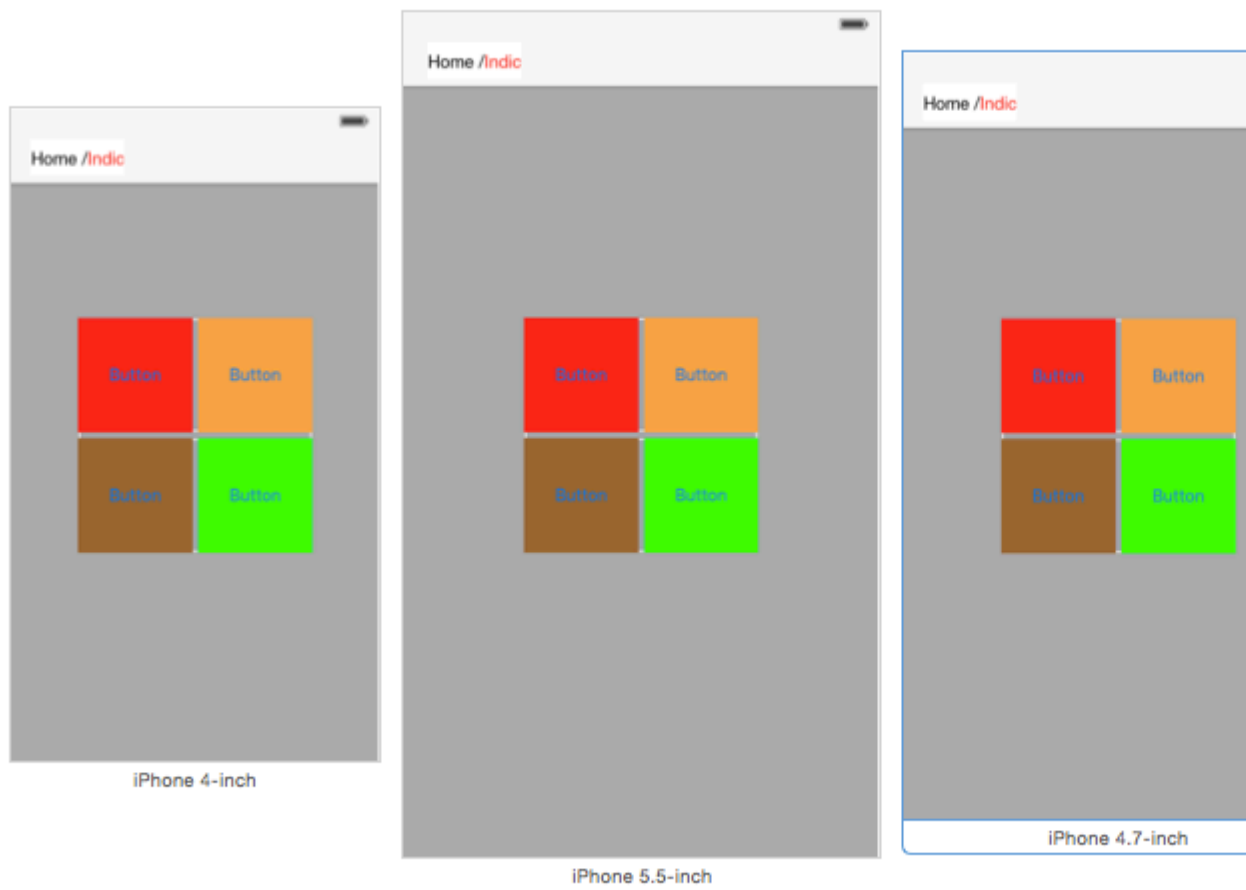


Schritt 7: - Setzen Sie nun Constrain auf main stackview

```
center Horizontally in container  
  
center vertically in container  
  
and select Update Frame.
```



Schritt 8: - Es ist Zeit für die Ausgabe für alle Geräte.



UIStackView online lesen: <https://riptutorial.com/de/ios/topic/1390/uistackview>

Kapitel 188: UIStackView dynamisch aktualisieren

Examples

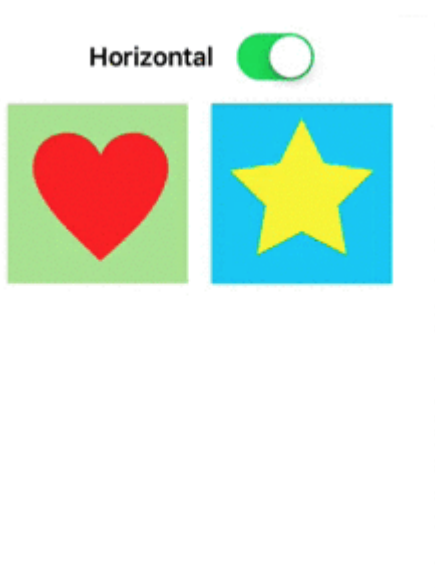
Verbinden Sie den UISwitch mit einer Aktion, mit der Sie zwischen horizontalem oder vertikalem Layout der Bildansichten wechseln können

```
@IBAction func axisChange(sender: UISwitch) {
    UIView.animateWithDuration(1.0) {
        self.updateConstraintsForAxis()
    }
}
```

Die updateConstraintForAxis-Funktion legt nur die Achse der Stapelansicht fest, die die beiden Bildansichten enthält:

```
private func updateConstraintsForAxis() {
    if (axisSwitch.on) {
        stackView.axis = .Horizontal
    } else {
        stackView.axis = .Vertical
    }
}
```

Das animierte Gif unten gibt Ihnen eine Vorstellung davon, wie dies aussieht:



[UIStackView dynamisch aktualisieren online lesen:](#)

<https://riptutorial.com/de/ios/topic/5884/uistackview-dynamisch-aktualisieren>

Kapitel 189: UIStoryboard

Einführung

Ein UIStoryboard-Objekt kapselt den in einer Interface Builder-Storyboard-Ressourcendatei gespeicherten View Controller-Graphen. Dieses Ansichtssteuerungsdiagramm stellt die Ansichtssteuerungen für die gesamte Benutzeroberfläche Ihrer Anwendung oder einen Teil davon dar.

Examples

Eine programmgesteuerte Instanz von UIStoryboard abrufen

SCHNELL:

Das programmatische **Abrufen** einer Instanz von **UINavigationController** kann wie folgt durchgeführt werden:

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

woher:

- **name** => Name des Storyboards ohne Erweiterung
- **Bundle** => das Bundle, das die Storyboard-Datei und die zugehörigen Ressourcen enthält. Wenn Sie nil angeben, sucht diese Methode im Hauptpaket der aktuellen Anwendung.

Sie können beispielsweise die oben erstellte Instanz verwenden, um auf einen bestimmten **UIViewController zuzugreifen, der** in diesem Storyboard instanziiert wird:

```
let viewController = storyboard.instantiateViewController(withIdentifier: "yourIdentifier")
```

ZIEL C:

Das **Abrufen** einer **UINavigationController**- Instanz in Objective-C kann wie folgt durchgeführt werden:

```
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"MainStoryboard" bundle:nil];
```

Beispiel für den Zugriff auf **UIViewController, der** innerhalb dieses Storyboards instanziiert wurde:

```
MyViewController *myViewController = [storyboard
```

```
instantiateViewControllerWithIdentifier:@"MyViewControllerIdentifier"];
```

Öffne ein anderes Storyboard

```
let storyboard = UIStoryboard(name: "StoryboardName", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "ViewControllerID") as
YourViewController
self.present(vc, animated: true, completion: nil)
```

UIStoryboard online lesen: <https://riptutorial.com/de/ios/topic/8795/uistoryboard>

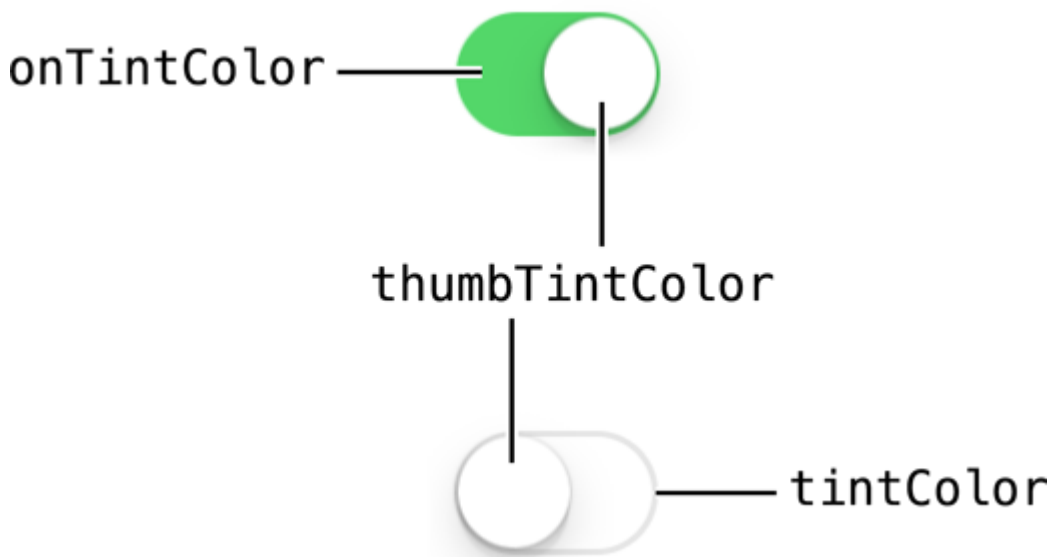
Kapitel 190: UISwitch

Syntax

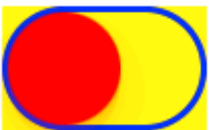
- (instancetype) initWithFrame: (CGRect) -Frame;
- (void) setOn: (BOOL) bei animiert: (BOOL) animiert;
- (nullfähiger Instanztyp) initWithCoder: (NSCoder *) aDecoder;

Bemerkungen

1. UISwitch-Referenz: [Apple-Dokumentation](#)



2. Eine weitere Referenz von: [Enoch Huang](#)



Examples

Ein / Aus einstellen

Ziel c

```
[mySwitch setOn:YES];  
//or  
[mySwitch setOn:YES animated:YES];
```


Schnell

```
mySwitch.setOn(false)
//or
mySwitch.setOn(false, animated: false)
```

Hintergrundfarbe einstellen

Ziel c

```
mySwitch.backgroundColor = [UIColor yellowColor];
[mySwitch setBackgroundColor: [UIColor yellowColor]];
mySwitch.backgroundColor = [UIColor colorWithRed:255/255.0 green:0/255.0 blue:0/255.0
alpha:1.0];
mySwitch.backgroundColor= [UIColor colorWithWhite: 0.5 alpha: 1.0];
mySwitch.backgroundColor=[UIColor colorWithHue: 0.4 saturation: 0.3 brightness:0.7 alpha:
1.0];
```

Schnell

```
mySwitch.backgroundColor = UIColor.yellow
mySwitch.backgroundColor = UIColor(red: 255.0/255, green: 0.0/255, blue: 0.0/255, alpha: 1.0)
mySwitch.backgroundColor = UIColor(white: 0.5, alpha: 1.0)
mySwitch.backgroundColor = UIColor(hue: 0.4,saturation: 0.3,brightness: 0.7,alpha: 1.0)
```

Tönungsfarbe einstellen

Ziel c

```
//for off-state
mySwitch.tintColor = [UIColor blueColor];
[mySwitch setTintColor: [UIColor blueColor]];

//for on-state
mySwitch.onTintColor = [UIColor cyanColor];
[mySwitch setOnTintColor: [UIColor cyanColor]];
```

Schnell

```
//for off-state
mySwitch.tintColor = UIColor.blueColor()

//for on-state
mySwitch.onTintColor = UIColor.cyanColor()
```

Bild für Ein / Aus einstellen

Ziel c

```
//set off-image
mySwitch.offImage = [UIImage imageNamed:@"off_image"];
```

```
[mySwitch setOffImage:[UIImage imageNamed:@"off_image"]];  
  
//set on-image  
mySwitch.onImage = [UIImage imageNamed:@"on_image"];  
[mySwitch setOnImage:[UIImage imageNamed:@"on_image"]];
```

Schnell

```
//set off-image  
mySwitch.offImage = UIImage(named: "off_image")  
  
//set on-image  
mySwitch.onImage = UIImage(named: "on_image")
```

UISwitch online lesen: <https://riptutorial.com/de/ios/topic/2182/uiswitch>

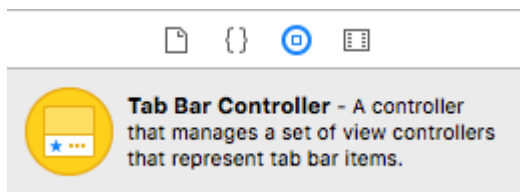
Kapitel 191: UITabBarController

Examples

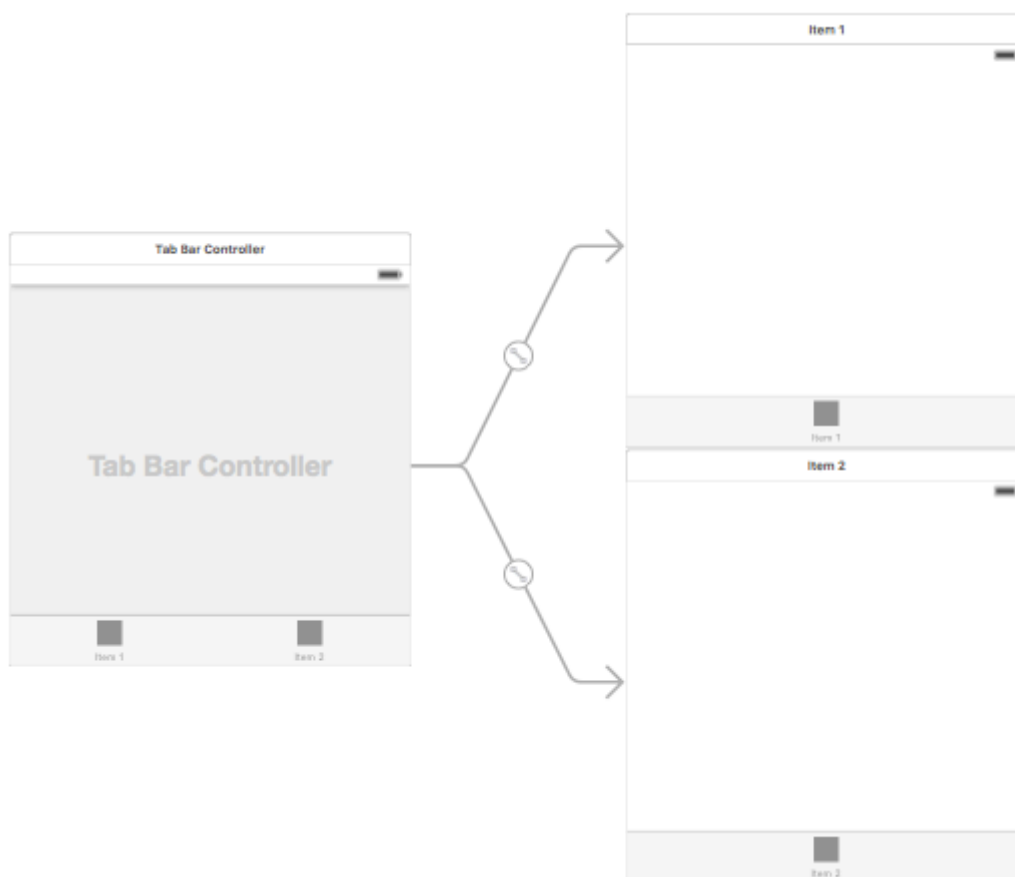
Erstellen Sie eine Instanz

Eine 'Tab-Leiste' ist in den meisten iOS-Apps zu finden und wird verwendet, um verschiedene Ansichten auf jeder Registerkarte anzuzeigen.

Ziehen Sie zum Erstellen eines Registerkarten-Controllers mit dem Schnittstellen-Generator einen Registerkarten-Controller aus der Objektbibliothek in die Leinwand.



Standardmäßig verfügt ein Registerkarten-Controller über zwei Ansichten. Wenn Sie weitere Ansichten hinzufügen möchten, ziehen Sie das Steuerelement aus der Registerkartenleiste in die neue Ansicht und wählen Sie im Dropdown-Menü die Option "Ansichtssteuerungen".



Ändern des Titeltitels und Symbols der Tab-Leiste

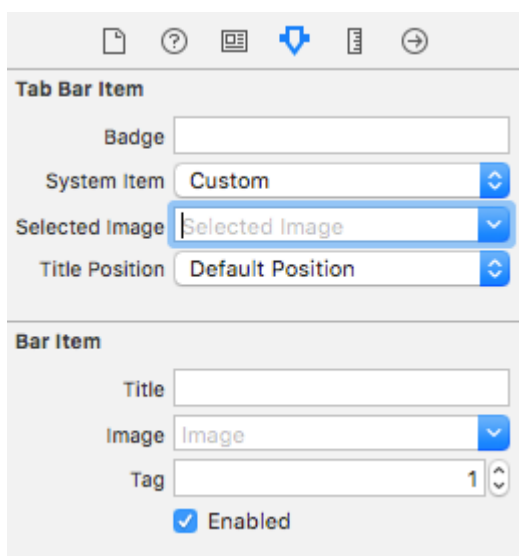
Verwenden des Storyboards:

Wählen Sie im entsprechenden View-Controller das Element der Registerkartenleiste aus, und wechseln Sie zum Attribut-Inspector

Wenn Sie ein integriertes Symbol und einen integrierten Titel haben möchten, setzen Sie das 'System Item' auf den entsprechenden Wert.

Fügen Sie für ein benutzerdefiniertes Symbol die erforderlichen Bilder zum Ordner "Assets" hinzu, und setzen Sie das "Systemelement" von früher auf "Benutzerdefiniert".

Legen Sie nun fest, dass das Symbol angezeigt wird, wenn die Registerkarte aus dem Dropdown-Menü "Ausgewähltes Bild" und das Standard-Registersymbol aus dem Dropdown-Menü "Bild" ausgewählt wird. Fügen Sie den entsprechenden Titel in das Feld "Titel" ein.



Programmatisch:

`viewDidLoad()` in der `viewDidLoad()` Methode des View-Controllers den folgenden Code hinzu:

Ziel c:

```
self.title = @"item";

self.tabBarItem.image = [UIImage imageNamed:@"item"];
self.tabBarItem.selectedImage = [UIImage imageNamed:@"item_selected"];
```

Schnell:

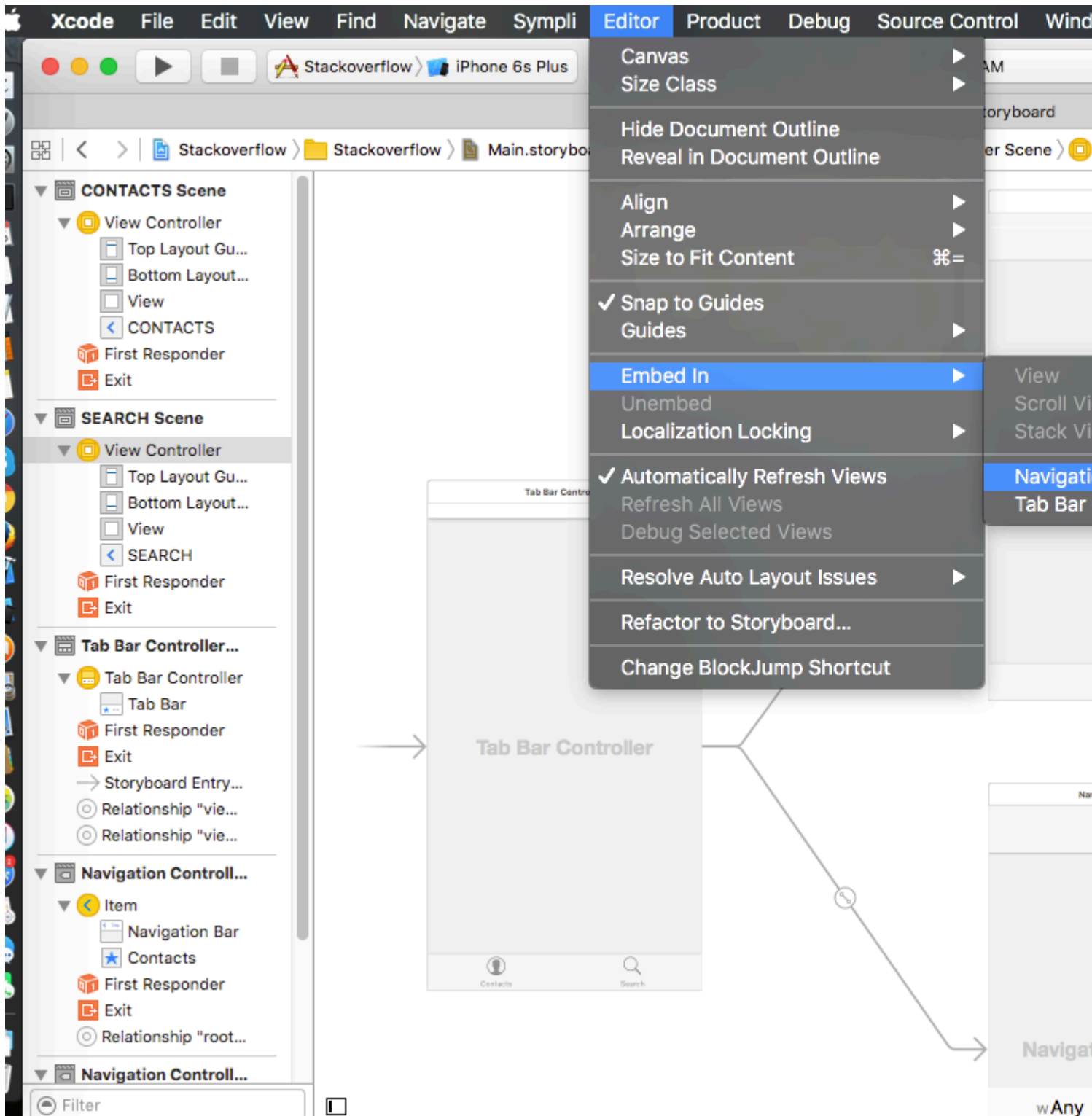
```
self.title = "item"
self.tabBarItem.image = UIImage(named: "item")
self.tabBarItem.selectedImage = UIImage(named: "item_selected")
```

Navigationscontroller mit TabBar

Der Navigations-Controller kann mithilfe des Storyboards in jede Registerkarte eingebettet werden. Es kann wie im Screenshot hinzugefügt werden.

Um einem View Controller, der eine Verbindung mit dem Registerkarten-Controller herstellt, einen Navigations-Controller hinzuzufügen, folgen Sie den Anweisungen

- Wählen Sie den Ansichtscontroller aus, für den Sie den Navigationscontroller hinzufügen müssen. Hier sei es der Search View Controller als Auswahlanzeige.
- **Wählen Sie im Editor- Menü des Xcodes die Option Einbetten in -> Navigations-Controller**



Farbe der Tab-Leistenfarbe

```
[[UITabBar appearance] setTintColor:[UIColor whiteColor]];
[[UITabBar appearance] setBarTintColor:[UIColor tabBarBackgroundColor]];
[[UITabBar appearance] setBackgroundColor:[UIColor tabBarInactiveColor]];
[[UINavigationController appearance] setBarTintColor:[UIColor appBlueColor]];
[[UINavigationController appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationController appearance] setBarStyle:UIBarStyleBlack];
```

UITabBarController mit benutzerdefinierter Farbauswahl

UITabBarController in Swift 3 erstellen Swift 3 Ändern Sie die Bildfarbe und den Titel entsprechend der Auswahl, indem Sie die Farbe der ausgewählten Registerkarten ändern.

```
import UIKit

class TabBarController: UITabBarController {

    override func viewDidLoad() {
        super.viewDidLoad()

        self.navigationController?.isNavigationBarHidden = true

        UITabBar.appearance().tintColor = UIColor.purple

        // set red as selected background color
        let numberOfItems = CGFloat(tabBar.items!.count)
        let tabBarItemSize = CGSize(width: tabBar.frame.width / numberOfItems, height:
tabBar.frame.height)
        tabBar.selectionIndicatorImage =
UIImage.imageWithColor(UIColor.lightText.withAlphaComponent(0.5), size:
tabBarItemSize).resizableImage(withCapInsets: UIEdgeInsets.zero)

        // remove default border
        tabBar.frame.size.width = self.view.frame.width + 4
        tabBar.frame.origin.x = -2
    }

    override func viewWillAppear(_ animated: Bool) {
        // For Images
        let firstViewController:UIViewController = NotificationVC()
        // The following statement is what you need
        let customTabBarItem:UITabBarItem = UITabBarItem(title: nil, image: UIImage(named:
"notification@2x")?.withRenderingMode(UIImageRenderingMode.alwaysOriginal), selectedImage:
UIImage(named: "notification_sel@2x"))
        firstViewController.tabBarItem = customTabBarItem

        for item in self.tabBar.items! {
            let unselectedItem = [NSForegroundColorAttributeName: UIColor.white]
            let selectedItem = [NSForegroundColorAttributeName: UIColor.purple]

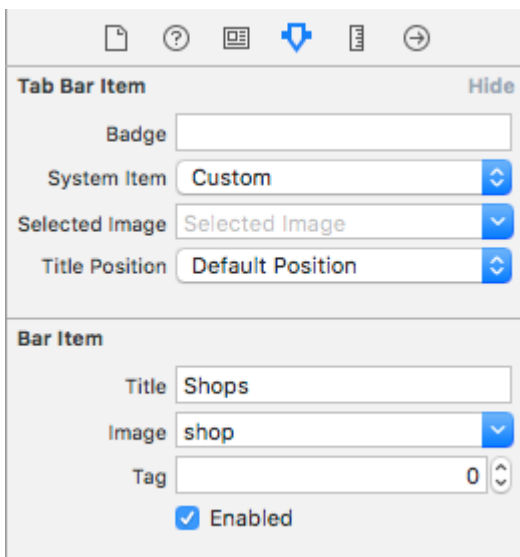
            item.setTitleTextAttributes(unselectedItem, for: .normal)
            item.setTitleTextAttributes(selectedItem, for: .selected)
        }
    }
}
```

```

extension UIImage {
    class func colorWithColor(_ color: UIColor, size: CGSize) -> UIImage {
        let rect: CGRect = CGRect(origin: CGPoint(x: 0,y :0), size: CGSize(width: size.width,
height: size.height))
        UIGraphicsBeginImageContextWithOptions(size, false, 0)
        color.setFill()
        UIRectFill(rect)
        let image: UIImage = UIGraphicsGetImageFromCurrentImageContext()!
        UIGraphicsEndImageContext()
        return image
    }
}

```

Bild für die Tab-Leiste auswählen und hier den Tab-Titel einstellen





Auswahl einer anderen Registerkarte



Erstellen Sie programmgesteuert einen Tab-Leisten-Controller ohne Storyboard

```
class AppDelegate: UIResponder, UIApplicationDelegate {  
  
    var window: UIWindow?  
  
    var firstTabNavigationController : UINavigationController!  
    var secondTabNavigationControoller : UINavigationController!  
    var thirdTabNavigationController : UINavigationController!  
    var fourthTabNavigationControoller : UINavigationController!  
    var fifthTabNavigationController : UINavigationController!
```



```

func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    Fabric.with([Crashlytics.self])

    window = UIWindow(frame: UIScreen.main.bounds)

    window?.backgroundColor = UIColor.black

    let tabBarController = UITabBarController()

    firstTabNavigationController = UINavigationController.init(rootViewController:
FirstViewController())
    secondTabNavigationControoller = UINavigationController.init(rootViewController:
SecondViewController())
    thirdTabNavigationController = UINavigationController.init(rootViewController:
ThirdViewController())
    fourthTabNavigationControoller = UINavigationController.init(rootViewController:
FourthViewController())
    fifthTabNavigationController = UINavigationController.init(rootViewController:
FifthViewController())

    tabBarController.viewControllers = [firstTabNavigationController,
secondTabNavigationControoller, thirdTabNavigationController, fourthTabNavigationControoller,
fifthTabNavigationController]

    let item1 = UITabBarItem(title: "Home", image: UIImage(named: "ico-home"), tag: 0)
    let item2 = UITabBarItem(title: "Contest", image: UIImage(named: "ico-contest"), tag:
1)
    let item3 = UITabBarItem(title: "Post a Picture", image: UIImage(named: "ico-photo"),
tag: 2)
    let item4 = UITabBarItem(title: "Prizes", image: UIImage(named: "ico-prizes"), tag:
3)
    let item5 = UITabBarItem(title: "Profile", image: UIImage(named: "ico-profile"), tag:
4)

    firstTabNavigationController.tabBarItem = item1
    secondTabNavigationControoller.tabBarItem = item2
    thirdTabNavigationController.tabBarItem = item3
    fourthTabNavigationControoller.tabBarItem = item4
    fifthTabNavigationController.tabBarItem = item5

    UITabBar.appearance().tintColor = UIColor(red: 0/255.0, green: 146/255.0, blue:
248/255.0, alpha: 1.0)

    self.window?.rootViewController = tabBarController

    window?.makeKeyAndVisible()

    return true
}

```

UITabBarController online lesen: <https://riptutorial.com/de/ios/topic/2763/uitabBarController>

Kapitel 192: UITableView

Einführung

Eine einfache, weit verbreitete und dennoch sehr leistungsfähige Ansicht, die Daten in Listenform mithilfe von Zeilen und einer einzelnen Spalte darstellen kann. Benutzer können in einer Tabellenansicht vertikal durch die Elemente blättern und den Inhalt optional bearbeiten und auswählen.

Syntax

- - (CGFloat) tableView: (UITableView *) tableView heightForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (CGFloat) tableView: (UITableView *) tableView heightForHeaderInSection: (NSInteger) Abschnitt;
- - (CGFloat) tableView: (UITableView *) tableView heightForFooterInSection: (NSInteger) Abschnitt;
- - (UIView *) tableView: (UITableView *) tableView viewForHeaderInSection: (NSInteger) Abschnitt;
- - (UIView *) tableView: (UITableView *) tableView viewForFooterInSection: (NSInteger) Abschnitt;
- - (UITableViewCellAccessoryType) tableView: (UITableView *) tableView AccessoryTypeForRowWithIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView AccessoryButtonTappedForRowWithIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView didSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (UITableViewCellEditingStyle) tableView: (UITableView *) tableView processingStyleForRowAtIndexPath: (NSIndexPath *) indexPath;

- - (NSString *) tableView: (UITableView *) tableView
titleForDeleteConfirmationButtonForRowAtIndexPath: (NSIndexPath *) indexPath
- - (BOOL) tableView: (UITableView *) tableView shouldIndentWhileEditingRowAtIndexPath:
(NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView willBeginEditingRowAtIndexPath:
(NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didEndEditingRowAtIndexPath: (NSIndexPath
*) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView
targetIndexPathForMoveFromRowAtIndexPath: (NSIndexPath *) sourceIndexPath
toProposedIndexPath: (NSIndexPath *) vorgeschlagenDestinationIndexPath;
- - (NSInteger) tableView: (UITableView *) tableView indentationLevelForRowAtIndexPath:
(NSIndexPath *) indexPath;
- - (NSInteger) tableView: (UITableView *) tableView numberOfRowsInSection: (NSInteger)
Abschnitt;
- - (UITableViewCell *) tableView: (UITableView *) tableView cellForRowAtIndexPath:
(NSIndexPath *) indexPath;
- - (NSInteger) numberOfSectionsInTableView: (UITableView *) tableView;
- - (NSString *) tableView: (UITableView *) tableView titleForHeaderInSection: (NSInteger)
Abschnitt; // festgelegter Schriftstil Verwenden Sie die benutzerdefinierte Ansicht (UILabel),
wenn Sie etwas anderes wünschen
- - (NSString *) tableView: (UITableView *) tableView titleForFooterInSection: (NSInteger)
Abschnitt;
- - (BOOL) tableView: (UITableView *) tableView canEditRowAtIndexPath: (NSIndexPath *)
indexPath;
- - (BOOL) tableView: (UITableView *) tableView canMoveRowAtIndexPath: (NSIndexPath *)
indexPath;
- - (NSArray *) sectionIndexTitlesForTableView: (UITableView *) tableView;
- - (NSInteger) tableView: (UITableView *) tableView sectionForSectionIndexTitle: (NSString *)
title atIndex: (NSInteger) index;
- - (void) tableView: (UITableView *) tableView commitEditingStyle:
(UITableViewCellEditingStyle) editStyle forRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView moveRowAtIndexPath: (NSIndexPath *)
sourceIndexPath toIndexPath: (NSIndexPath *) destinationIndexPath;

Bemerkungen

`UITableView` ist eine Unterklasse von `UIScrollView`. Klassen, die dem `UITableViewDelegate` Protokoll folgen, folgen ebenfalls dem `UIScrollViewDelegate` Protokoll. `UITableView` kann nützlich sein, um lange oder unbestimmte Listen durch seine Zellen `UITableViewCell`, während `UIScrollView` besser ist, wenn die Größe der `UIScrollView` Ansichten vorher bekannt ist.

Examples

Zellen, die die Größe selbst bestimmen

In iOS 8 hat Apple die Zelle zur Größenanpassung eingeführt. Gestalten Sie Ihre `UITableViewCell`s explizit mit `Autolayout` und `UITableView` erledigt den Rest für Sie. Die `rowHeight` wird automatisch berechnet. `rowHeight` Wert für `rowHeight` ist standardmäßig `UITableViewAutomaticDimension`.

`UITableView` Eigenschaft `estimatedRowHeight` verwendet wird, wenn Selbst Sizing Zelle Berechnung ist.

Wenn Sie eine Tabellensichtzelle mit eigener Größe erstellen, müssen Sie diese Eigenschaft festlegen und Einschränkungen verwenden, um die Größe der Zelle zu definieren.

- *Apple, UITableView-Dokumentation*

```
self.tableView.estimatedRowHeight = 44.0
```

Beachten Sie, dass der `heightForRowAtIndexPath` des Delegierten der `heightForRowAtIndexPath` *nicht* `heightForRowAtIndexPath` ist, wenn Sie für alle Zellen eine dynamische Höhe wünschen. Setzen Sie einfach die obige Eigenschaft, wenn nötig und vor dem erneuten Laden oder Laden der Tabellenansicht. Sie können jedoch die Höhe bestimmter Zellen festlegen, während andere über die folgende Funktion dynamisch sind:

Schnell

```
override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
    switch indexPath.section {
    case 1:
        return 60
    default:
        return UITableViewAutomaticDimension
    }
}
```

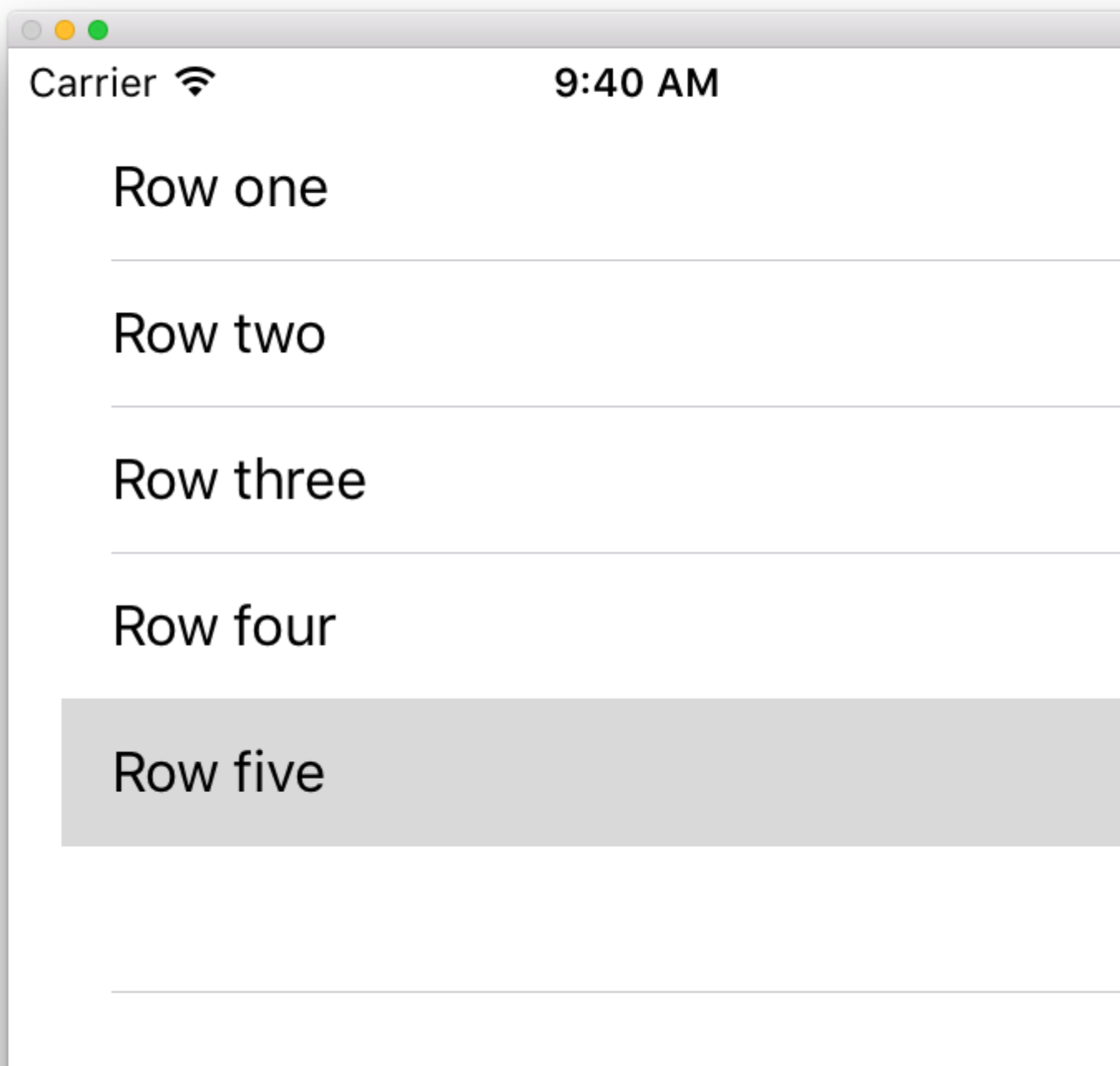
Ziel c

```
-(CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
```

```
{
  switch (indexPath.section) {
    case 1:
      return 60;
    default:
      return UITableViewAutomaticDimension;
  }
}
```

Erstellen einer UITableView

Eine Tabellenansicht ist eine Liste von Zeilen, die ausgewählt werden können. Jede Zeile wird aus einer Datenquelle gefüllt. In diesem Beispiel wird eine einfache Tabellenansicht erstellt, in der jede Zeile eine einzelne Textzeile ist.



Fügen Sie Ihrem Storyboard eine UITableView hinzu

Obwohl es mehrere Möglichkeiten gibt, eine `UITableView` zu erstellen, besteht die einfachste Möglichkeit darin, eine zu einem Storyboard hinzuzufügen. Öffnen Sie Ihr Storyboard und ziehen Sie eine `UITableView` auf Ihren `UIViewController`. Stellen Sie sicher, dass Sie das automatische Layout verwenden, um die Tabelle richtig auszurichten (alle vier Seiten feststecken).

Füllen Sie Ihre Tabelle mit Daten

Um Inhalte dynamisch anzuzeigen (dh aus einer Datenquelle wie einem Array, einem Core-Data-Modell, einem Netzwerkservers usw.) in der Tabellensicht zu laden, müssen Sie die Datenquelle einrichten.

Erstellen einer einfachen Datenquelle

Eine Datenquelle könnte, wie oben erwähnt, alles mit Daten sein. Es liegt ganz bei Ihnen, wie Sie es formatieren und was darin enthalten ist. Die einzige Voraussetzung ist, dass Sie es später lesen können müssen, damit Sie bei Bedarf jede Zeile Ihrer Tabelle mit Daten füllen können.

In diesem Beispiel legen wir einfach ein Array mit einigen Strings (Text) als Datenquelle fest:

Schnell

```
let mydataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]
```

Ziel c

```
// You'll need to define this variable as a global variable (like an @property) so that you can access it later when needed.
NSArray *mydataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];
```

Einrichten Ihrer Datenquelle in Ihrem View Controller

`UITableViewDataSource` Sie sicher, dass Ihr View Controller dem `UITableViewDataSource` Protokoll entspricht.

Schnell

```
class ViewController: UIViewController, UITableViewDataSource {
```

Ziel c

```
@interface ViewController : UIViewController <UITableViewDataSource>
```

Sobald Ihre View - Controller erklärt sie dem **entsprechen** , werden `UITableViewDataSource` (das ist , was wir oben gerade getan), werden Sie zumindest die folgenden Methoden in Ihrer View - Controller - Klasse implementieren *erforderlich*:

- `tableView:numberOfRowsInSection` , werden Sie `tableView:numberOfRowsInSection` , wie viele Zeilen Ihre Tabellenansicht enthalten soll.

```
// Swift

func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return self.mydataArray.count
}
```

- `tableView:cellForRowAtIndexPath` , fordert an, dass Sie eine Zelle für jede in `tableView:numberOfRowsInSection` angegebene Zeile erstellen und zurückgeben. Wenn Sie also sagten, Sie hätten 10 Zeilen benötigt, wird diese Methode zehnmal für jede Zeile aufgerufen, und Sie müssen für jede dieser Zeilen eine Zelle erstellen.

```
// Swift

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    // Create a new cell here. The reuseIdentifier needs to match the reuse
    // identifier from the cell in your Storyboard
    let cell: UITableViewCell =
    tableView.dequeueReusableCellWithIdentifier(reuseIdentifier) as UITableViewCell!

    // Set the label on your cell to the text from your data array
    cell.textLabel?.text = self.mydataArray[indexPath.row]

    return cell
}
```

WARNUNG : Sie dürfen `cellForRowAtIndexPath:` für Zellen in `cellForRowAtIndexPath:` **NICHT** zurückgeben `cellForRowAtIndexPath:` Dadurch stürzt Ihre App ab und in der Konsole wird folgender Fehler angezeigt:

```
Uncaught exception 'NSInternalInconsistencyException', reason: 'UITableView
dataSource must return a cell from tableView:cellForRowAtIndexPath:'
```

Verbinden der Datenquelle der Tabellensicht mit Ihrem Ansichtscontroller

Sie können dies entweder über Code tun, indem Sie die `dataSource` Ihrer Tabelle für Ihren View-Controller als `self.dataSource` . Sie können auch Ihre Tabellenansicht in Ihrem Storyboard auswählen, den Inspector für Attribute öffnen, das `dataSource` "Outlets" auswählen und von `dataSource` auf Ihren View-Controller ziehen (**HINWEIS** : Stellen Sie sicher, dass Sie eine Verbindung zum `UIViewController` herstellen, **nicht zu** einem `UIView` oder einem anderen

Objekt *in* Ihrem UIViewController).

Zeilenauswahlen behandeln

Wenn ein Benutzer in der Tabellensicht auf eine Zeile tippt, sollten Sie im Allgemeinen etwas tun - um zu antworten. Wenn Sie in vielen Apps auf eine Zeile tippen, werden weitere Informationen zu dem Element angezeigt, auf das Sie tippen. Denken Sie an die Nachrichten-App: Wenn Sie auf die Zeile mit einem Ihrer Kontakte tippen, wird die Konversation mit dieser Person auf dem Bildschirm angezeigt.

Dazu müssen Sie das `UITableViewDelegate` Protokoll `UITableViewDelegate` . Dies entspricht der Konformität mit dem Datenquellenprotokoll. Dieses Mal fügen Sie es jedoch neben `UITableViewDataSource` und trennen es mit einem Komma. Es sollte also so aussehen:

Schnell

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
```

Ziel c

```
@interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
```

Es sind keine erforderlichen Methoden zum Implementieren für den Delegaten der Tabellensicht erforderlich. Für die Auswahl von Zeilen müssen Sie jedoch die folgende Methode verwenden:

- `tableView:didSelectRowAtIndexPath` , dies wird aufgerufen, wenn auf eine Zeile `tableView:didSelectRowAtIndexPath` , `tableView:didSelectRowAtIndexPath` Sie etwas tun können. In unserem Beispiel drucken wir einfach eine Bestätigungsanweisung in das Xcode-Protokoll.

```
// Swift

func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
    print("You tapped cell number \(indexPath.row).")
}

// Objective-C

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}
```

Das finale Resultat

Siehe unten für die vollständige Einrichtung mit nur Code, keine Erklärung.

Schnell

```
import UIKit
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // Data model: These strings will be the data for the table view cells
    let mydataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]

    // cell reuse id (cells that scroll out of view can be reused)
    let reuseIdentifier = "cell"

    // don't forget to hook this up from the storyboard
    @IBOutlet var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Register the table view cell class and its reuse id
        myTableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:
cellReuseIdentifier)

        // This view controller itself will provide the delegate methods and row data for the
table view.
        myTableView.delegate = self
        myTableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.mydataArray.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {

        // create a new cell if needed or reuse an old one
        let cell:UITableViewCell =
tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

        // set the text from the data model
        cell.textLabel?.text = self.mydataArray[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }
}
```

Ziel c

ViewController.h

```
#import <UIKit/UIKit.h>
```

```

@interface ViewController: UIViewController <UITableViewDelegate, UITableViewDataSource> {
    IBOutlet UITableView *myTableView;
    NSArray *myDataArray;
}

@end

```

ViewController.m

```

#import "ViewController.h"

// cell reuse id (cells that scroll out of view can be reused)
NSString * _Nonnull cellReuseIdentifier = @"cell";

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    // Data model: These strings will be the data for the table view cells
    myDataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];

    // Register the table view cell class and its reuse id
    [myTableView registerClass:[UITableViewCell class]
    forCellReuseIdentifier:cellReuseIdentifier];

    // This view controller itself will provide the delegate methods and row data for the
    table view.
    myTableView.delegate = self;
    myTableView.dataSource = self;
}

// number of rows in table view
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return myDataArray.count;
}

// create a cell for each table view row
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    // create a new cell if needed or reuse an old one
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellReuseIdentifier];

    // set the text from the data model
    cell.textLabel.text = myDataArray[indexPath.row];

    return cell;
}

// method to run when table view cell is tapped
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}

@end

```

Delegieren und Datenquelle

Mit `UITableViewDelegate` wird festgelegt, wie die Tabelle angezeigt wird. `UITableViewDataSource` `UITableView` die Daten von `UITableView` definiert. Es gibt zwei erforderliche und viele optionale Methoden, mit denen Größe, Abschnitte, Überschriften und Zellen in der `UITableView` .

UITableViewDataSource

Erforderliche Methoden

`numberOfRowsInSection`: Diese Methode definiert, wie viele Zellen in jedem Abschnitt der Tabellenansicht angezeigt werden.

Ziel c

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count
}
```

`cellForRowAtIndexPath`: dieser Methode werden die Zellen der `UITableView` erstellt und konfiguriert. Sollte entweder eine `UITableViewCell` oder eine benutzerdefinierte Unterklasse zurückgeben.

Hinweis: Bei Verwendung von `dequeueReusableCellWithIdentifier:forIndexPath`: müssen Sie die Klasse oder die Spitze für diesen Bezeichner mithilfe der `UITableView` `registerClass:forCellReuseIdentifier` oder `registerNib:forCellReuseIdentifier` des `UITableView` `registerClass:forCellReuseIdentifier` . Normalerweise wird dies in der `UIViewController` -Methode von `viewDidLoad` .

Ziel c

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    MyCustomCell *cell = [tableView dequeueReusableCellWithIdentifier:@"MyCustomCell"
                                                                forIndexPath:indexPath];

    // All additional customization goes here
    cell.titleLabel.text = [NSString stringWithFormat:@"Title Row %lu", indexPath.row];

    return cell;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("MyCustomCell",
forIndexPath:indexPath)

    // All additional customization goes here
    cell.titleLabel.text = String(format:"Title Row %lu", indexPath.row)

    return cell
}
```

Optionale Methoden

`titleForHeaderInSection`: Definiert eine Zeichenfolge als Titel für jeden Abschnittsheader in der Tabellenansicht. Bei dieser Methode kann nur der Titel geändert werden. Weitere Anpassungen können durch Definieren der Ansicht für den Header vorgenommen werden.

Ziel c

```
- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section {
    switch(section) {
        case 0:
            return @"Title 1";
            break;

        case 1:
            return @"Title 2";
            break;

        default:
            return nil;
            break;
    }
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    switch section {
        case 0:
            return "Title 1"
        case 1:
            return "Title 2"
        default:
            return nil
    }
}
```

`titleForFooterInSection`: Definiert eine Zeichenfolge als Titel für jeden Abschnittsheader in der Tabellenansicht.

Ziel c

```
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger)section {
    return @"Footer text";
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForFooterInSection section: Int) -> String? {
    return "Footer text"
}
```

`canEditRowAtIndexPath`: Wird verwendet, um zu bestimmen, ob die Bearbeitungsoberfläche für die angegebene Zeile angezeigt werden soll. Sollte `YES` wenn die angegebene Zeile gelöscht oder hinzugefügt werden kann.

Ziel c

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return YES;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool
{
    return true
}
```

`commitEditingStyle:forRowAtIndexPath` Sollte die Arbeit ausführen, die für das Hinzufügen oder Entfernen der angegebenen Zeile erforderlich ist. Entfernen Sie beispielsweise die Zelle aus der `UITableView` mit Animation und entfernen Sie das zugehörige Objekt aus dem Datenmodell der Tabelle.

Ziel c

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath {
    switch (editingStyle) {
        case UITableViewCellEditingStyleInsert:
            // Insert new data into the backing data model here
            [self insertNewDataIntoDataModel];
            [tableView insertRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        case UITableViewCellEditingStyleDelete:
            [self removeDataFromDataModelAtIndex:indexPath.row];
            [tableView deleteRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
            break;
    }
}
```

```
}
```

Swift 3

```
func tableView(_ tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {
    switch editingStyle {
        case .Insert:
            self.insertNewDataIntoDataModel()
            tableView.insertRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        case .Delete:
            self.removeDataFromDataModelAtIndex(indexPath.row)
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
    }
}
```

`editActions:forRowAt` Ermöglicht das Hinzufügen von zusätzlichen Aktionen oder Schaltflächen zum Bearbeitungsmodus einer Zeile in einer `UITableView`. Wenn Sie z. B. zwei Schaltflächen wünschen, eine Schaltfläche zum Bearbeiten und Löschen, wenn der Benutzer die Zeile bearbeitet, wischen Sie diese Methode.

Swift 3

```
override func tableView(_ tableView: UITableView, editActionsForRowAt indexPath: IndexPath) ->
[UITableViewRowAction]? {
    // In the handler you will get passed the action as well as the indexPath for
    // the row that is being edited
    let editAction = UITableViewRowAction(style: .normal, title: "Edit", handler: { [unowned
self] action, indexPath in
        // Do something when edit is tapped
    })

    // Change the color of the edit action
    editAction.backgroundColor = UIColor.blue

    let deleteAction = UITableViewRowAction(style: .destructive, title: "Delete", handler: {
[unowned self] action, indexPath in
        // Handel the delete event
    })

    return [deleteAction, editAction]
}
```

UITableViewDelegate

Alle Methoden in `UITableViewDelegate` sind optional, aber ein Delegat, der sie implementiert, ermöglicht zusätzliche Funktionen für die `UITableView`.

`numberOfSectionsInTableView`: Standardmäßig wird 1 zurückgegeben. Die Unterstützung

mehrerer Abschnitte wird jedoch durch die Rückgabe einer anderen Anzahl von Abschnitten aktiviert.

Ziel c

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.numSections;
}
```

Swift 3

```
func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return self.numSections
}
```

`viewForHeaderInSection` Ermöglicht die Konfiguration einer benutzerdefinierten Ansicht als Header für den Abschnitt.

Ziel c

```
- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section {

    UIView *view = [[UIView alloc] initWithFrame:CGRectMake(0, 0,
CGRectGetWidth(tableView.frame), 22)];
    view.backgroundColor = [UIColor groupTableViewBackgroundColor];

    UILabel *label = [[UILabel alloc] init];
    label.font = [UIFont systemFontOfSize:12];
    label.textColor = [UIColor darkGrayColor];

    switch (section) {
        case 1: {
            label.text = @"Title";
            label.frame = labelFrame;

            UIButton *more = [[UIButton alloc] initWithFrame:btnFrame];
            [more setTitle:@"See more" forState:UIControlStateNormal];
            [more.titleLabel setFont:[UIFont systemFontOfSize:12]];
            [view addSubview:more];
        } break;

        default:
            label.frame = CGRectMake(0, 0, 0, 0);
            break;
    }

    [view addSubview:label];
    return view;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
    let view = UIView(frame: CGRect(x: 0, y: 0, width: tableView.frame.size.width, height:
22))
}
```

```

view.backgroundColor = UIColor.groupTableViewBackgroundColor()

let label = UILabel()
label.font = UIFont.systemFont(ofSize:12)
label.textColor = UIColor.darkGrayColor()

switch section {
    case 1:
        label.text = "Title"
        label.frame = labelFrame

        let more = UIButton(frame: btnFrame)
        more.setTitle("See more", forState:.Normal)
        view.addSubview(more)

    default:
        label.frame = CGRect.zero
}

view.addSubview(label)
return view;
}

```

`heightForRowAtIndexPath`: Definieren Sie die Höhe jeder Zelle in der Tabellenansicht.

Ziel c

```

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 44;
}

```

Swift 3

```

func tableView(_ tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) ->
CGFloat {
    return 44
}

```

`heightForHeaderInSection`: und `heightForFooterInSection` Definiert die Höhe für die Kopf- und Fußzeile jedes Abschnitts in der Tabellensicht

Ziel c

```

- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return 33;
}

```

Swift 3

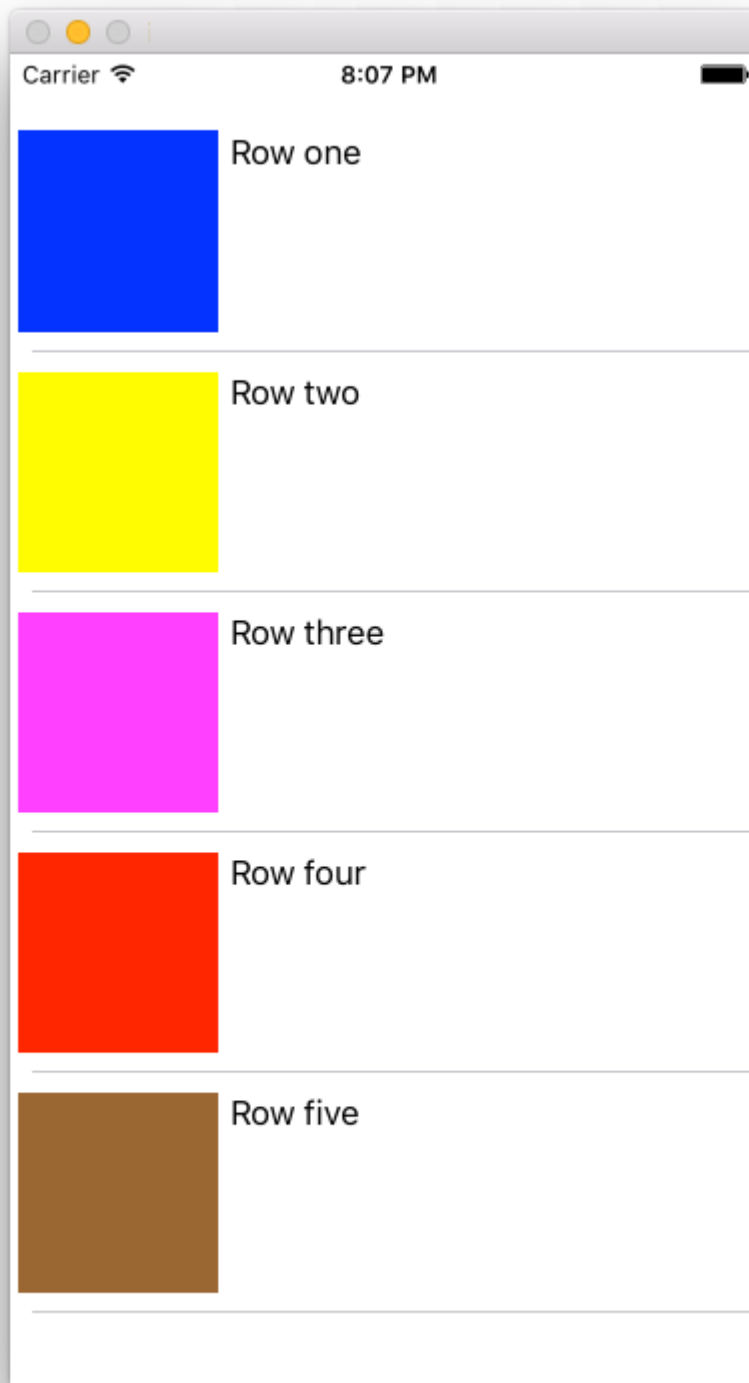
```

func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
    return 33
}

```


Kundenspezifische Zellen

Das Anpassen einer `UITableViewCell` kann sehr leistungsfähige, dynamische und `UITableViewCell` Schnittstellen ermöglichen. Mit umfangreichen Anpassungen und in Kombination mit anderen Techniken können Sie beispielsweise bestimmte Eigenschaften oder Oberflächenelemente bei Änderungen aktualisieren, Objekte in der Zelle animieren oder zeichnen, Video effizient laden, wenn der Benutzer einen Bildlauf durchführt, oder sogar Bilder anzeigen, wenn sie von einem Computer heruntergeladen werden Netzwerk. Die Möglichkeiten hier sind nahezu unbegrenzt. Nachfolgend finden Sie ein einfaches Beispiel, wie eine benutzerdefinierte Zelle aussehen kann.



Dieser Abschnitt behandelt die Grundlagen und wird hoffentlich erweitert, um komplexere Prozesse wie die oben beschriebenen detailliert darzustellen.

Erstellen Sie Ihre eigene Zelle

Erstellen Sie zunächst eine neue Unterklasse von `UITableViewCell` (erstellen Sie eine neue Cocoa Touch-Klasse in Xcode und legen Sie `UITableViewCell` als Superklasse fest). Nachfolgend sehen Sie Ihren Code nach der Unterklasse.

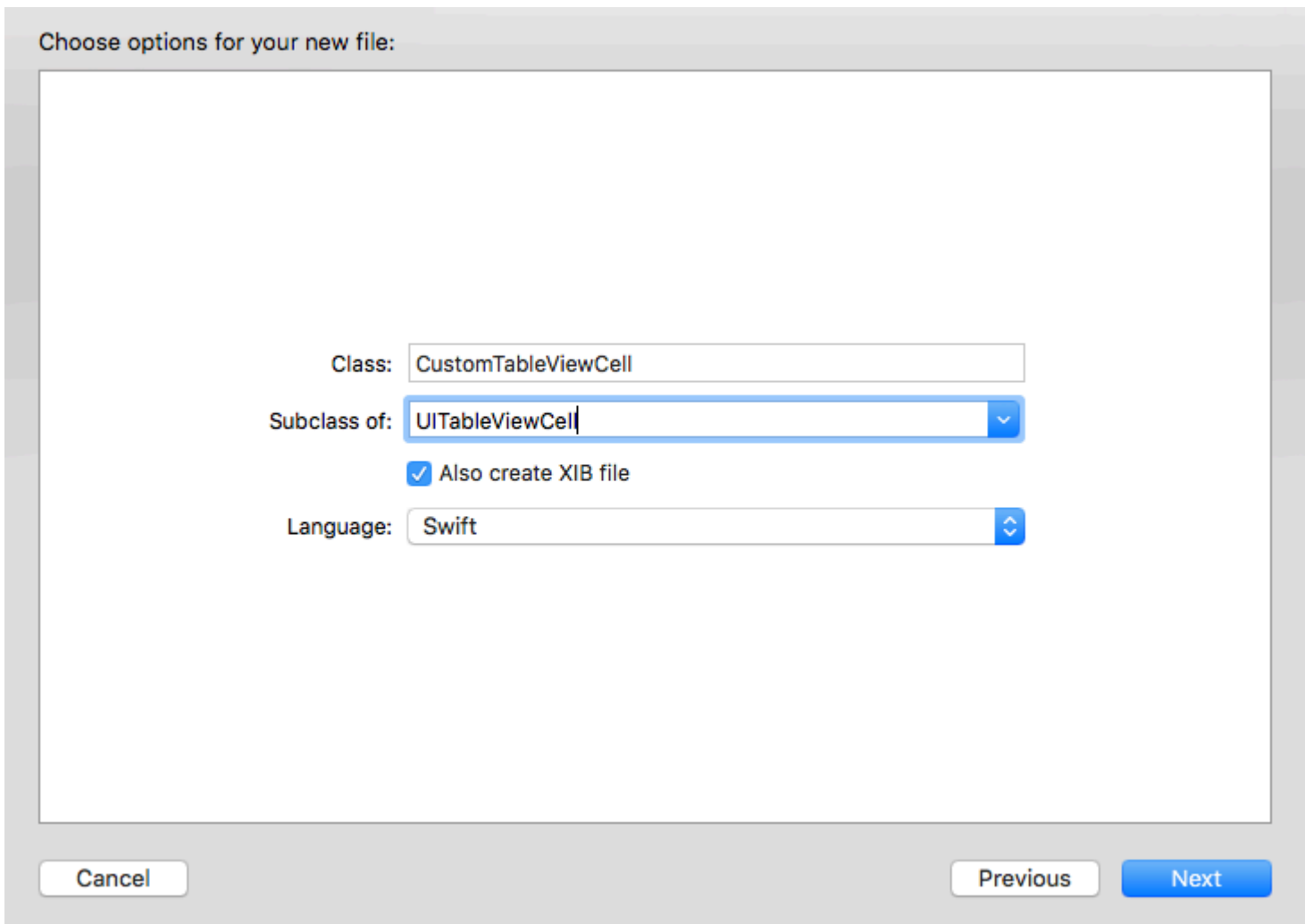
Schnell

```
class CustomTableViewCell: UITableViewCell {
    static var identifier: String {
        return NSStringFromClass(self)
    }

    var customLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
        customLabel = UILabel(frame: CGRect(x: 0, y: 0, width: contentView.frame.width,
height: contentView.frame.height))
        customLabel.textAlignment = .center
        contentView.addSubview(customLabel)
    }
}
```

Aktivieren Sie optional die Option "XIB-Datei erstellen", wenn Sie eine neue Datei zum Anpassen mit dem Interface Builder erstellen. In dem Fall verbinden Sie `customLabel` als `@IBOutlet`



`UITableViewCell` in einem `UIViewController`, der die `tableView`, die Klasse der neuen benutzerdefinierten Zelle (siehe unten). *Beachten Sie, dass dies nur notwendig ist, wenn Sie die Zelle nicht mit einem Storyboard in der Benutzeroberfläche Ihrer Tabellenansicht entwerfen.*

Schnell

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Register Cell Class
    tableView.register(CustomTableViewCell.self, forCellReuseIdentifier:
CustomTableViewCell.identifier)
}
```

Wenn Sie sich für eine XIB-Datei entschieden haben, `registerNib` stattdessen `registerNib`:

Schnell

```
// Register Nib
tableView.register(UINib(nibName: CustomTableViewCell.identifier, bundle: nil),
forCellReuseIdentifier: CustomTableViewCell.identifier)
```

Nun, da Ihre `tableView` über Ihre benutzerdefinierte Zelle `tableView` weiß, können Sie sie in `cellForRowAtIndexPath`:

Schnell

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    // Load the CustomTableViewCell. Make sure the identifier supplied here matches the one
    from your cell
    let cell: CustomTableViewCell =
tableView.dequeueReusableCellWithIdentifier(CustomTableViewCell.identifier) as!
CustomTableViewCell

    // This is where the magic happens - setting a custom property on your very own cell
    cell.customLabel.text = "My Custom Cell"

    return cell
}
```

UITableViewCells erweitern und reduzieren

Fügen Sie in Ihrem Storyboard ein UITableView-Objekt zu Ihrem UIViewController hinzu, und lassen Sie es die gesamte Ansicht abdecken. UITableViewDataSource UITableViewDelegate Verbindungen UITableViewDataSource und UITableViewDelegate .

Ziel c

In Ihrer .h Datei

```
NSMutableArray *arrayForBool;
NSMutableArray *sectionTitleArray;
```

In Ihrer .m Datei

```
- (void)viewDidLoad {
    [super viewDidLoad];

    arrayForBool = [[NSMutableArray alloc] init];
    sectionTitleArray = @[@"Sam",@"Sanju",@"John",@"Staffy"];

    for (int i=0; i<[sectionTitleArray count]; i++) {
        [arrayForBool addObject:[NSNumber numberWithInt:NO]];
    }

    _tableView.dataSource = self;
    _tableView.delegate = self;
}

// Declare number of rows in section
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    if ([[arrayForBool objectAtIndex:section] boolValue]) {
        return section+2;
    } else {
        return 0;
    }
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
```

```

static NSString *cellid=@"hello";
UITableViewCell *cell=[tableView dequeueReusableCellWithIdentifier:cellid];
if (cell==nil) {
    cell=[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:cellid];
}
    BOOL manyCells = [[arrayForBool objectAtIndex:indexPath.section] boolValue];

    /** If the section supposed to be closed*****/
    if(!manyCells){
        cell.backgroundColor=[UIColor clearColor];
        cell.textLabel.text=@"";
    }
    /** If the section supposed to be Opened*****/
    else{
        cell.textLabel.text=[NSString stringWithFormat:@"%d", [sectionTitleArray
objectAtIndex:indexPath.section], indexPath.row+1];
        cell.backgroundColor=[UIColor whiteColor];
        cell.selectionStyle=UITableViewCellSelectionStyleNone ;
    }
cell.textLabel.textColor=[UIColor blackColor];

/** Add a custom Separator with cell*/
    UIView* separatorLineView = [[UIView alloc] initWithFrame:CGRectMakeMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
separatorLineView.backgroundColor = [UIColor blackColor];
[cell.contentView addSubview:separatorLineView];
return cell;
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return [sectionTitleArray count];
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
/***** Close the section, once the data is selected
*****/
[arrayForBool replaceObjectAtIndex:indexPath.section withObject:[NSNumber numberWithInt:NO]];

[_expandableTableView reloadSections:[NSIndexPath indexPathWithIndex:indexPath.section]
withRowAnimation:UITableViewRowAnimationAutomatic];

}

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath
{
if ([[arrayForBool objectAtIndex:indexPath.section] boolValue]) {
    return 40;
}
return 0;

}

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger) section
{

```

```

UIView *sectionView=[[UIView alloc]initWithFrame:CGRectMake(0, 0, 280,40)];
sectionView.tag=section;
UILabel *viewLabel=[[UILabel alloc]initWithFrame:CGRectMake(10, 0,
_expandableTableView.frame.size.width-10, 40)];
viewLabel.backgroundColor=[UIColor clearColor];
viewLabel.textColor=[UIColor blackColor];
viewLabel.font=[UIFont systemFontOfSize:15];
viewLabel.text=[NSString stringWithFormat:@"List of %@",[sectionTitleArray
objectAtIndex:section]];
[sectionView addSubview:viewLabel];
    /***** Add a custom Separator with Section view *****/
UIView* separatorLineView = [[UIView alloc] initWithFrame:CGRectMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
separatorLineView.backgroundColor = [UIColor blackColor];
[sectionView addSubview:separatorLineView];

/***** Add UITapGestureRecognizer to SectionView *****/

UITapGestureRecognizer *headerTapped = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(sectionHeaderTapped)];
[sectionView addGestureRecognizer:headerTapped];

return sectionView;

}

- (void)sectionHeaderTapped:(UITapGestureRecognizer *)gestureRecognizer{

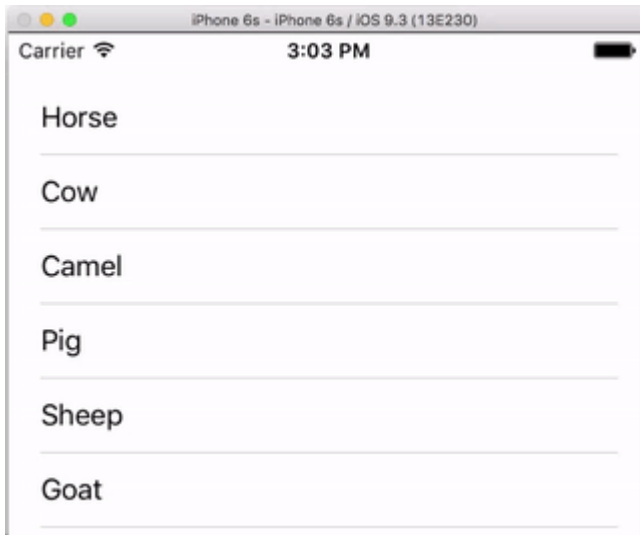
NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSection:gestureRecognizer.view.tag];
if (indexPath.row == 0) {
    BOOL collapsed = [[arrayForBool objectAtIndex:indexPath.section] boolValue];
    for (int i=0; i<[sectionTitleArray count]; i++) {
        if (indexPath.section==i) {
            [arrayForBool replaceObjectAtIndex:i withObject:[NSNumber
numberWithBool:!collapsed]];
        }
    }
    [_expandableTableView reloadData:[NSIndexPath
indexPathSetWithIndex:gestureRecognizer.view.tag]
withRowAnimation:UITableViewRowAnimationAutomatic];

}
}
}

```

Zum Löschen von Zeilen streichen

Ich finde es immer schön, ein sehr einfaches, in sich geschlossenes Beispiel zu haben, so dass nichts angenommen wird, wenn ich eine neue Aufgabe lerne. Diese Antwort dient zum Löschen von `UITableView` Zeilen. Das Projekt läuft folgendermaßen ab:



Dieses Projekt basiert auf dem [UITableView-Beispiel für Swift](#) .

Fügen Sie den Code hinzu

Erstellen Sie ein neues Projekt, und ersetzen Sie den Code ViewController.swift durch den folgenden Code.

```
import UIKit
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // These strings will be the data for the table view cells
    var animals: [String] = ["Horse", "Cow", "Camel", "Pig", "Sheep", "Goat"]

    let reuseIdentifier = "cell"

    @IBOutlet var tableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // It is possible to do the following three things in the Interface Builder
        // rather than in code if you prefer.
        self.tableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:
cellReuseIdentifier)
        tableView.delegate = self
        tableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.animals.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {

        let cell:UITableViewCell =
self.tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!
```

```

        cell.textLabel?.text = self.animals[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }

    // this method handles row deletion
    func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellStyle, forRowAtIndexPath indexPath: NSIndexPath) {

        if editingStyle == .Delete {

            // remove the item from the data model
            animals.removeAtIndex(indexPath.row)

            // delete the table view row
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)

        } else if editingStyle == .Insert {
            // Not used in our example, but if you were adding a new row, this is where you
            would do it.
        }
    }
}

```

Die Single-Key-Methode im obigen Code, die das Löschen von Zeilen ermöglicht, ist die letzte. Hier ist es wieder zur Betonung:

```

func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellStyle, forRowAtIndexPath indexPath: NSIndexPath) {

    if editingStyle == .Delete {

        // remove the item from the data model
        animals.removeAtIndex(indexPath.row)

        // delete the table view row
        tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
    }
}

```

Storyboard

Fügen `UITableView` dem View Controller im Storyboard eine `UITableView` . Verwenden Sie das automatische Layout, um die vier Seiten der Tabellenansicht an den Rändern des View Controllers zu befestigen. Ziehen Sie das Drag & Drop von der Tabellensicht im Storyboard auf die `@IBOutlet var tableView: UITableView!` Zeile im Code.

Fertig

Das ist alles. Sie sollten Ihre App jetzt ausführen und Zeilen löschen können, indem Sie nach links wischen und auf "Löschen" tippen.

Anmerkungen

- Dies ist nur unter iOS 8 verfügbar. Weitere Informationen finden Sie in [dieser Antwort](#) .
- Wenn Sie die Anzahl der angezeigten Schaltflächen oder den Schaltflächentext ändern müssen, finden Sie weitere Informationen in [dieser Antwort](#) .

Lesen Sie weiter

- [So erstellen Sie eine durchzupfbare Tabellenzelle mit Aktionen - ohne verrückte Muttern mit Bildlaufansichten](#)
- [Apple-Dokumentation](#)

Trennlinien

Breite der Trennlinien bearbeiten

Sie können festlegen, dass die `layoutMargins:` Ihrer Tabellenansicht die Breite der Tabelle auf verschiedene Breiten erweitern, indem Sie die Eigenschaft `layoutMargins:` in Ihren Zellen ändern. Dies kann auf verschiedene Weise erreicht werden.

Ändern der Trennlinien für bestimmte Zellen

Entweder in der Tabellenansicht der Datenquelle `cellForRowAtIndexPath:` Methode oder der `willDisplayCell:` Methode, stellen Sie die Zelle `layoutMargins:` Eigenschaft `UIEdgeInsetsZero` (erstreckt sich auf volle Breite der Tabelle), oder was auch immer Sie hier wünschen.

Ziel c

```
[cell setLayoutMargins:UIEdgeInsetsZero];  
  
// May also use separatorInset  
[cell setSeparatorInset:UIEdgeInsetsZero];
```

Schnell

```
func tableView(tableView: UITableView, willDisplayCell cell: UITableViewCell,  
forRowAtIndexPath indexPath: NSIndexPath) {  
    cell.separatorInset = UIEdgeInsetsZero
```

```
cell.layoutMargins = UIEdgeInsetsZero
}

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell
{
    cell.separatorInset = UIEdgeInsetsZero
    cell.layoutMargins = UIEdgeInsetsZero
}
```

Entfernen Sie alle Trennlinien

Die dünnen grauen Linien zwischen den einzelnen Zellen sind möglicherweise nicht genau das, was Sie sich wünschen. Es ist ziemlich einfach, sie vor den Augen zu verbergen.

`UIViewController` in Ihrer umfassenden `UIViewController` -Methode `viewDidLoad`: den folgenden Code hinzu. Sie können diese Eigenschaft auch jederzeit festlegen, bevor Sie die Tabellensicht laden oder neu laden (muss nicht unbedingt in der `viewDidLoad`: -Methode enthalten sein).

Schnell:

```
tableView.separatorStyle = .None
```

Ziel c:

```
tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
```

Alternativ können Sie die Eigenschaft in Ihrem Storyboard oder XIB ändern, indem Sie `tableView` auswählen und das `separator` (unter dem Attributinspektor) auf `None` .

Überschüssige Trennlinien ausblenden

Sie können die `UITableViewCell` Trennlinien für leere Zellen ausblenden, indem Sie eine leere Fußzeilenansicht am unteren Rand einer `UITableView` festlegen:

Schnell

```
tableView.tableFooterView = UIView()
```

Ziel c

```
tableView.tableFooterView = [[UIView alloc] initWithFrame:CGRectZero];
```

Kapitel 193: UITableViewCell

Einführung

Benutzerdefinierte Zellen-Xib-Datei verwendet die Zellkategorieklasse, es ist keine Registrierung der Nib-Datei erforderlich

Examples

Xib-Datei von UITableViewCell

Erstellen Sie eine `UITableView` .

UITableViewCell + RRCell.h-Datei

```
#import <UIKit/UIKit.h>

@interface UITableViewCell (RRCell)

-(id)initWithOwner:(id)owner;

@end
```

UITableViewCell + RRCell.m-Datei

```
#import "UITableViewCell+RRCell.h"

@implementation UITableViewCell (RRCell)

#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wobjc-designated-initializers"

-(id)initWithOwner:(id)owner {

    if (self = [super init]) {

        NSArray *nib = [[NSBundle mainBundle]loadNibNamed:NSStringFromClass([self class])
owner:self options:nil];
        self = [nib objectAtIndex:0];
    }
    return self;
}

#pragma clang diagnostic pop

@end
```

Importieren Sie die `cellForRowAtIndexPath` für die Verwendung dieser Methode in die `cellForRowAtIndexPath` Methode

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    //Creted custom cell xib file to load by cell category class
    CustomCell *cell = [[CustomCell alloc] initWithOwner:self];

    return cell;
}
```

UITableViewCell online lesen: <https://riptutorial.com/de/ios/topic/10101/uitableviewcell>

Kapitel 194: UITableViewController

Einführung

UITableViewController-Controllerobjekt, das eine Tabellensicht verwaltet. Für bestimmte Situationen wird die Verwendung von UITableViewController empfohlen, z. B. wenn Sie viele Zellen haben und einige über ein UITextfeld verfügen.

Examples

TableView mit dynamischen Eigenschaften mit tableViewCellStyle basic.

```
override func numberOfSections(in tableView: UITableView) -> Int {
    // You need to return minimum one to show the cell inside the tableView
    return 1
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableView.
    return 3
}

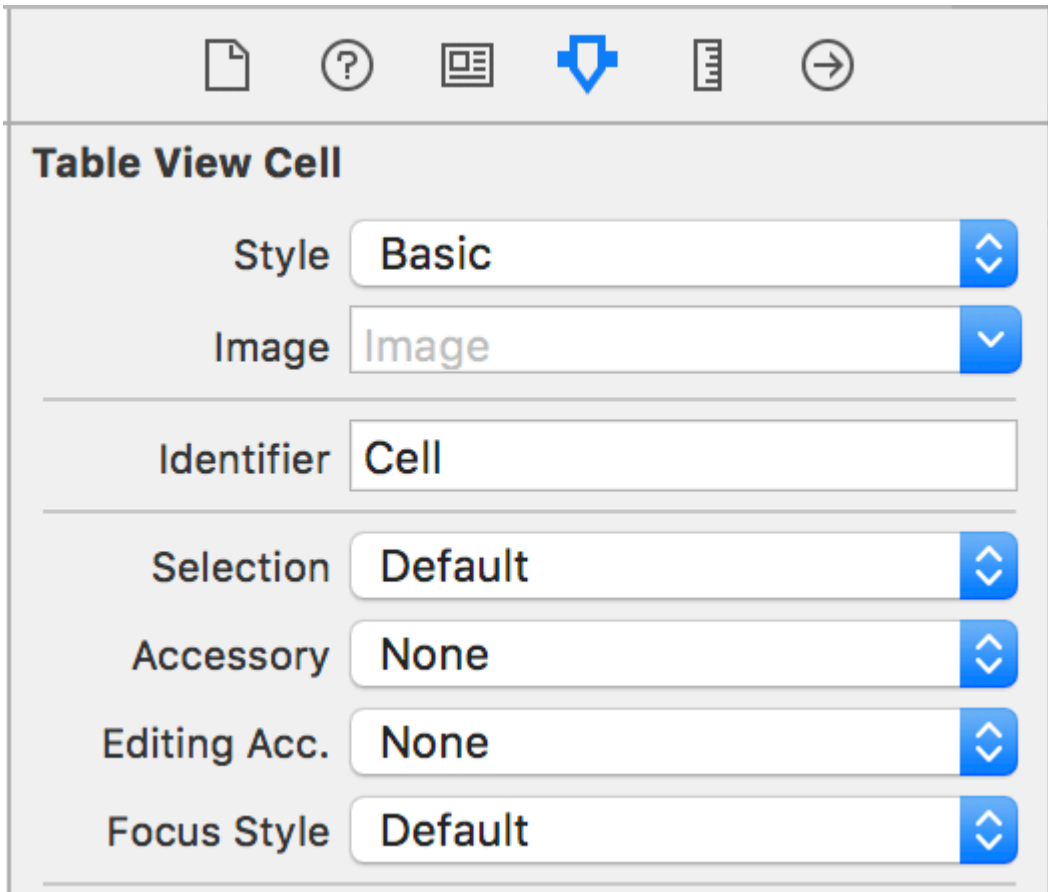
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
    // identifier string should be same as what you have entered in the cell Attribute inspector -
    > identifier (see the image).

    // Configure the cell...
    cell.textLabel?.text = "Cell \(indexPath.row) :" + "Hello"
    //cell have different style Custom, basic, right detail, left detail, subtitle.
    //For custom you can use your own objects and constrains, for other styles all
    //is ready just select according to your design. (see the image for changing the style)

    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    // this delegate method will trigger when you click a cell
}
```



TableView mit benutzerdefinierter Zelle

Für eine benutzerdefinierte Tableview-Zelle benötigen Sie eine Klasse, die eine Unterklasse von `UITableViewCell` ist.

```
class TableViewCell: UITableViewCell {  
  
    @IBOutlet weak var lblTitle: UILabel!  
  
    override func awakeFromNib() {  
        super.awakeFromNib()  
        // Initialization code  
    }  
  
    override func setSelected(_ selected: Bool, animated: Bool) {  
        super.setSelected(selected, animated: animated)  
  
        // Configure the view for the selected state  
    }  
  
}
```

Ihre Tableview-Delegierten

```
override func numberOfSections(in tableView: UITableView) -> Int {  
    // You need to return minimum one to show the cell inside the tableview  
    return 1  
}
```

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableview.
    return 3
}

override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as!
    TableViewCell
    // identifier string should be same as what you have entered in the cell Attribute
    inspector -> identifier.

    // Configure the cell...
    cell.lblTitle.text = "Cell \(indexPath.row) :" + "Hello"

    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    // this delegate method will trigger when you click a cell
}
```

UITableViewController online lesen: <https://riptutorial.com/de/ios/topic/10953/uitableviewController>

Kapitel 195: UI-Test

Syntax

- XCUIApplication () // Proxy für eine Anwendung. Die Informationen zur Identifizierung der Anwendung werden in den Xcode-Zieleinstellungen als "Zielanwendung" angegeben.
- XCUIElement () // Ein Benutzeroberflächenelement in einer Anwendung.

Examples

Testdateien zu Xcode Project hinzufügen

Beim Erstellen des Projekts

Sie sollten im Projekterstellungsdialog "UI-Tests einbeziehen" aktivieren.

Language:

Devices:

Use Core Data

Include Unit Tests

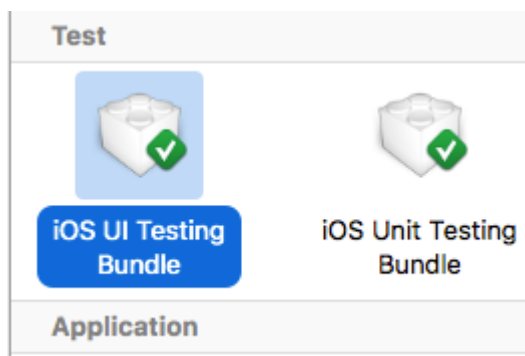
Include UI Tests

Nachdem Sie das Projekt erstellt haben

Wenn Sie beim Erstellen eines Projekts die Überprüfung des `UI target` verpasst haben, können Sie später immer ein Testziel hinzufügen.

Setps:

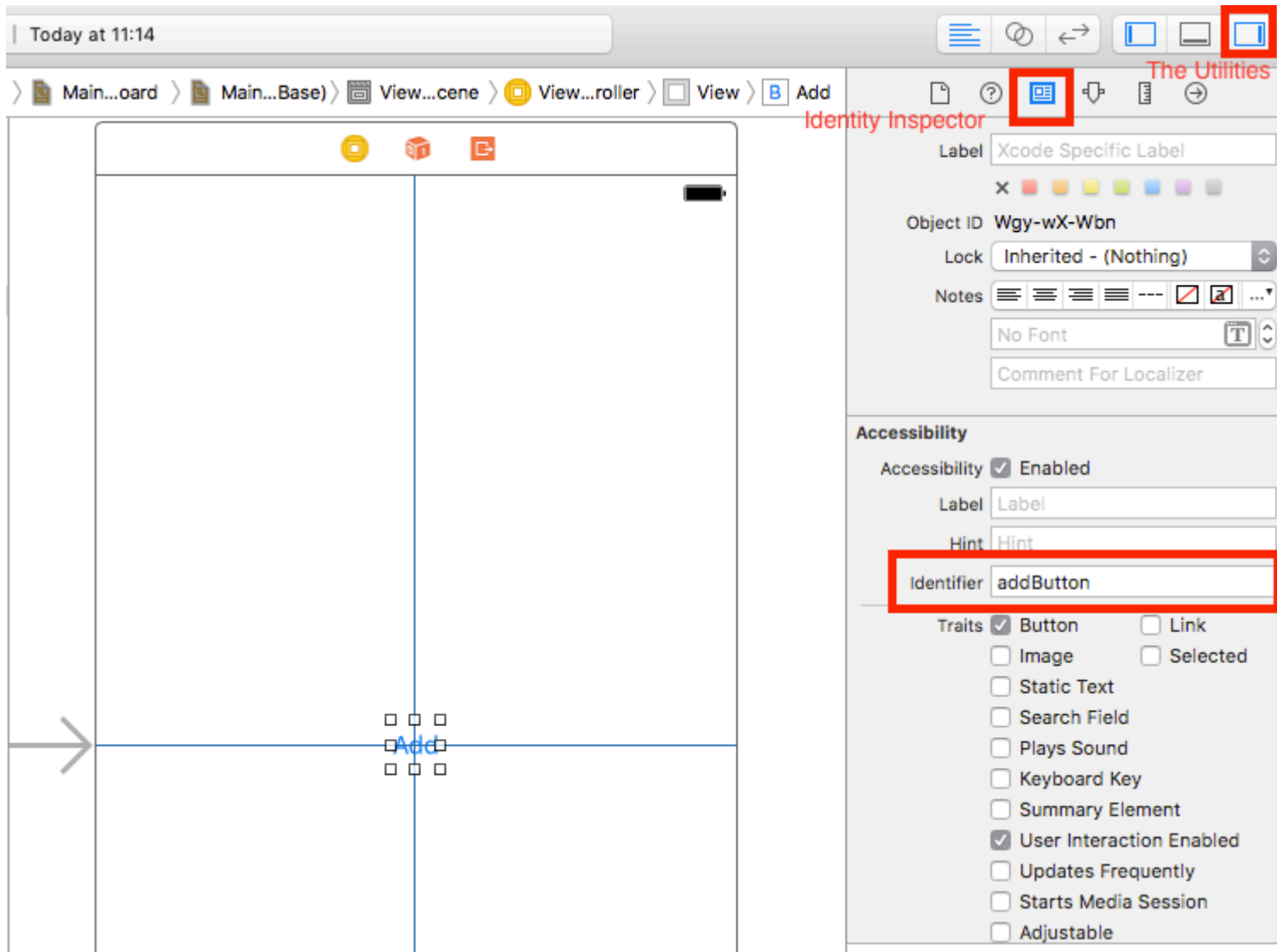
- Während das Projekt geöffnet ist, gehen Sie zu `File -> New -> Target`
- Suchen Sie ein `iOS UI Testing Bundle`



Eingabehilfe

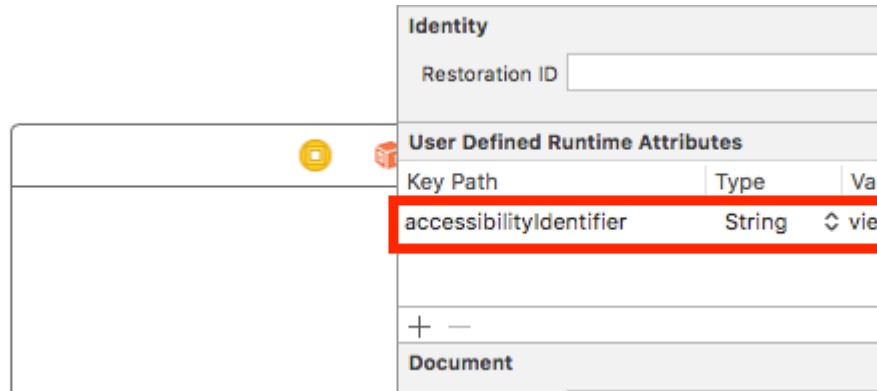
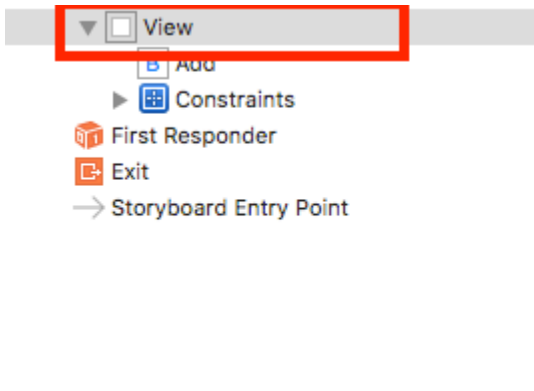
Wenn Eingabehilfen in Dienstprogrammen aktiviert ist

- storyboard .
- Erweitern Sie the Utilities
- Wählen Sie Identity Inspector
- Wählen Sie Ihr Element im Storyboard aus
- Neuen Accessibility-Bezeichner addButton (im Beispiel addButton)



Wenn Eingabehilfen in Dienstprogrammen deaktiviert ist

- storyboard .
- Erweitern Sie the Utilities
- Wählen Sie Identity Inspector
- Wählen Sie Ihr Element im Storyboard aus
- Attribut in User Defined Runtime Attributes
- Für Key Path - accessibilityIdentifier
- Für Type - `String
- Für Value - neue Zugänglichkeit Kennung für Ihr Element (in Beispiel view)



Einrichten in UITest-Datei

```
import XCTest

class StackOverFlowUITests: XCTestCase {

    private let app = XCUIApplication()

    //Views

    private var view: XCUIElement!

    //Buttons

    private var addButton: XCUIElement!

    override func setUp() {
        super.setUp()

        app.launch()

        //Views

        view = app.otherElements["view"]

        //Buttons

        addButton = app.buttons["addButton"]
    }

    func testMyApp() {

        addButton.tap()
        view.tap()
    }
}
```

Fügen Sie in [] Accessibility Identifier für Element hinzu.

UIView, UIImageView, UIScrollView

```
let imageView = app.images["imageView"]
let scrollView = app.scrollViews["scrollView"]
```

```
let view = app.otherElements["view"]
```

UILabel

```
let label = app.staticTexts["label"]
```

UIStackView

```
let stackView = app.otherElements["stackView"]
```

UITableView

```
let tableView = app.tables["tableView"]
```

UITableViewCell

```
let tableViewCell = tableView.cells["tableViewCell"]
```

UITableViewCell-Elemente

```
let tableViewCellButton = tableView.cells.element(boundBy: 0).buttons["button"]
```

UICollectionView

```
let collectionView = app.collectionViews["collectionView"]
```

UIButton, UIBarButtonItem

```
let button = app.buttons["button"]  
let barButtonItem = app.buttons["barButtonItem"]
```

UITextField

- normales UITextField

```
let textField = app.textFields["textField"]
```

- Kennwort UITextField

```
let passwordTextField = app.secureTextFields["passwordTextField"]
```

UITextView

```
let textView = app.textViews["textView"]
```

UISwitch

```
let switch = app.switches["switch"]
```

Alarme

```
let alert = app.alerts["About yourself"] // Title of presented alert
```

Deaktivieren Sie Animationen während des UI-Tests

In einem Test können Sie Animationen deaktivieren, indem Sie `setUp` hinzufügen:

```
app.launchEnvironment = ["animations": "0"]
```

Wo ist `app` Instanz von `XCUIApplication`.

Mittagessen und Anwendung während der Ausführung beenden

Mittagsanwendung zum Testen

```
override fun setUp() {
    super.setUp()

    let app = XCUIApplication()

    app.launch()
}
```

Anwendung beenden

```
func testStacOverFlowApp() {

    app.terminate()
}
```

Geräte drehen

Das Gerät kann durch Ändern der `orientation` in `XCUIDevice.shared().orientation`

```
XCUIDevice.shared().orientation = .landscapeLeft
```

```
XCUIDevice.shared().orientation = .portrait
```

UI-Test online lesen: <https://riptutorial.com/de/ios/topic/7526/ui-test>

Kapitel 196: UITextField

Einführung

UITextField ist Teil des UIKit-Frameworks und wird verwendet, um einen Bereich zum Erfassen von Texteingaben des Benutzers mithilfe der Bildschirmtastatur anzuzeigen

Syntax

- `UITextField.text`: `String` // Ermittelt oder setzt den Text, den das Feld anzeigt.
- `UITextField.attributedText`: `NSAttributedString` // Abrufen oder Festlegen des zugeordneten Felds.
- `UITextField.textColor`: `UIColor` // Ermittelt oder setzt die Farbe des Texts im Feld
- `UITextField.font`: `UIFont` // Ermittelt oder setzt die Schriftart des Texts im Feld
- `UITextField.textAlignment`: `NSTextAlignment` // Standardwert ist `NSLeftTextAlignment`
- `UITextField.borderStyle`: `UITextBorderStyle` // Der Standardwert ist `UITextBorderStyleNone`. Wenn `UITextBorderStyleRoundedRect` eingestellt ist, werden benutzerdefinierte Hintergrundbilder ignoriert.
- `UITextField.placeholder`: `String` // default ist nil. Zeichenfolge ist zu 70% grau gezeichnet
- `UITextField.attributedPlaceholder`: `NSAttributedString` // Ermittelt oder setzt den attributierten Platzhalter des Feldes
- `UITextField.clearsOnBeginEditing`: `Bool` // Standardeinstellung ist `NO`, wodurch der Cursor zur angeklickten Position bewegt wird. Wenn ja, wird der gesamte Text gelöscht
- `UITextField.adjustsFontSizeToFitWidth`: `Bool` // Standardwert ist `NO`. Bei `JA` wird der Text entlang der Grundlinie auf `minFontSize` verkleinert
- `UITextField.minimumFontSize`: `CGFloat` // Standard ist 0,0. tatsächliche min kann an etwas Lesbares geheftet werden. Wird verwendet, wenn `adjustsFontSizeToFitWidth` den Wert `YES` hat
- `UITextField.delegate`: `UITextFieldDelegate?` // Standard ist `Null`. schwache Referenz
- `UITextField.clearButtonMode`: `UITextFieldViewMode` // legt fest, wann die Schaltfläche "Löschen" angezeigt wird. Der Standardwert ist `UITextFieldViewModeNever`
- `UITextField.leftView`: `UIView?` // zB Lupe
- `UITextField.leftViewMode`: `UITextFieldViewMode` // legt fest, wann die linke Ansicht angezeigt wird. Der Standardwert ist `UITextFieldViewModeNever`
- `UITextField.rightView`: `UIView?` // zB Lesezeichen-Button
- `UITextField.rightViewMode`: `UITextFieldViewMode` // legt fest, wann die rechte Ansicht angezeigt wird. Der Standardwert ist `UITextFieldViewModeNever`
- `UITextField.inputView`: `UIView?` // Wird angezeigt, wenn das Objekt zum Ersthelfer wird. Bei der Einstellung `Null` wird die folgende Answerkette zurückgesetzt. Wenn diese Option beim ersten Antwort eingestellt ist, wird sie erst wirksam, wenn `reloadInputViews` aufgerufen wird.
- `UITextField.inputAccessoryView`: `UIView?`
- `UITextField.isSecureTextEntry`: `Bool` // zB Wenn das Feld eine vertrauliche Eingabe wie Kennwort oder Kartenummer enthält

Examples

Textfeld initialisieren

Schnell

```
let frame = CGRect(x: 0, y: 0, width: 100, height: 100)
let textField = UITextField(frame: frame)
```

Ziel c

```
CGRect *frame = CGRectMake(0, 0, 100, 100);
UITextField *textField = [[UITextField alloc] initWithFrame:frame];
```

Interface Builder

Sie können ein `UITextField` zu einem Storyboard hinzufügen, indem Sie es aus der `UITextField` ziehen.



Eingabe-Zubehöransicht (Symbolleiste)

Fügen Sie eine Zubehöransicht über der Tastatur hinzu. Diese Option wird häufig zum Hinzufügen von Schaltflächen für die nächsten / vorherigen Tasten oder für zusätzliche Schaltflächen wie Done / Submit verwendet (insbesondere für Tastaturtypen mit Nummern- / Telefon- / Dezimalauflagen, die keine integrierte Eingabetaste haben).

Schnell

```
let textField = UITextField() // initialized however

let toolbar = UIToolbar(frame: CGRect(x: 0, y: 0, width: view.frame.size.width, height: 0))

let flexibleSpace = UIBarButtonItem(barButtonItemSystemItem: .FlexibleSpace, target: nil, action: nil)

let doneButton = UIBarButtonItem(barButtonItemSystemItem: .Done, target: self, action: Selector("done"))
```



```
let items = [flexibleSpace, doneButton] // pushes done button to right side

toolbar.setItems(items, animated: false) // or toolbar.items = ...
toolbar.sizeToFit()

textField.inputAccessoryView = toolbar
```

Ziel c

```
UITextField *textField = [[UITextField alloc] init];

UIToolbar *toolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 0)];

UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil action:nil];
UIBarButtonItem *doneButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemDone target:self action:@selector(done)];
NSArray *items = @[
    flexibleSpace,
    doneButton
];

[toolbar setItems:items];
[toolbar sizeToFit];

textField.inputAccessoryView = toolbar;
```

Automatische Kapitalisierung

Schnell

```
textField.autocapitalizationType = .None
```

Ziel c

```
textField.autocapitalizationType = UITextAutocapitalizationTypeNone;
```

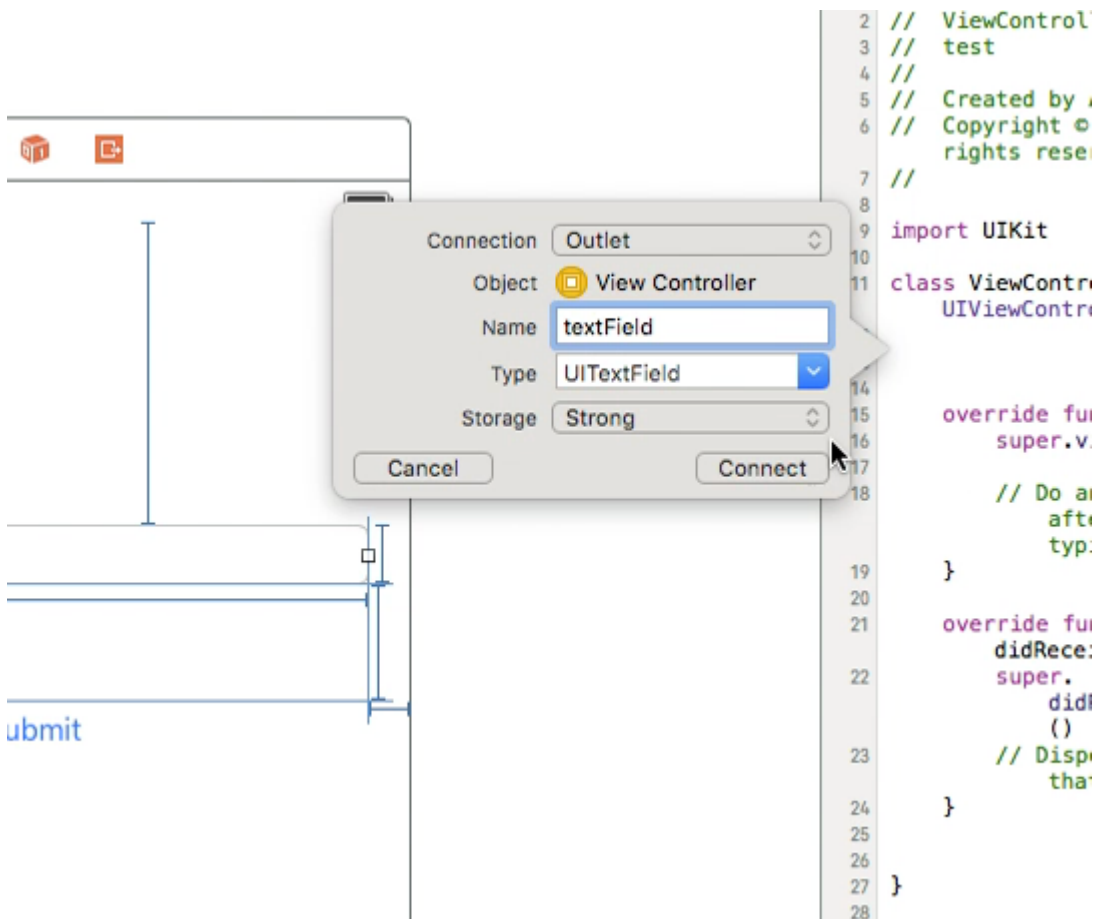
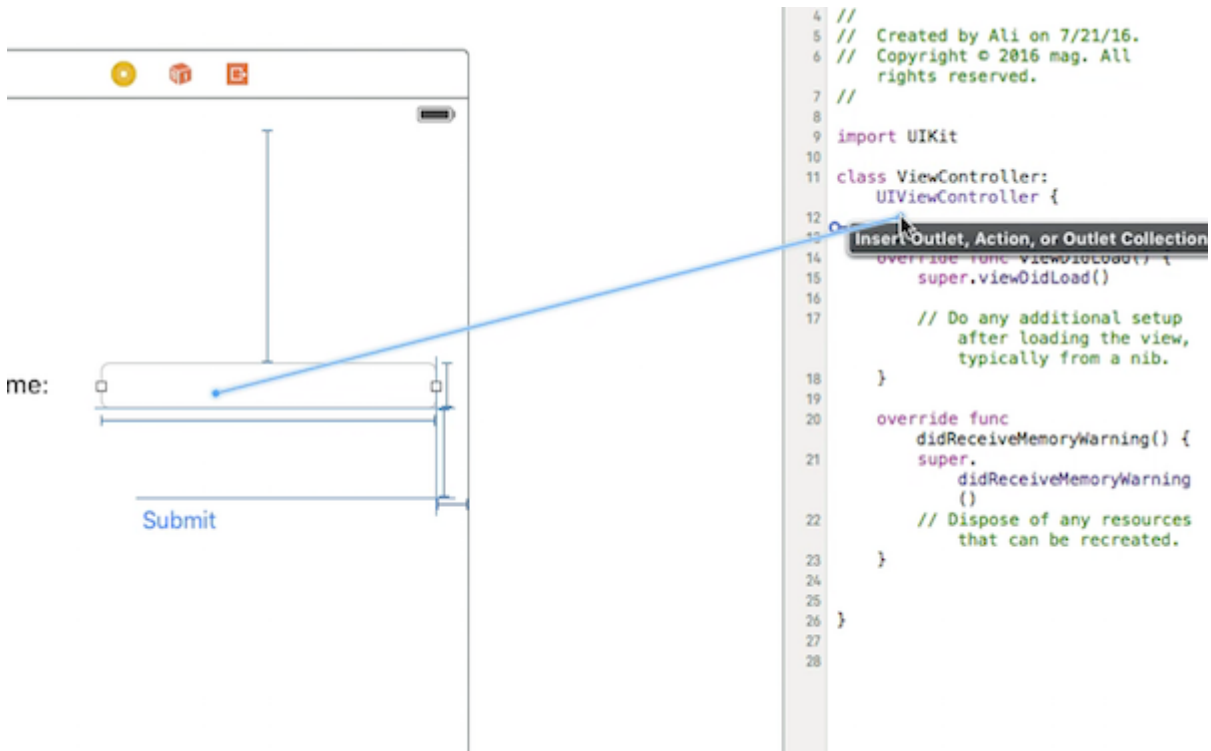
Alle Optionen:

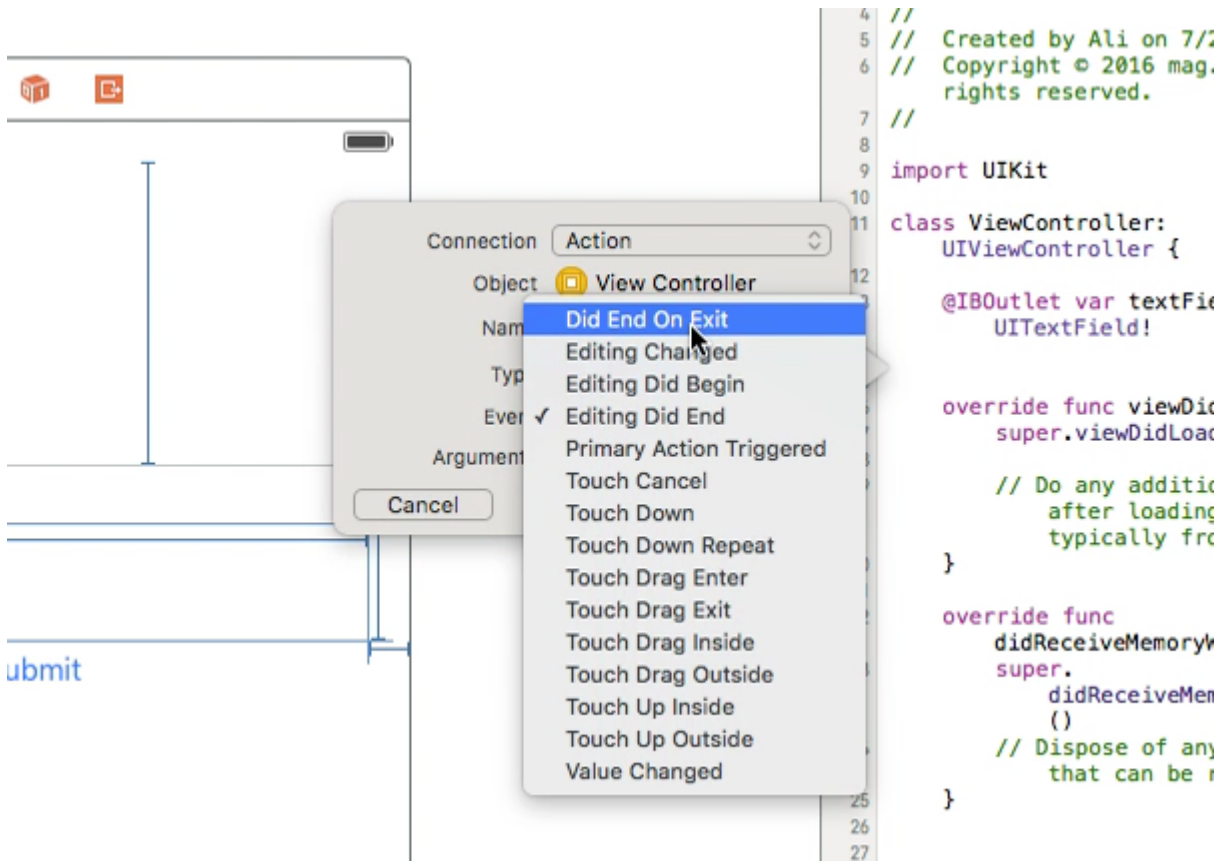
- `.None \ UITextAutocapitalizationTypeNone` : Autokapitalisieren Sie nichts
- `.Words \ UITextAutocapitalizationTypeWords` : AutoCapitalize jedes Wort
- `.Sentences \ UITextAutocapitalizationTypeSentences` : Autokapitalisieren Sie das erste Wort eines Satzes
- `.AllCharacters \ UITextAutocapitalizationTypeAllCharacters` : Alle Buchstaben `.AllCharacters` `UITextAutocapitalizationTypeAllCharacters` (dh `UITextAutocapitalizationTypeAllCharacters` sperren)

Tastatur schließen

Schnell

Strg + Ziehen Sie aus dem UI-Textfeld in MainStoryboard in die ViewController-Klasse, und erstellen Sie ein UITextField-Outlet





Wählen Sie anschließend das UITextField erneut aus und ziehen Sie die Strg-Taste in der ViewController-Klasse. Wählen Sie diesmal **Action** connection. Wählen Sie im Speicher **Did End On Exit** und klicken Sie auf connect.

Geben Sie in der soeben erstellten Aktion den Namen Ihres UITextField `.resignFirstResponder()`

```

@IBAction func textFieldResign(sender: AnyObject) {
    yourTextFieldName.resignFirstResponder()
}

```

Dadurch wird die Tastatur ausgeblendet, wenn Sie die Eingabetaste auf der Tastatur drücken.

Ein weiteres Beispiel für das Ausblenden der Tastatur, wenn die Eingabetaste gedrückt wird:

neben UIViewController fügen wir das UITextFieldDelegate Protokoll UIViewController

In der Funktion `self.yourTextFieldName.delegate = self` fügen wir `self.yourTextFieldName.delegate = self`

Und zum Schluss fügen wir das hinzu

```

func textFieldShouldReturn(textField: UITextField) -> Bool {
    yourTextFieldName.resignFirstResponder()
    return true
}

```

Der endgültige Code lautet:

```

class ViewController: UIViewController, UITextFieldDelegate {

@IBOutlet var textField: UITextField!

    func textFieldShouldReturn(textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

    override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
        view.endEditing(true)
        super.touchesBegan(touches, withEvent: event)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.textField.delegate = self
    }

}

```

Ziel c

```
[textField resignFirstResponder];
```

Ausrichtung einstellen

Schnell

```
textField.textAlignment = .Center
```

Ziel c

```
[textField setTextAlignment: NSTextAlignmentCenter];
```

In diesem Beispiel haben wir `NSTextAlignment` auf center gesetzt. Sie können auch einstellen `.Left`, `.Right`, `.Justified` und `.Natural`.

`.Natural` ist die Standardausrichtung für die aktuelle Lokalisierung. Das bedeutet für Sprachen von links nach rechts (z. B. Englisch) die Ausrichtung `.Left`; für Rechts-nach-Links-Sprachen ist es `.Right`.

KeyboardType

Um die Darstellung der Tastatur zu ändern, können die folgenden Typen für jede `UITextField` Eigenschaft einzeln `UITextField : keyboardType`

```
typedef NS_ENUM(NSInteger, UIKeyboardType) {
```

```

    UIKeyboardTypeDefault, // Default type for the current input method.
    UIKeyboardTypeASCIICapable, // Displays a keyboard which can enter ASCII
characters, non-ASCII keyboards remain active
    UIKeyboardTypeNumbersAndPunctuation, // Numbers and assorted punctuation.
    UIKeyboardTypeURL, // A type optimized for URL entry (shows . / .com
prominently).
    UIKeyboardTypeNumberPad, // A number pad (0-9). Suitable for PIN entry.
    UIKeyboardTypePhonePad, // A phone pad (1-9, *, 0, #, with letters under the
numbers).
    UIKeyboardTypeNamePhonePad, // A type optimized for entering a person's name or
phone number.
    UIKeyboardTypeEmailAddress, // A type optimized for multiple email address entry
(shows space @ . prominently).
    UIKeyboardTypeDecimalPad NS_ENUM_AVAILABLE_IOS(4_1), // A number pad with a decimal
point.
    UIKeyboardTypeTwitter NS_ENUM_AVAILABLE_IOS(5_0), // A type optimized for twitter
text entry (easy access to @ #)
    UIKeyboardTypeWebSearch NS_ENUM_AVAILABLE_IOS(7_0), // A default keyboard type with
URL-oriented addition (shows space . prominently).

    UIKeyboardTypeAlphabet = UIKeyboardTypeASCIICapable, // Deprecated
};

```

Verschieben des Bildlaufs, wenn UITextView zum Ersthelfer wird

Beachten Sie die Benachrichtigungen `UIKeyboardWillShowNotification` und `UIKeyboardWillHideNotification`. Aktualisieren Sie die `scrollView` Inhaltsinsets entsprechend der Tastaturhöhe. Blättern Sie dann zum fokussierten Steuerelement.

```

- (void)viewDidLoad
{
    [super viewDidLoad];

    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(keyboardWillShow:)
                                             name:UIKeyboardWillShowNotification
                                             object:self.view.window];

    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(keyboardWillHide:)
                                             name:UIKeyboardWillHideNotification
                                             object:self.view.window];
}

// Called when UIKeyboardWillShowNotification is sent
- (void)keyboardWillShow:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = [notification userInfo];

    CGRect keyboardFrameInWindow;
    [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardFrameInWindow];
}

```

```

// the keyboard frame is specified in window-level coordinates. this calculates the frame
as if it were a subview of our view, making it a sibling of the scroll view
CGRect keyboardFrameInView = [self.view convertRect:keyboardFrameInWindow fromView:nil];

CGRect scrollViewKeyboardIntersection = CGRectIntersection(_scrollView.frame,
keyboardFrameInView);
UIEdgeInsets newContentInsets = UIEdgeInsetsMake(0, 0,
scrollViewKeyboardIntersection.size.height, 0);

// this is an old animation method, but the only one that retains compaitibility between
parameters (duration, curve) and the values contained in the userInfo-Dictionary.
[UIView beginAnimations:nil context:NULL];
[UIView setAnimationDuration:[userInfo
objectForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
[UIView setAnimationCurve:[userInfo objectForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

_scrollView.contentInset = newContentInsets;
_scrollView.scrollIndicatorInsets = newContentInsets;

/*
 * Depending on visual layout, _focusedControl should either be the input field
(UITextField,..) or another element
 * that should be visible, e.g. a purchase button below an amount text field
 * it makes sense to set _focusedControl in delegates like -textFieldShouldBeginEditing:
if you have multiple input fields
 */
if (_focusedControl) {
    CGRect controlFrameInScrollView = [_scrollView convertRect:_focusedControl.bounds
fromView:_focusedControl]; // if the control is a deep in the hierarchy below the scroll view,
this will calculate the frame as if it were a direct subview
    CGRect controlFrameInScrollView = CGRectInset(controlFrameInScrollView, 0, -10); // replace
10 with any nice visual offset between control and keyboard or control and top of the scroll
view.

    CGFloat controlVisualOffsetToTopOfScrollview = controlFrameInScrollView.origin.y -
_scrollView.contentOffset.y;
    CGFloat controlVisualBottom = controlVisualOffsetToTopOfScrollview +
controlFrameInScrollView.size.height;

    // this is the visible part of the scroll view that is not hidden by the keyboard
    CGFloat scrollViewVisibleHeight = _scrollView.frame.size.height -
scrollViewKeyboardIntersection.size.height;

    if (controlVisualBottom > scrollViewVisibleHeight) { // check if the keyboard will
hide the control in question
        // scroll up until the control is in place
        CGPoint newContentOffset = _scrollView.contentOffset;
        newContentOffset.y += (controlVisualBottom - scrollViewVisibleHeight);

        // make sure we don't set an impossible offset caused by the "nice visual offset"
// if a control is at the bottom of the scroll view, it will end up just above the
keyboard to eliminate scrolling inconsistencies
        newContentOffset.y = MIN(newContentOffset.y, _scrollView.contentSize.height -
scrollViewVisibleHeight);

        [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
    } else if (controlFrameInScrollView.origin.y < _scrollView.contentOffset.y) {
        // if the control is not fully visible, make it so (useful if the user taps on a
partially visible input field

```

```

        CGPoint newContentOffset = _scrollView.contentOffset;
        newContentOffset.y = controlFrameInScrollView.origin.y;

        [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
    }
}

[UIView commitAnimations];
}

// Called when the UIKeyboardWillHideNotification is sent
- (void)keyboardWillHide:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = notification.userInfo;

    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:[userInfo
valueForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
    [UIView setAnimationCurve:[userInfo valueForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

    // undo all that keyboardWillShow-magic
    // the scroll view will adjust its contentOffset appropriately
    _scrollView.contentInset = UIEdgeInsetsZero;
    _scrollView.scrollIndicatorInsets = UIEdgeInsetsZero;

    [UIView commitAnimations];
}

```

Tastaturfokus erhalten und Tastatur ausblenden

Holen Sie sich den Fokus

Schnell

```
textField.becomeFirstResponder()
```

Ziel c

```
[textField becomeFirstResponder];
```

Zurücktreten

Schnell

```
textField.resignFirstResponder()
```

Ziel c

```
[textField resignFirstResponder];
```

Ersetzen Sie die Tastatur durch UIPickerView

In einigen Fällen möchten Sie Ihren Benutzern eine `UIPickerView` mit vordefiniertem Inhalt für ein `UITextField` anstelle einer Tastatur `UITextField`.

Erstellen Sie eine benutzerdefinierte UIPickerView

Zunächst benötigen Sie eine benutzerdefinierte Wrapper-Klasse für `UIPickerView`, die den Protokollen `UIPickerViewDataSource` und `UIPickerViewDelegate`.

```
class MyPickerView: UIPickerView, UIPickerViewDataSource, UIPickerViewDelegate
```

Sie müssen die folgenden Methoden für die `DataSource` und den `Stellvertreter` implementieren:

```
public func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    if data != nil {
        return data!.count
    } else {
        return 0
    }
}

public func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

public func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
    if data != nil {
        return data![row]
    } else {
        return ""
    }
}
```

Um die Daten zu verarbeiten, `MyPickerView` muss die Eigenschaften `data`, `selectedValue` und `textFieldBeingEdited`:

```
/**
 * The data for the `UIPickerViewDelegate`
 *
 * Always needs to be an array of `String`! The `UIPickerView` can ONLY display Strings
 */
public var data: [String]? {
    didSet {
```



```

        super.delegate = self
        super.dataSource = self
        self.reloadAllComponents()
    }
}

/**
 * Stores the UITextField that is being edited at the moment
 */
public var textFieldBeingEdited: UITextField?

/**
 * Get the selected Value of the picker
 */
public var selectedValue: String {
    get {
        if data != nil {
            return data![selectedRow(inComponent: 0)]
        } else {
            return ""
        }
    }
}
}

```

Bereiten Sie Ihren ViewController vor

Der `ViewController`, der Ihr `textField` enthält, muss eine Eigenschaft für Ihre benutzerdefinierte `UIPickerView`. (Vorausgesetzt, Sie haben bereits eine andere Eigenschaft oder ein `@IBOutlet` das Ihr `textField` enthält.)

```

/**
 * The picker view to present as keyboard
 */
var picker: MyPickerView?

```

In Ihrem `viewDidLoad()` müssen Sie den `picker` initialisieren und ein wenig konfigurieren:

```

picker = MyPickerView()
picker?.autoresizingMask = [.flexibleHeight, .flexibleWidth]
picker?.backgroundColor = UIColor.white()

picker?.data = ["One", "Two", "Three", "Four", "Five"] //The data shown in the picker

```

Jetzt können Sie den `MyPicker` als `inputView` Ihres `UITextField`:

```

textField.inputView = picker

```

Entnahme der Picker-Tastatur

Jetzt haben Sie die Tastatur durch eine `UIPickerView`, die Möglichkeit besteht jedoch nicht. Dies kann mit einer benutzerdefinierten `.inputAccessoryView`:

Fügen Sie den Property `pickerAccessoryView` Ihrem `ViewController`.

```
/**
 * A toolbar to add to the keyboard when the `picker` is presented.
 */
var pickerAccessory: UIToolbar?
```

In `viewDidLoad()` müssen Sie eine `UIToolbar` für `inputAccessoryView` :

```
pickerAccessory = UIToolbar()
pickerAccessory?.autoresizingMask = .flexibleHeight

//this customization is optional
pickerAccessory?.barStyle = .default
pickerAccessory?.barTintColor = UIColor.red()
pickerAccessory?.backgroundColor = UIColor.red()
pickerAccessory?.isTranslucent = false
```

Sie sollten den Rahmen Ihrer Symbolleiste festlegen. Um in das Design von iOS zu passen, wird empfohlen, eine Höhe von `44.0` :

```
var frame = pickerAccessory?.frame
frame?.size.height = 44.0
pickerAccessory?.frame = frame!
```

Für eine gute Benutzererfahrung sollten Sie zwei Tasten hinzufügen ("Done" und "Cancel"), aber es funktioniert auch nur mit einer, die die Tastatur verwirft.

```
let cancelButton = UIBarButtonItem(barButtonItemSystemItem: .cancel, target: self, action:
#selector (ViewController.cancelBtnClicked(_:)))
cancelButton.tintColor = UIColor.white()
let flexSpace = UIBarButtonItem(barButtonItemSystemItem: .flexibleSpace, target: nil, action: nil)
//a flexible space between the two buttons
let doneButton = UIBarButtonItem(barButtonItemSystemItem: .done, target: self, action:
#selector (ViewController.doneBtnClicked(_:)))
doneButton.tintColor = UIColor.white()

//Add the items to the toolbar
pickerAccessory?.items = [cancelButton, flexSpace, doneButton]
```

Jetzt können Sie die Symbolleiste als `inputAccessoryView`

```
textField.inputAccessoryView = pickerAccessory
```

Bevor Sie Ihr Projekt erstellen können, müssen Sie die Methoden implementieren. Die Schaltflächen rufen auf:

```
/**
 * Called when the cancel button of the `pickerAccessory` was clicked. Dismisses the picker
 */
func cancelBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
}

/**
 * Called when the done button of the `pickerAccessory` was clicked. Dismisses the picker and
```

```
puts the selected value into the textField
*/
func doneBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
    textField.text = picker?.selectedValue
}
```

Führen Sie Ihr Projekt aus, tippen Sie auf das `textField` und statt der Tastatur sollte ein Auswahlfenster wie `textField`:

Cancel

Done

One

Two

Three

Four

Wählen Sie einen Wert programmgesteuert aus (optional)

Wenn Sie nicht möchten, dass die erste Zeile automatisch ausgewählt wird, können Sie die ausgewählte Zeile wie in `UIPickerView`:

```
picker?.selectRow(3, inComponent: 0, animated: false) //Will select the row at index 3
```

Tastatur schließen, wenn der Benutzer die Rückkehrtaste drückt

Richten Sie Ihren View-Controller ein, um die Bearbeitung von Text für das Textfeld zu verwalten.

```
class MyViewController: UITextFieldDelegate {

    override viewDidLoad() {
        super.viewDidLoad()

        textField.delegate = self
    }

}
```

`textFieldShouldReturn` wird jedes Mal aufgerufen, wenn die Return-Taste auf der Tastatur gedrückt wird.

Schnell:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    textField.resignFirstResponder()
    return true;
}
```

Ziel c:

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    [textField resignFirstResponder];
    return true;
}
```

Cursorposition abrufen und einstellen

Nützliche Informationen

Ganz am Anfang des Textfeldes:

```
let startPosition: UITextPosition = textField.beginningOfDocument
```

Ganz am Ende des Textfeldes:

```
let endPosition: UITextPosition = textField.endOfDocument
```

Der aktuell ausgewählte Bereich:

```
let selectedRange: UITextRange? = textField.selectedTextRange
```

Cursorposition abrufen

```
if let selectedRange = textField.selectedTextRange {
    let cursorPosition = textField.offsetFromPosition(textField.beginningOfDocument,
    toPosition: selectedRange.start)
    print("\(cursorPosition)")
}
```

Cursorposition einstellen

Um die Position festzulegen, setzen alle diese Methoden tatsächlich einen Bereich mit den gleichen Start- und Endwerten.

Zu Beginn

```
let newPosition = textField.beginningOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Bis zum Ende

```
let newPosition = textField.endOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

An eine Position links von der aktuellen Cursorposition

```
// only if there is a currently selected range
if let selectedRange = textField.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textField.positionFromPosition(selectedRange.start, inDirection:
UITextLayoutDirection.Left, offset: 1) {

        // set the new position
        textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

Zu einer beliebigen Position

Beginnen Sie am Anfang und verschieben Sie 5 Zeichen nach rechts.

```
let arbitraryValue: Int = 5
if let newPosition = textField.positionFromPosition(textField.beginningOfDocument,
inDirection: UITextLayoutDirection.Right, offset: arbitraryValue) {

    textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

verbunden

Markieren Sie den gesamten Text

```
textField.selectedTextRange = textField.textRangeFromPosition(textField.beginningOfDocument,
toPosition: textField.endOfDocument)
```

Wählen Sie einen Textbereich aus

```
// Range: 3 to 7
let startPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 3)
let endPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 7)

if startPosition != nil && endPosition != nil {
    textField.selectedTextRange = textField.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```

Fügen Sie den Text an der aktuellen Cursorposition ein

```
textField.insertText("Hello")
```

Anmerkungen

- Dieses Beispiel stammt ursprünglich aus [dieser Stack Overflow-Antwort](#) .
- Diese Antwort verwendet ein Textfeld, jedoch gelten dieselben Konzepte für `UITextView` .
- Verwenden Sie `textField.becomeFirstResponder()` , um das Textfeld zu fokussieren und die Tastatur `textField.becomeFirstResponder()` .
- In [dieser Antwort](#) erfahren Sie, wie Sie den Text in einem bestimmten Bereich erhalten.

verbunden

- [So erstellen Sie einen Bereich in Swift](#) (beschäftigt sich indirekt mit der Frage, warum wir `selectedTextRange` hier anstelle von `selectedRange`)

Blinkende Einfügemarke ausblenden

Um die blinkende Einfügemarke auszublenen, müssen Sie die `caretRectForPosition` eines `UITextField` überschreiben und `CGRectZero` zurückgeben.

Swift 2,3 <

```
public override func caretRectForPosition(position: UITextPosition) -> CGRect {
    return CGRectZero
}
```

Swift 3

```
override func caretRect(for position: UITextPosition) -> CGRect {
    return CGRect.zero
}
```

```
}
```

Ziel c

```
- (CGRect) caretRectForPosition:(UITextPosition*) position{  
    return CGRectZero;  
}
```

Ändern Sie die Farbe und die Schriftart des Platzhalters

Wir können den Stil des Platzhalters ändern, indem Sie `attributedString` (einen `NSAttributedString`) setzen.

```
var placeholderAttributes = [String: AnyObject]()  
placeholderAttributes[NSForegroundColorAttributeName] = color  
placeholderAttributes[NSFontAttributeName] = font  
  
if let placeholder = textField.placeholder {  
    let newAttributedString = NSAttributedString(string: placeholder, attributes:  
        placeholderAttributes)  
    textField.attributedString = newAttributedString  
}
```

In diesem Beispiel ändern wir nur `color` und `font`. Sie können andere Eigenschaften wie Unterstreichungs- oder Durchstreichungsstil ändern. `NSAttributedString` zu den Eigenschaften, die geändert werden können, finden Sie unter `NSAttributedString`.

Erstellen Sie ein UITextField

Initialisieren Sie das `UITextField` mit einem `CGRect` als Frame:

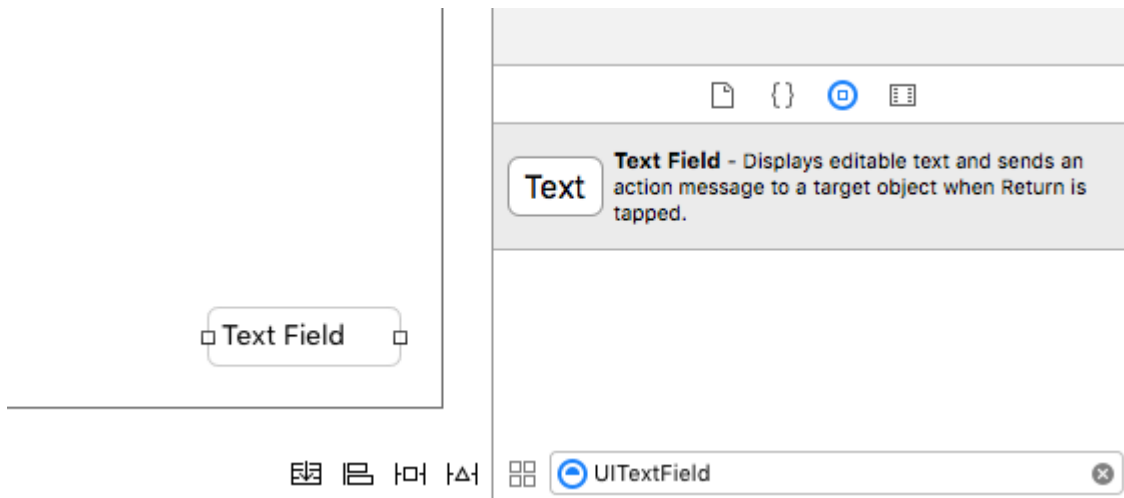
Schnell

```
let textField = UITextField(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

Ziel c

```
UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

Sie können auch ein `UITextField` im Interface Builder erstellen:



UITextField online lesen: <https://riptutorial.com/de/ios/topic/1630/uitextfield>

Kapitel 197: UITextField-Delegat

Examples

UITextField - Beschränkt das Textfeld auf bestimmte Zeichen

Wenn Sie eine Überprüfung der Benutzereingabe Ihres Textfelds durchführen möchten, verwenden Sie den folgenden Code-Snippet:

```
// MARK: - UITextFieldDelegate

let allowedCharacters =
CharacterSet(charactersIn:"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz").inverted

func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange,
replacementString string: String) -> Bool {

    let components = string.components(separatedBy: allowedCharacters)
    let filtered = components.joined(separator: "")

    if string == filtered {

        return true

    } else {

        return false

    }
}
```

Ziel c

```
#define ACCEPTABLE_CHARACTERS @"0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

- (BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange) range
replacementString:(NSString *)string
{
    NSCharacterSet *cs = [[NSCharacterSet
characterSetWithCharactersInString:ACCEPTABLE_CHARACTERS] invertedSet];

    NSString *filtered = [[string componentsSeparatedByCharactersInSet:cs]
componentsJoinedByString:@""];

    return [string isEqualToString:filtered];
}
```

Darüber hinaus können Sie die von Apple bereitgestellten Zeichensätze zur Validierung verwenden:

Besuchen Sie <https://developer.apple.com/reference/foundation/nscharacterset>

```
let allowedCharacters = CharacterSet.alphanumerics.inverted
let allowedCharacters = CharacterSet.capitalizedLetters.inverted
```

Nächstes Tag suchen und Tastatur verwalten

Das Textfeld ruft verschiedene Delegatmethoden auf (nur wenn Delegaten gesetzt sind). Eine von Textfeld aufgerufene Delegatmethode ist * - **(BOOL) textFieldShouldReturn: (UITextField *) textField**

Diese Methode wird aufgerufen, wenn Benutzer auf die Zurück-Schaltfläche tippen. Mit dieser Methode können Sie jedes benutzerdefinierte Verhalten implementieren.

Zum Beispiel,

Im folgenden Beispiel wird der nächste Responder anhand des Tags ermittelt und die Tastatur verwaltet. Hier ist 20 die Konstante, As-Tag, das dem Textfeld zugewiesen ist, ist wie folgt: 50,70,90 usw.

Beim Suchen eines neuen Textfeldobjekts als Responder wird das aktuelle Textfeld als neuer Responder erstellt und die Tastatur entsprechend geöffnet.

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    NSInteger nextTag = textField.tag+20;
    // Try to find next responder
    UIResponder *nextResponder = [textField.superview viewWithTag:nextTag];
    if (nextResponder)
    {
        // Found next responder, so set it.
        [nextResponder becomeFirstResponder];
    }
    else
    {
        // Not found, so remove keyboard.
        [textField resignFirstResponder];
    }
    return YES;
}
```

Aktionen, wenn ein Benutzer die Interaktion mit einem Textfeld gestartet / beendet hat

Für Swift 3.1:

Im ersten Beispiel kann man sehen, wie Sie den Benutzer abhalten, der während des Schreibens mit einem Textfeld interagiert. In ähnlicher Weise gibt es in [UITextFieldDelegate](#) Methoden, die [aufgerufen](#) werden, wenn ein Benutzer seine Interaktion mit einem TextField gestartet und beendet hat.

Um auf diese Methoden zugreifen zu können, müssen Sie das [UITextFieldDelegate](#)- Protokoll [einhalten](#) und für jedes Textfeld, über das Sie benachrichtigt werden möchten, die übergeordnete

Klasse als Delegat zuweisen:

```
class SomeClass: UITextFieldDelegate {  
  
    @IBOutlet var textField: UITextField!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        textField.delegate = self  
    }  
  
}
```

Jetzt können Sie alle UITextFieldDelegate-Methoden implementieren.

Um benachrichtigt zu werden, wenn ein Benutzer mit der Bearbeitung eines Textfelds begonnen hat, können Sie die `textFieldDidBeginEditing (_:)`-Methode wie folgt implementieren:

```
func textFieldDidBeginEditing(_ textField: UITextField) {  
    // now you can perform some action  
    // if you have multiple textfields in a class,  
    // you can compare them here to handle each one separately  
    if textField == emailTextField {  
        // e.g. validate email  
    }  
    else if textField == passwordTextField {  
        // e.g. validate password  
    }  
}
```

Wenn Sie benachrichtigt werden, wenn ein Benutzer die Interaktion mit einem Textfeld beendet hat, können Sie die `textFieldDidEndEditing (_:)`-Methode wie folgt verwenden :

```
func textFieldDidEndEditing(_ textField: UITextField) {  
    // now you can perform some action  
    // if you have multiple textfields in a class,  
    // you can compare them here to handle each one separately  
    if textField == emailTextField {  
        // e.g. validate email  
    }  
    else if textField == passwordTextField {  
        // e.g. validate password  
    }  
}
```

Wenn Sie steuern möchten, ob ein TextField mit der Bearbeitung beginnen / beenden soll, können die Methoden `textFieldShouldBeginEditing (_:)` und `textFieldShouldEndEditing (_:)` verwendet werden, indem Sie entsprechend Ihrer erforderlichen Logik true / false zurückgeben.

UITextField-Delegat online lesen: <https://riptutorial.com/de/ios/topic/7185/uitextfield-delegat>

Kapitel 198: UITextView

Examples

Text ändern

Schnell

```
textView.text = "Hello, world!"
```

Ziel c:

```
textView.text = @"Hello, world!";
```

Attributierten Text festlegen

```
// Modify some of the attributes of the attributed string.
let attributedText = NSMutableAttributedString(attributedString: textView.attributedText!)

// Use NSString so the result of rangeOfString is an NSRange.
let text = textView.text! as NSString

// Find the range of each element to modify.
let tintedRange = text.range(of: NSLocalizedString("tinted", comment: ""))
let highlightedRange = text.range(of: NSLocalizedString("highlighted", comment: ""))

// Add tint.
attributedText.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue, range:
tintedRange)

// Add highlight.
attributedText.addAttribute(NSBackgroundColorAttributeName, value: UIColor.yellow, range:
highlightedRange)

textView.attributedText = attributedText
```

Textausrichtung ändern

Schnell

```
textView.textAlignment = .left
```

Ziel c

```
textView.textAlignment = NSTextAlignmentLeft;
```

UITextViewDelegate-Methoden

Antworten auf das Bearbeiten von Benachrichtigungen

- `textViewShouldBeginEditing(_:)`
- `textViewDidBeginEditing(_:)`
- `textViewShouldEndEditing(_:)`
- `textViewDidEndEditing(_:)`

Antworten auf Textänderungen

- `textView(_:shouldChangeTextIn:replacementText:)`
- `textViewDidChange(_:)`

Auf URL antworten

- `textView(_: UITextView, shouldInteractWithURL: NSURL, inRange: NSRange) -> Bool`

Schriftart ändern

Schnell

```
//System Font
textView.font = UIFont.systemFont(ofSize: 12)

//Font of your choosing
textView.font = UIFont(name: "Font Name", size: 12)
```

Ziel c

```
//System Font
textView.font = [UIFont systemFontOfSize:12];

//Font of your choosing
textView.font = [UIFont fontWithName:@"Font Name" size:12];
```

Textfarbe ändern

Schnell

```
textView.textColor = UIColor.red
```

Ziel c

```
textView.textColor = [UIColor redColor];
```

UITextView mit HTML-Text

```
NSString *htmlString = @"<p> This is an <b>HTML</b> text</p>";
NSAttributedString *attributedString = [[NSMutableAttributedString alloc]
                                       initWithData: [htmlString
                                       dataUsingEncoding:NSUTF8StringEncoding]
                                       options: @{
```

```
NSDocumentTypeDocumentAttribute: NSHTMLTextDocumentType }
                                documentAttributes: nil
                                error: nil
                                ];
    _yourTextView.attributedString = attributedString;
    // If you want to modify the font
    field.font = [UIFont fontWithName:@"Raleway-Regular" size:15];
```

Automatische Erkennung von Links, Adressen, Datumsangaben und mehr

`UITextView` unterstützt die automatische Erkennung einer Vielzahl von Daten. Zu den Daten, die derzeit automatisch erkannt werden können, gehören:

```
enum {
    UIDataDetectorTypePhoneNumber = 1 << 0,
    UIDataDetectorTypeLink       = 1 << 1,
    UIDataDetectorTypeAddress    = 1 << 2,
    UIDataDetectorTypeCalendarEvent = 1 << 3,
    UIDataDetectorTypeNone       = 0,
    UIDataDetectorTypeAll        = NSUIntegerMax
};
```

Aktivieren der automatischen Erkennung

```
// you may add as many as you like by using the `|` operator between options
textView.dataDetectorTypes = (UIDataDetectorTypeLink | UIDataDetectorTypePhoneNumber);
```

Wenn aktiviert, wird der Text in der `UITextView` als Hyperlink `UITextView`

Anklickbare Daten

Damit der Link angeklickt werden kann (was je nach Datentyp zu unterschiedlichen Aktionen führt), müssen Sie sicherstellen, dass die `UITextView` auswählbar aber nicht bearbeitbar ist und die Benutzerinteraktion aktiviert ist

```
textView.editable = NO;
textView.selectable = YES;
textView.userInteractionEnabled = YES; // YES by default
```

Überprüfen Sie, ob leer oder null ist

Schnell

```
if let text = self.textView.text where !text.isEmpty {
    // Do stuff for text
} else {
    // Do stuff for nil text or empty string
}
```

Ziel c

```
if (self.textView.text.length > 0){
    // Do stuff for text
} else {
    // Do stuff for nil text or empty string
}
```

Cursorposition abrufen und einstellen

Nützliche Informationen

Ganz am Anfang des Textfeldes:

```
let startPosition: UITextPosition = textView.beginningOfDocument
```

Ganz am Ende des Textfeldes:

```
let endPosition: UITextPosition = textView.endOfDocument
```

Der aktuell ausgewählte Bereich:

```
let selectedRange: UITextRange? = textView.selectedTextRange
```

Cursorposition abrufen

```
if let selectedRange = textView.selectedTextRange {
    let cursorPosition = textView.offsetFromPosition(textView.beginningOfDocument, toPosition:
selectedRange.start)
    print("\(cursorPosition)")
}
```

Cursorposition einstellen

Um die Position festzulegen, setzen alle diese Methoden tatsächlich einen Bereich mit den gleichen Start- und Endwerten.

Zu Beginn

```
let newPosition = textView.beginningOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Bis zum Ende

```
let newPosition = textView.endOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

An eine Position links von der aktuellen Cursorposition

```
// only if there is a currently selected range
if let selectedRange = textView.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textView.positionFromPosition(selectedRange.start, inDirection:
UITextLayoutDirection.Left, offset: 1) {

        // set the new position
        textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

Zu einer beliebigen Position

Beginnen Sie am Anfang und verschieben Sie 5 Zeichen nach rechts.

```
let arbitraryValue: Int = 5
if let newPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: arbitraryValue) {

    textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

verbunden

Markieren Sie den gesamten Text

```
textView.selectedTextRange = textView.textRangeFromPosition(textView.beginningOfDocument,
toPosition: textView.endOfDocument)
```

Wählen Sie einen Textbereich aus

```
// Range: 3 to 7
let startPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 3)
let endPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 7)

if startPosition != nil && endPosition != nil {
    textView.selectedTextRange = textView.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```


Fügen Sie den Text an der aktuellen Cursorposition ein

```
textView.insertText("Hello")
```

Anmerkungen

- Dieses Beispiel stammt ursprünglich aus einer Anpassung [dieser Stack Overflow-Antwort](#) .
- Diese Antwort verwendet ein Textfeld, jedoch gelten dieselben Konzepte für `UITextView` .
- Verwenden Sie `textView.becomeFirstResponder()` , um den Fokus auf das Textfeld zu legen und die Tastatur erscheinen zu lassen.
- In [dieser Antwort](#) erfahren Sie, wie Sie den Text in einem bestimmten Bereich erhalten.

verbunden

- [So erstellen Sie einen Bereich in Swift](#) (beschäftigt sich indirekt mit der Frage, warum wir `selectedTextRange` hier anstelle von `selectedRange`)

Entfernen Sie zusätzliche Füllungen, um einen genau gemessenen Text zu erhalten.

`UITextView` hat standardmäßig zusätzliche Auffüllungen. Manchmal ist es ärgerlich, vor allem, wenn Sie Text ohne Ansichtsinstanz ausmessen und ihn genau in einem Bereich platzieren möchten.

So entfernen Sie solche Polsterungen.

```
messageTextView.textContainerInset = UIEdgeInsetsZero  
messageTextView.textContainer.lineFragmentPadding = 0
```

Jetzt können Sie die Textgröße mit `NSAttributedString.boundingRectWithSize(...)` messen und die Größe einer `UITextView` , um sie an den Text `UITextView` .

```
let budget = getSomeCGSizeBudget()  
let text = getSomeAttributedString()  
let textSize = text.boundingRectWithSize(budget, options: [.UsesLineFragmentOrigin,  
.UsesFontLeading], context: nil).size  
messageTextView.frame.size = textSize // Just fits.
```

[UITextView online lesen: https://riptutorial.com/de/ios/topic/1043/uitextView](https://riptutorial.com/de/ios/topic/1043/uitextView)

Kapitel 199: UIView

Syntax

1. // Ziel c
2. [UIView neu] // Ein Ansichtsobjekt zugewiesen bekommen und initialisiert
3. [[UIView-Zuordnung] initWithFrame: (Pass CGRect)] // Die Ansicht zugewiesen und mit einem Frame initialisiert
4. [[UIView-Zuordnung] init] // Ein Ansichtsobjekt zugewiesen und initialisiert erhalten
5. // schnell
6. UIView () // Erstellt eine UIView-Instanz mit CGRect.zero-Frame
7. UIView (frame: CGRect) // Erstellt eine UIView-Instanz, die den Frame angibt
8. UIView.addSubview (UIView) // Eine weitere UIView-Instanz als Untersicht hinzufügen
9. UIView.hidden // Ermitteln Sie die Sichtbarkeit der Ansicht
10. UIView.alpha // Ermittelt oder setzt die Deckkraft der Ansicht
11. UIView.setNeedsLayout () // Erzwingt, dass die Ansicht ihr Layout aktualisiert

Bemerkungen

Die **UIView**- Klasse definiert einen rechteckigen Bereich auf dem Bildschirm und die Schnittstellen zum Verwalten des Inhalts in diesem Bereich. Zur Laufzeit übernimmt ein Ansichtsobjekt das Rendern von Inhalten in seinem Bereich sowie alle Interaktionen mit diesem Inhalt.

Examples

Erstellen Sie eine UIView

Ziel c

```
CGRect myFrame = CGRectMake(0, 0, 320, 35)
UIView *view = [[UIView alloc] initWithFrame:myFrame];

//Alternative way of defining the frame
UIView *view = [[UIView alloc] init];
CGRect myFrame = view.frame;
myFrame.size.width = 320;
myFrame.size.height = 35;
myFrame.origin.x = 0;
```

```
myFrame.origin.y = 0;
view.frame = myFrame;
```

Schnell

```
let myFrame = CGRect(x: 0, y: 0, width: 320, height: 35)
let view = UIView(frame: myFrame)
```

Machen Sie die Ansicht gerundet

Um eine abgerundete `UIView`, geben `cornerRadius` für den `layer` der Ansicht einen `cornerRadius` an.

Dies gilt auch für jede Klasse, die von `UIView` erbt, z. B. `UIImageView`.

Programmatisch

SWIFT-Code

```
someImageView.layoutIfNeeded()
someImageView.clipsToBounds = true
someImageView.layer.cornerRadius = 10
```

Ziel-C-Code

```
[someImageView layoutIfNeeded];
someImageView.clipsToBounds = YES;
someImageView.layer.cornerRadius = 10;
```

Beispiel

```
//Swift code
topImageView.layoutIfNeeded()
bottomImageView.layoutIfNeeded()
topImageView.clipsToBounds = true
topImageView.layer.cornerRadius = 10
bottomImageView.clipsToBounds = true
bottomImageView.layer.cornerRadius = bottomImageView.frame.width / 2

//Objective-C code
[topImageView layoutIfNeeded]
[bottomImageView layoutIfNeeded];
topImageView.clipsToBounds = YES;
topImageView.layer.cornerRadius = 10;
bottomImageView.clipsToBounds = YES;
bottomImageView.cornerRadius = CGRectGetGetWidth(bottomImageView.frame) / 2;
```

Das Ergebnis zeigt den abgerundeten Ansichtseffekt mit dem angegebenen Eckenradius:



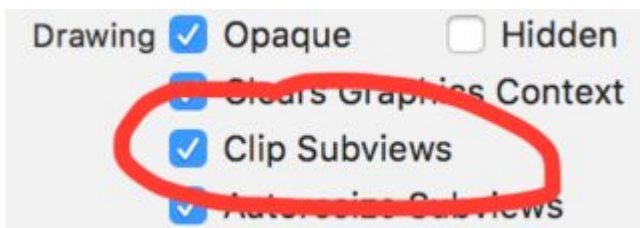
Hinweis

Dazu müssen Sie das QuartzCore-Framework einbeziehen.

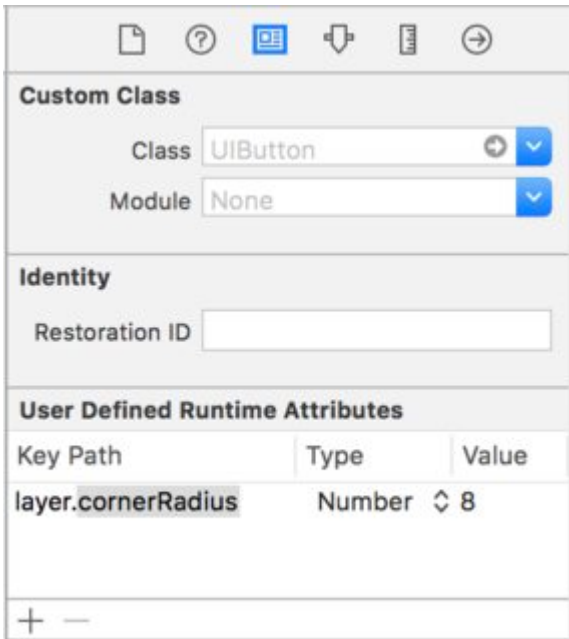
```
#import <QuartzCore/QuartzCore.h>
```

Storyboard-Konfiguration

Ein abgerundeter Ansichtseffekt kann auch `non-programmatically` erzielt werden, indem die entsprechenden Eigenschaften im **Storyboard festgelegt werden** .



Da `layer` Eigenschaften im Storyboard nicht `cornerRadius` werden, müssen Sie das `cornerRadius` Attribut im Abschnitt Benutzerdefinierte Laufzeitattribute ändern.



Schnelle Erweiterung

Sie können diese praktische Erweiterung verwenden, um eine abgerundete Ansicht anzuwenden, solange sie die gleiche Breite und Höhe hat.

```
extension UIView {
    @discardableResult
    public func setAsCircle() -> Self {
        self.clipsToBounds = true
        let frameSize = self.frame.size
        self.layer.cornerRadius = min(frameSize.width, frameSize.height) / 2.0
        return self
    }
}
```

Um es zu benutzen:

```
yourView.setAsCircle()
```

Momentaufnahme machen

Sie können eine Momentaufnahme aus einer `UIView` wie `UIView` :

Schnell

```
let snapshot = view.snapshotView(afterScreenUpdates: true)
```

Ziel c

```
UIView *snapshot = [view snapshotViewAfterScreenUpdates: YES];
```

Verwendung von IBInspectable und IBDesignable

Ein (oder zwei) der coolsten neuen Features in den letzten Xcode - Versionen sind die `IBInspectable` Eigenschaften und `IBDesignable` `UIView` s. Diese haben nichts mit der Funktionalität Ihrer Anwendung zu tun, sondern beeinflussen die Entwicklererfahrung in Xcode. Ziel ist es, benutzerdefinierte Ansichten in Ihrer iOS-Anwendung visuell zu prüfen, ohne sie auszuführen. `CustomView` Sie also an, Sie haben eine benutzerdefinierte Ansicht namens `CustomView` , die von `UIView` erbt. In dieser benutzerdefinierten Ansicht wird eine Textzeichenfolge mit einer bestimmten Farbe angezeigt. Sie können auch festlegen, dass kein Text angezeigt wird. Wir brauchen drei Eigenschaften:

```
var textColor: UIColor = UIColor.blackColor()
var text: String?
var showText: Bool = true
```

Wir können dann die `drawRect` Funktion in der Klasse überschreiben:

```
if showText {
    if let text = text {
        let s = NSString(string: text)
        s.drawInRect(rect,
                    withAttributes: [
                        NSForegroundColorAttributeName: textColor,
                        NSFontAttributeName: UIFont(name: "Helvetica Neue", size: 18)!
                    ])
    }
}
```

Unter der Annahme, dass die `text` ist, wird beim Ausführen der Anwendung eine Zeichenfolge in der oberen linken Ecke der Ansicht gezeichnet. Das Problem ist, dass wir nicht wissen, wie es aussieht, ohne die Anwendung auszuführen. Hier setzen `IBInspectable` und `IBDesignable` an. Mit `IBInspectable` können Sie Eigenschaftswerte der Ansicht in Xcode visuell festlegen, genau wie bei den integrierten Steuerelementen. `IBDesignable` zeigt uns eine visuelle Vorschau im Storyboard. So sollte die Klasse aussehen:

```
@IBDesignable
class CustomView: UIView {
    @IBInspectable var textColor: UIColor = UIColor.blackColor()
    @IBInspectable var text: String?
    @IBInspectable var showText: Bool = true

    override func drawRect(rect: CGRect) {
        // ...
    }
}
```

Oder in Ziel C:

```
IB_DESIGNABLE
@interface CustomView: UIView

@property (nonatomic, strong) IBInspectable UIColor* textColor;
@property (nonatomic, strong) IBInspectable NSString* text;
@property (nonatomic, assign) IBInspectable BOOL showText;
```

```

@end

@implementation CustomView

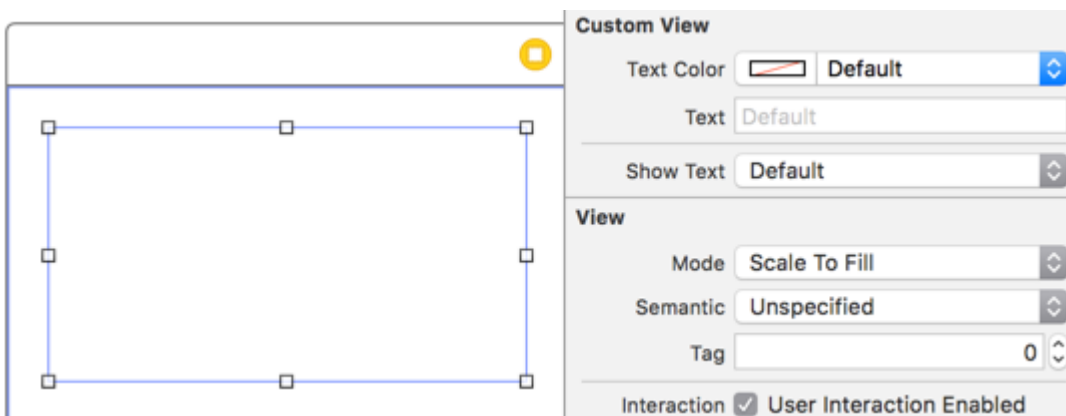
- (instancetype)init {
    if(self = [super init]) {
        self.textColor = [UIColor blackColor];
        self.showText = YES;
    }
    return self;
}

- (void)drawRect:(CGRect)rect {
    //...
}

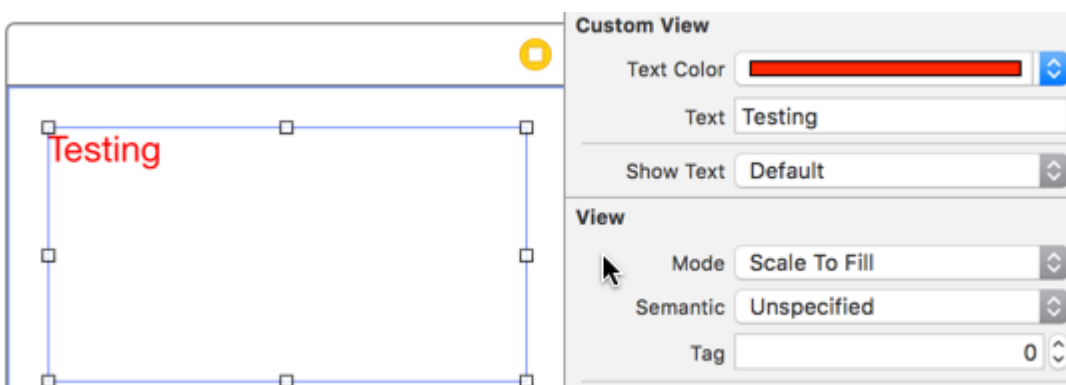
@end

```

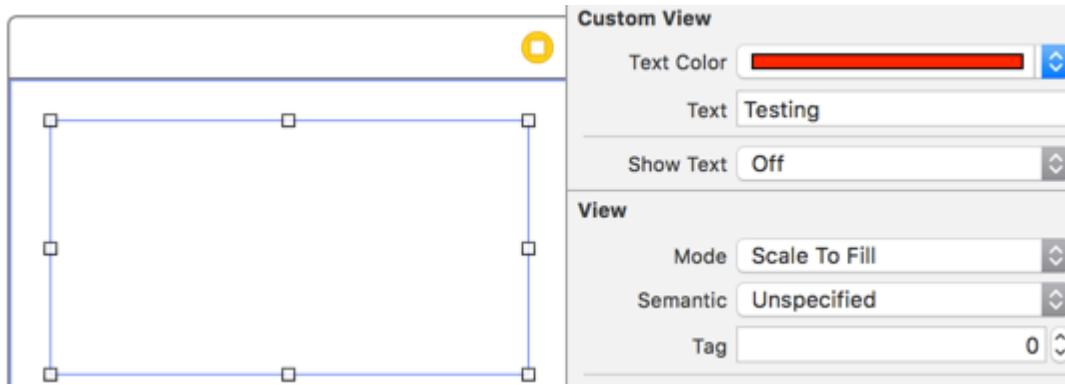
Die nächsten Screenshots zeigen, was in Xcode passiert. Der erste ist, was passiert, nachdem die überarbeitete Klasse hinzugefügt wurde. Beachten Sie, dass es drei neue Oberflächenelemente für die drei Eigenschaften gibt. Die *Textfarbe* zeigt eine Farbauswahl an, *Text* ist nur ein Eingabefeld und *Text anzeigen* gibt uns die Optionen für *Off* und *On* die jeweils *false* und *true* sind.



Das nächste ist, nachdem Sie die *Textfarbe* mit der Farbauswahl in Rot geändert haben. Außerdem wurde etwas Text bereitgestellt, damit die `drawRect` Funktion ihn anzeigen kann. Beachten Sie, dass auch die Ansicht im Interface Builder aktualisiert wurde.

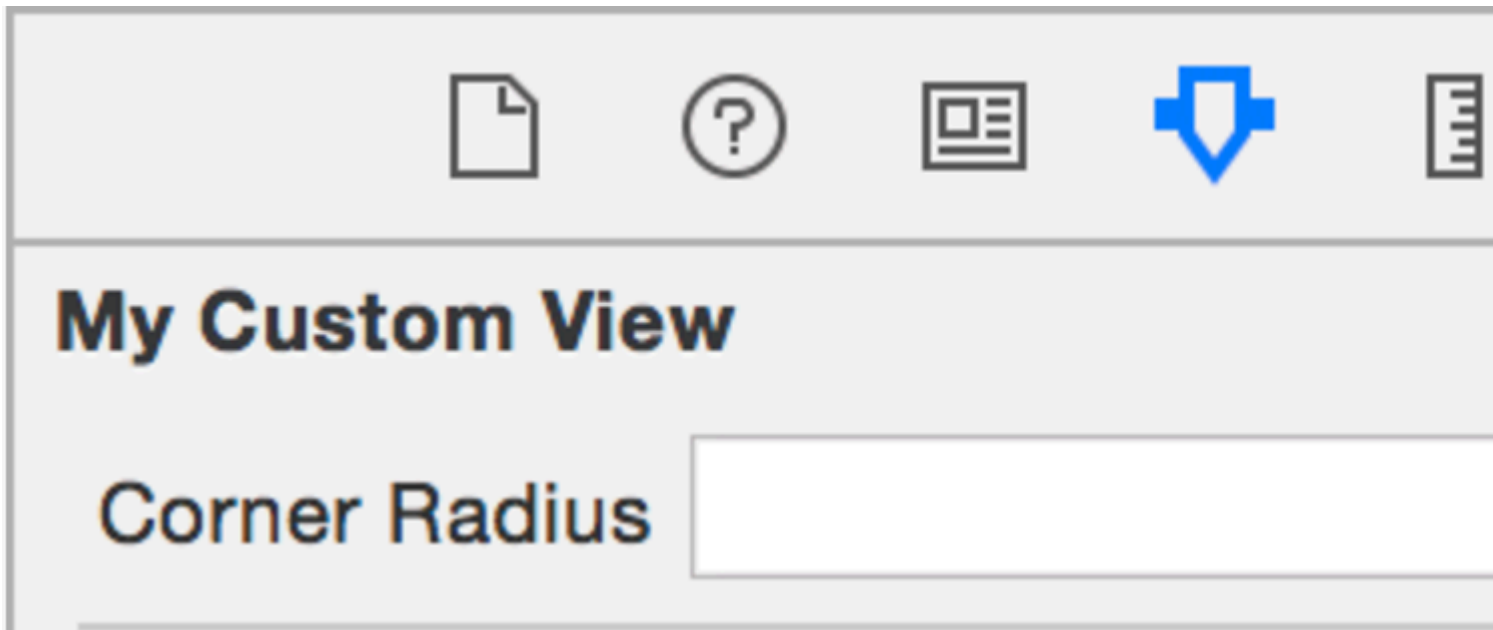


Wenn *Off* im Eigenschafteninspektor die Option „Text anzeigen“ auf „Off“ wird die Textanzeige im Interface Builder ausgeblendet.



Wir kommen jedoch alle zu einer Situation, in der wir abgerundete `UIView` in mehreren Ansichten in Ihrem `Storyboard` erstellen müssen. Statt `IBDesignable` für jede Ansicht von `Storyboard` deklarieren, ist es besser, eine `Extension` von `UIView` zu erstellen und eine Benutzeroberfläche zu erstellen, die nur für jede `UIView` Erstellen Sie eine abgerundete Ansicht, indem Sie den Eckenradius festlegen. Ein konfigurierbarer Randradius auf einer beliebigen `UIView`, die Sie im `Storyboard` erstellen.

```
extension UIView {
    @IBInspectable var cornerRadius:CGFloat {
        set {
            layer.cornerRadius = newValue
            clipsToBounds = newValue > 0
        }
        get {
            return layer.cornerRadius
        }
    }
}
```



UIView animieren

```
let view = UIView(frame: CGRect(x: 0, y: 0, width: 100, height: 100))
view.backgroundColor = UIColor.orange
```



```

self.view.addSubview(view)
UIView.animate(withDuration: 0.75, delay: 0.5, options: .curveEaseIn, animations: {
    //This will cause view to go from (0,0) to
    // (self.view.frame.origin.x,self.view.frame.origin.y)
    view.frame.origin.x = self.view.frame.origin.x
    view.frame.origin.y = self.view.frame.origin.y
}) { (finished) in
    view.backgroundColor = UIColor.blueColor()
}

```

UIView-Erweiterung für Größen- und Rahmenattribute

Wenn wir die x-Koordinate des Ursprungs der Ansicht erhalten möchten, müssen wir wie folgt schreiben:

```
view.frame.origin.x
```

Für die Breite müssen wir schreiben:

```
view.frame.size.width
```

Wenn wir jedoch eine einfache Erweiterung zu einem `UIView`, können wir alle Attribute ganz einfach `UIView`, z.

```

view.x
view.y
view.width
view.height

```

Es wird auch dabei helfen, diese Attribute festzulegen:

```

view.x = 10
view.y = 10
view.width = 100
view.height = 200

```

Und die einfache Erweiterung wäre:

```

extension UIView {

    var x: CGFloat {
        get {
            return self.frame.origin.x
        }
        set {
            self.frame = CGRect(x: newValue, y: self.frame.origin.y, width:
self.frame.size.width, height: self.frame.size.height)
        }
    }

    var y: CGFloat {
        get {
            return self.frame.origin.y

```

```

    }
    set {
        self.frame = CGRect(x: self.frame.origin.x, y: newValue, width:
self.frame.size.width, height: self.frame.size.height)
    }
}

var width: CGFloat {
    get {
        return self.frame.size.width
    }
    set {
        self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
newValue, height: self.frame.size.height)
    }
}

var height: CGFloat {
    get {
        return self.frame.height
    }
    set {
        self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
self.frame.size.width, height: newValue)
    }
}
}
}

```

Wir müssen diese Klassendatei in ein Projekt einfügen und sie kann im gesamten Projekt verwendet werden!

Programmgesteuertes Verwalten des Einfügens und Löschens von UIView in und aus einem anderen UIView

Angenommen, Sie haben einen `parentView` in dem Sie eine neue einfügen möchten `subView` programmatisch (z. B. wenn Sie einen einfügen möchten `UIImageView` in eine `UIViewController`, `UIScrollView` Ansicht), als Sie es wie unten tun können.

Ziel c

```
[parentView addSubview:subView];
```

Schnell

```
parentView.addSubview(subView)
```

Sie können die Unteransicht auch unter einer anderen `subView2`, die bereits eine Unteransicht von `parentView` ist, indem Sie folgenden Code verwenden:

Ziel c

```
[parentView insertSubview:subView belowSubview:subView2];
```

Schnell

```
parentView.insertSubview(subView, belowSubview: subView2)
```

Wenn Sie es oberhalb von `subView2` einfügen `subView2` , können Sie dies folgendermaßen tun:

Ziel c

```
[parentView insertSubview:subView aboveSubview:subView2];
```

Schnell

```
parentView.insertSubview(subView, aboveSubview: subView2)
```

Wenn Sie irgendwo in Ihrem Code ein bestimmtes `subView` nach vorne bringen müssen, so können Sie vor allem die anderen `parentView` des `parentView` so machen:

Ziel c

```
[parentView bringSubviewToFront:subView];
```

Schnell

```
parentView.bringSubviewToFront (subView)
```

Wenn Sie `subView` aus `parentView` entfernen `parentView` , können Sie Folgendes tun:

Ziel c

```
[subView removeFromSuperview];
```

Schnell

```
subView.removeFromSuperview()
```

Erstellen Sie UIView mit Autolayout

```
UIView *view = [[UIView alloc] init];

[self.view addSubview:view];

//Use the function if you want to use height as constraint
[self addSubview:view onParentView:self.view withHeight:200.f];

//Use this function if you want to add view with respect to parent and should resize with it
[self addFullResizeConstraintForSubview:view addedOnParentView:self.view];
```

Funktionen

Funktion zum Hinzufügen einer Ansicht mit fester Höhe unter Verwendung von Autolayout-Einschränkungen

```
-(void)addView:(UIView*)subView onView:(UIView*)parentView withHeight:(CGFloat)height {
    subView.translatesAutoresizingMaskIntoConstraints = NO;

    NSLayoutConstraint *trailing = [NSLayoutConstraint
                                   constraintWithItem:subView
                                   attribute:NSLayoutAttributeTrailing
                                   relatedBy:NSLayoutRelationEqual
                                   toItem:parent
                                   attribute:NSLayoutAttributeTrailing
                                   multiplier:1.0
                                   constant:10.f];

    NSLayoutConstraint *top = [NSLayoutConstraint
                                constraintWithItem:subView
                                attribute:NSLayoutAttributeTop
                                relatedBy:NSLayoutRelationEqual
                                toItem:parent
                                attribute:NSLayoutAttributeTop
                                multiplier:1.0
                                constant:10.f];

    NSLayoutConstraint *leading = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeLeading
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeLeading
                                    multiplier:1.0
                                    constant:10.f];

    [parent addConstraint:trailing];
    [parent addConstraint:top];
    [parent addConstraint:leading];

    NSLayoutConstraint *heightConstraint = [NSLayoutConstraint
                                             constraintWithItem:subView
                                             attribute:NSLayoutAttributeHeight
                                             relatedBy:NSLayoutRelationEqual
                                             toItem:nil
                                             attribute:0
                                             multiplier:0.0
                                             constant:height];

    [subView addConstraint:heightConstraint];
}
```

Funktion fügt eine Einschränkung für die Größenänderung für die erstellte UIView hinzu.

```
-(void)addFullResizeConstraintForSubview:(UIView*)subView
addedOnParentView:(UIView*)parentView{

    subView.translatesAutoresizingMaskIntoConstraints = NO;

    NSLayoutConstraint *trailing = [NSLayoutConstraint
```

```

        constraintWithItem:subView
        attribute:NSLayoutAttributeTrailing
        relatedBy:NSLayoutRelationEqual
        toItem:parent
        attribute:NSLayoutAttributeTrailing
        multiplier:1.0
        constant:10.f];

NSLayoutConstraint *top = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeTop
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeTop
    multiplier:1.0
    constant:10.f];

NSLayoutConstraint *leading = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeLeading
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeLeading
    multiplier:1.0
    constant:10.f];

NSLayoutConstraint *bottom = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeBottom
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeBottom
    multiplier:1.0
    constant:0.f];

[parent addConstraint:trailing];
[parent addConstraint:top];
[parent addConstraint:leading];
[parent addConstraint:bottom];
}

```

Intrinsische Inhaltsgröße verwenden

Wenn Sie eine UIView-Unterklasse erstellen, hilft die Größe des intrinsischen Inhalts, das Festlegen von hartcodierten Höhen- und Breitenbeschränkungen zu vermeiden

ein grundlegender Einblick, wie eine Klasse dies nutzen kann

```

class ImageView: UIView {
    var image: UIImage {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    // omitting initializers
    // convenience init(image: UIImage)

    override func intrinsicContentSize() -> CGSize {

```

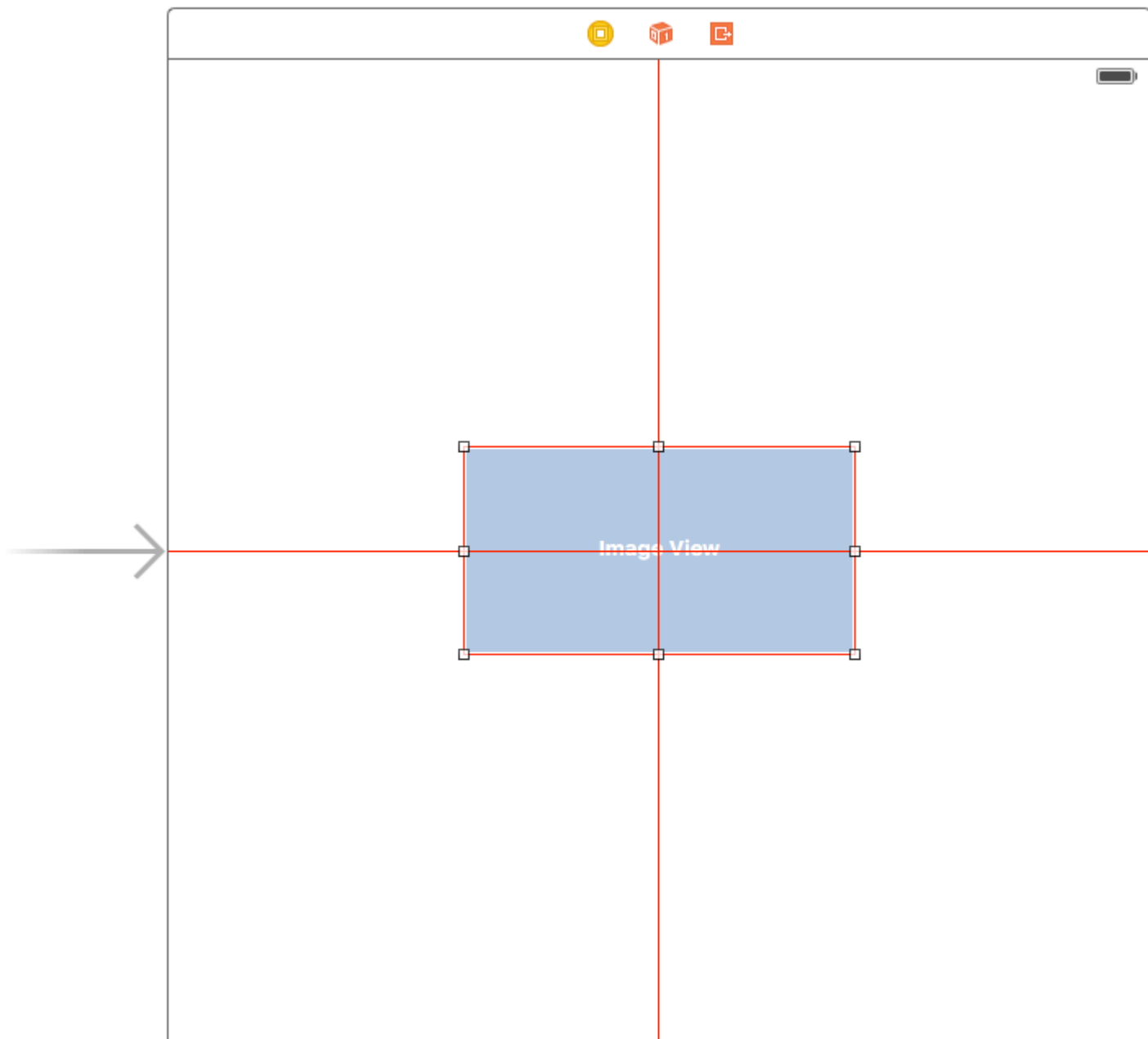
```
        return CGSize(width: image.size.width, height: image.size.height)
    }
}
```

Wenn Sie nur eine einzige Größe `UIViewNoIntrinsicMetric` möchten, können Sie den Wert `UIViewNoIntrinsicMetric` für den Wert `UIViewNoIntrinsicMetric`, den Sie ignorieren möchten.

```
override func intrinsicContentSize() -> CGSize {
    return CGSize(width: UIViewNoIntrinsicMetric, height: image.size.width)
}
```

Vorteile bei Verwendung von AutoLayout und Interface Builder

Man könnte diese `ImageView` (oder `UIImageView`) nehmen und die horizontale Ausrichtung auf das Übersichtszentrum X und die vertikale Ausrichtung auf das Übersichtszentrum Y einstellen.

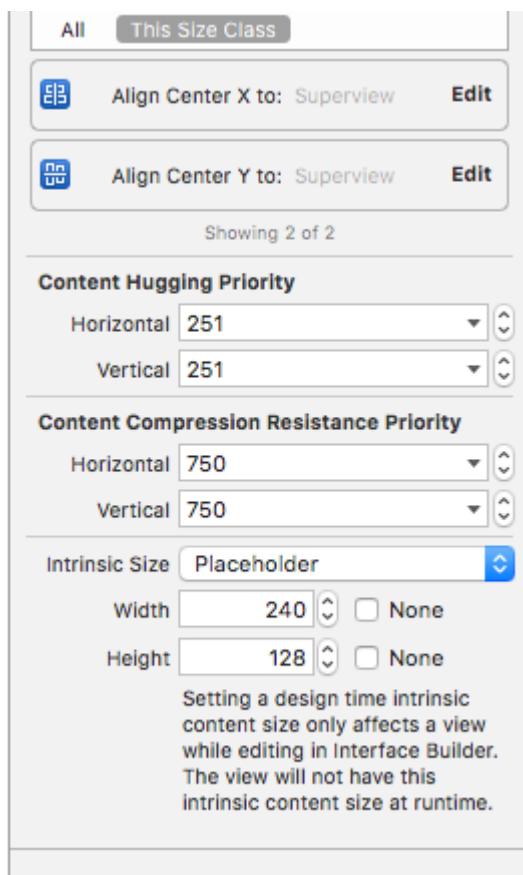


Der Interface Builder wird sich an dieser Stelle bei Ihnen beschweren und die folgende Warnung anzeigen:



An dieser Stelle setzt `Placeholder Intrinsic Size` an.

Im Größeninspektorfenster und in der Dropdown-Liste `Intrinsic Size (Intrinsic Size)` können Sie diesen Wert von `Default` auf `Placeholder` umstellen.



Jetzt entfernt der Interface Builder die vorherigen Warnungen und Sie können diese Größe verwenden, um Ansichten mit dynamischer Größe im Interface Builder zu erstellen.

Schüttele einen Blick

```
extension UIView {
    func shake() {
        let animation = CAKeyframeAnimation(keyPath: "transform.translation.x")
        animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)
        animation.duration = 0.6
        animation.values = [-10.0, 10.0, -7.0, 7.0, -5.0, 5.0, 0.0 ]
        layer.add(animation, forKey: "shake")
    }
}
```



```
}
```

Diese Funktion kann verwendet werden, um durch Schütteln die Aufmerksamkeit auf eine bestimmte Ansicht zu lenken.

UIView online lesen: <https://riptutorial.com/de/ios/topic/858/uiview>

Kapitel 200: UIViewController

Examples

Unterklasse

Mit Subclassing `UIControl` wir Zugriff auf die folgenden Methoden:

- `beginTrackingWithTouch` wird aufgerufen, wenn der Finger zum ersten Mal innerhalb der Grenzen des Steuerelements aufliegt.
- `continueTrackingWithTouch` wird wiederholt aufgerufen, wenn der Finger über das Steuerelement und sogar außerhalb der Grenzen des Steuerelements gleitet.
- `endTrackingWithTouch` wird aufgerufen, wenn der Finger vom Bildschirm abhebt.

MyCustomControl.swift

```
import UIKit

// These are our self-defined rules for how we will communicate with other classes
protocol ViewControllerCommunicationDelegate: class {
    func myTrackingBegan()
    func myTrackingContinuing(location: CGPoint)
    func myTrackingEnded()
}

class MyCustomControl: UIControl {

    // whichever class wants to be notified of the touch events must set the delegate to
    // itself
    weak var delegate: ViewControllerCommunicationDelegate?

    override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool {

        // notify the delegate (i.e. the view controller)
        delegate?.myTrackingBegan()

        // returning true means that future events (like continueTrackingWithTouch and
        // endTrackingWithTouch) will continue to be fired
        return true
    }

    override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool
    {

        // get the touch location in our custom control's own coordinate system
        let point = touch.locationInView(self)

        // Update the delegate (i.e. the view controller) with the new coordinate point
        delegate?.myTrackingContinuing(point)

        // returning true means that future events will continue to be fired
        return true
    }
}
```

```

override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {

    // notify the delegate (i.e. the view controller)
    delegate?.myTrackingEnded()
}
}

```

ViewController.swift

Auf diese Weise wird der View Controller als Delegat eingerichtet und reagiert auf Berührungseignisse von unserem benutzerdefinierten Steuerelement.

```

import UIKit
class ViewController: UIViewController, ViewControllerCommunicationDelegate {

    @IBOutlet weak var myCustomControl: MyCustomControl!
    @IBOutlet weak var trackingBeganLabel: UILabel!
    @IBOutlet weak var trackingEndedLabel: UILabel!
    @IBOutlet weak var xLabel: UILabel!
    @IBOutlet weak var yLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        myCustomControl.delegate = self
    }

    func myTrackingBegan() {
        trackingBeganLabel.text = "Tracking began"
    }

    func myTrackingContinuing(location: CGPoint) {
        xLabel.text = "x: \(location.x)"
        yLabel.text = "y: \(location.y)"
    }

    func myTrackingEnded() {
        trackingEndedLabel.text = "Tracking ended"
    }
}

```

Anmerkungen

- Alternative Methoden zum Erzielen des gleichen Ergebnisses ohne Unterklassifizierung umfassen das Hinzufügen eines Ziels oder die Verwendung eines Gestenerkenners.
- Es ist nicht erforderlich, einen Delegaten mit diesen Methoden zu verwenden, wenn sie nur im benutzerdefinierten Steuerelement selbst verwendet werden. Wir hätten einfach eine `print` Anweisung hinzufügen können, `print` zu zeigen, wie die Ereignisse aufgerufen werden. In diesem Fall würde der Code auf vereinfacht

```

import UIKit
class MyCustomControl: UIControl {

    override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) ->
    Bool {
        print("Began tracking")
    }
}

```

```

        return true
    }

    override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?)
-> Bool {
        let point = touch.locationInView(self)
        print("x: \(point.x), y: \(point.y)")
        return true
    }

    override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {
        print("Ended tracking")
    }
}

```

Erstellen Sie eine Instanz

Schnell

```
let viewController = UIViewController()
```

Ziel c

```
UIViewController *viewController = [UIViewController new];
```

Stellen Sie die Ansicht programmgesteuert ein

Schnell

```
class FooViewController: UIViewController {

    override func loadView() {
        view = FooView()
    }

}

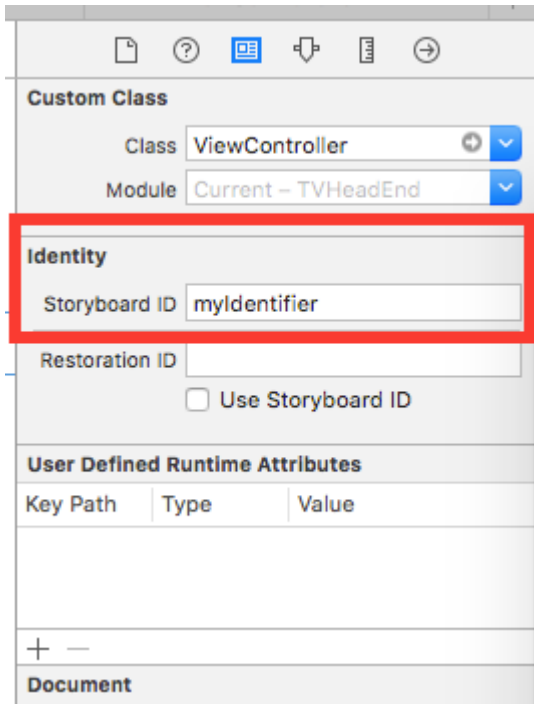
```

Instanziieren Sie aus einem Storyboard

```
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"Main" bundle:nil];
```

Mit einer Kennung :

Vergeben Sie der Szene im Identitätsinspektor des Storyboards eine Storyboard-ID.

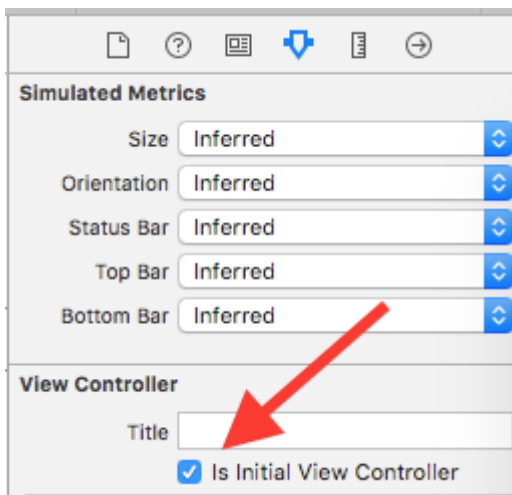


Im Code instanziiieren:

```
UIViewController *controller = [storyboard
instantiateViewControllerWithIdentifier:@"myIdentifier"];
```

Instanziiieren Sie einen ersten Viewcontroller :

Wählen Sie im Storyboard den View Controller aus, wählen Sie dann den Attribut-Inspector aus, und aktivieren Sie das Kontrollkästchen "Ist Initial View Controller".



```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
UIViewController *controller = [storyboard instantiateInitialViewController];
```

Rufen Sie den Container-View-Controller auf

Wenn der Ansichts-Controller innerhalb eines Registerkarten-Controllers angezeigt wird, können Sie wie folgt auf den Registerkarten-Controller zugreifen:

Schnell

```
let tabBarController = viewController.tabBarController
```

Ziel c

```
UITabBarController *tabBarController = self.tabBarController;
```

Wenn der View Controller Teil eines Navigationsstapels ist, können Sie wie folgt auf den Navigationscontroller zugreifen:

Schnell

```
let navigationController = viewController.navigationController
```

Ziel c

```
UINavigationController *navigationController = self.navigationController;
```

Hinzufügen / Entfernen eines untergeordneten Ansichtskontrollers

So fügen Sie einen untergeordneten Ansichtskontroller hinzu:

```
- (void)displayContentController:(UIViewController *)vc {  
    [self addChildViewController:vc];  
    vc.view.frame = self.view.frame;  
    [self.view addSubview:vc.view];  
    [vc didMoveToParentViewController:self];  
}
```

So entfernen Sie einen untergeordneten Ansichtskontroller:

```
- (void)hideContentController:(UIViewController *)vc {  
    [vc willMoveToParentViewController:nil];  
    [vc.view removeFromSuperview];  
    [vc removeFromParentViewController];  
}
```

UIViewController online lesen: <https://riptutorial.com/de/ios/topic/1956/uiviewcontroller>

Kapitel 201: UIWebView

Bemerkungen

UIWebView Delegate-Funktionen: -

Objective-C-Deklarationen

```
- (BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType;

- (void)webView:(UIWebView *)webView
didFailLoadWithError:(NSError *)error;

- (void)webViewDidFinishLoad:(UIWebView *)webView;

- (void)webViewDidStartLoad:(UIWebView *)webView;
```

Examples

Erstellen Sie eine UIWebView-Instanz

Schnell

```
let webview = UIWebView(frame: CGRect(x: 0, y: 0, width: 320, height: 480))
```

Ziel c

```
UIWebView *webview = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

//Alternative way of defining frame for UIWebView
UIWebView *webview = [[UIWebView alloc] init];
CGRect webviewFrame = webview.frame;
webviewFrame.size.width = 320;
webviewFrame.size.height = 480;
webviewFrame.origin.x = 0;
webviewFrame.origin.y = 0;
webview.frame = webviewFrame;
```

URL-Anfrage stellen

Laden Sie Inhalte aus der `url` in die Webansicht

Schnell

```
webview.loadRequest(NSURLRequest(URL: NSURL(string: "http://www.google.com"))) )
```

Ziel c

```
[webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]]];
```

Laden Sie keine Webinhalte mehr

Die Methode `stopLoading()` stoppt den aktuellen Ladevorgang der Webansicht.

Schnell

```
webView.stopLoading()
```

Ziel c

```
[webView stopLoading];
```

Laden Sie den aktuellen Webinhalt neu

Schnell

```
webView.reload()
```

Ziel c

```
[webView reload];
```

Festlegen der Inhaltsgröße

In vielen Fällen, z. B. bei der Verwendung von Webansichten in Tabellensichtzellen, ist es wichtig, die Inhaltsgröße der gerenderten HTML-Seite zu bestimmen. Nach dem Laden der Seite kann dies in der `UIWebViewDelegate` Delegatenmethode berechnet werden:

```
- (void) webViewDidFinishLoad:(UIWebView *) aWebView {
    CGRect frame = aWebView.frame;
    frame.size.height = 1;
    aWebView.frame = frame;
    CGSize fittingSize = [aWebView sizeThatFits:CGSizeZero];
    frame.size = fittingSize;
    aWebView.frame = frame;

    NSLog(@"size: %f, %f", fittingSize.width, fittingSize.height);
}
```

Der Code wendet einen zusätzlichen Trick an, die Höhe der Bahnansicht vor dem Messen der Fittinggröße kurz auf 1 zu setzen. Andernfalls würde einfach die aktuelle Bildgröße angegeben. Nach der Messung stellen wir sofort die Höhe auf die tatsächliche Inhaltshöhe ein.

Quelle

Laden Sie den HTML-String

Webansichten sind nützlich, um lokal generierte HTML-Zeichenfolgen zu laden.

```
NSString *html = @"<!DOCTYPE html><html><body>Hello World</body></html>";  
[webView loadHTMLString:html baseURL:nil];
```

Schnell

```
let htmlString = "<h1>My First Heading</h1><p>My first paragraph.</p>"  
webView.loadHTMLString(htmlString, baseURL: nil)
```

Eine lokale Basis-URL kann angegeben werden. Dies ist nützlich, um auf Bilder, Stylesheets oder Skripts aus dem Anwendungspaket zu verweisen:

```
NSString *html = @"<!DOCTYPE html><html><head><link href='style.css' rel='stylesheet'  
type='text/css'></head><body>Hello World</body></html>";  
[self loadHTMLString:html baseURL:[NSURL fileURLWithPath:[NSBundle mainBundle]  
resourcePath]]];
```

In diesem Fall wird `style.css` lokal aus dem Ressourcenverzeichnis der App geladen. Natürlich kann auch eine Remote-URL angegeben werden.

Laden Sie JavaScript

Wir können benutzerdefiniertes JavaScript für eine `UIWebView` mit der Methode `stringByEvaluatingJavaScriptFromString()` ausführen. Diese Methode gibt das Ergebnis der Ausführung des im Script-Parameter übergebenen JavaScript-Skripts zurück.

Schnell

Skript aus String laden

```
webView.stringByEvaluatingJavaScriptFromString("alert('This is JavaScript!');")
```

Skript aus lokaler Datei laden

```
//Suppose you have javascript file named "JavaScript.js" in project.  
let filePath = NSBundle.mainBundle().pathForResource("JavaScript", ofType: "js")  
do {  
    let jsContent = try String.init(contentsOfFile: filePath!, encoding:  
NSUTF8StringEncoding)  
    webView.stringByEvaluatingJavaScriptFromString(jsContent)  
}  
catch let error as NSError {  
    print(error.debugDescription)  
}
```

Ziel c

Skript aus String laden

```
[webview stringByEvaluatingJavaScriptFromString:@"alert('This is JavaScript!');"];
```

Skript aus lokaler Datei laden

```
//Suppose you have javascript file named "JavaScript.js" in project.  
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"JavaScript" ofType:@"js"];  
NSString *jsContent = [NSString stringWithContentsOfFile:filePath  
encoding:NSUTF8StringEncoding error:nil];  
[webview stringByEvaluatingJavaScriptFromString:jsContent];
```

Hinweis Die Methode `stringByEvaluatingJavaScriptFromString:` wartet synchron, bis die JavaScript-Bewertung abgeschlossen ist. Wenn Sie Webinhalte laden, deren JavaScript-Code Sie nicht überprüft haben, könnte der Aufruf dieser Methode Ihre App aufhängen. Es `WKWebView`, die `WKWebView` Klasse zu übernehmen und stattdessen die Methode `WKWebView` `evaluateJavaScript:completionHandler:` verwenden. `WKWebView` ist jedoch ab iOS 8.0 verfügbar.

Laden Sie Dokumentdateien wie .pdf, .txt, .doc usw.

Anstelle von Webseiten können wir die Dokumentdateien auch in iOS WebView wie PDF, TXT, DOC usw. `loadData`. Die `loadData` Methode wird zum Laden von `NSData` in `WebView` verwendet.

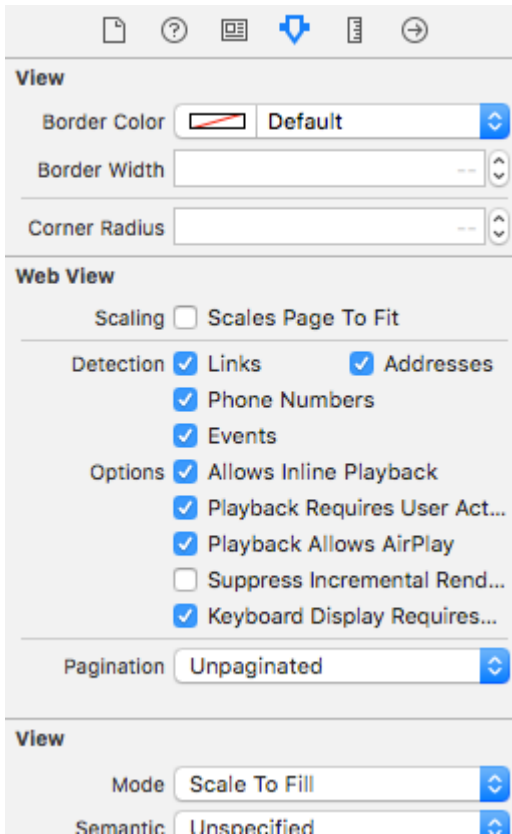
Schnell

```
//Assuming there is a text file in the project named "home.txt".  
let localFilePath = NSBundle.mainBundle().pathForResource("home", ofType:"txt");  
let data = NSFileManager.defaultManager().contentsAtPath(localFilePath!);  
webview.loadData(data!, mimeType: "application/txt", textEncodingName:"UTF-8", baseURL:  
NSURL())
```

Ziel c

```
//Assuming there is a text file in the project named "home.txt".  
NSString *localFilePath = [[NSBundle mainBundle] pathForResource:@"home" ofType:@"txt"];  
NSData *data = [[NSFileManager defaultManager] contentsAtPath:localFilePath];  
[webview loadData:data mimeType:@"application/txt" textEncodingName:@"UTF-8" baseURL:[NSURL  
new]];
```

Machen Sie Links, die in UIWebView anklickbar sind



In vc.h

```
@interface vc : UIViewController<UIWebViewDelegate>
```

in vc.m

```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType{

    if (navigationType == UIWebViewNavigationTypeLinkClicked){
        //open it on browser if you want to open it in same web view remove return NO;
        NSURL *url = request.URL;
        if ([[UIApplication sharedApplication] canOpenURL:url]) {
            [[UIApplication sharedApplication] openURL:url];
        }
        return NO;
    }

    return YES;
}
```

Lokale HTML-Datei in WebView laden

Fügen Sie zunächst die HTML-Datei zu Ihrem Projekt hinzu. (Wenn Sie gefragt werden, ob Sie Optionen zum Hinzufügen der Datei auswählen *möchten* , wählen Sie *ggf. Elemente kopieren.*)

Die folgende Codezeile lädt den Inhalt der HTML-Datei in die Webansicht

```
webView.loadRequest(NSURLRequest(URL: NSURL(fileURLWithPath:  
NSBundle mainBundle().pathForResource("YOUR HTML FILE", ofType: "html")!))
```

- Wenn Ihre HTML-Datei index.html heißt, ersetzen Sie **IHRE HTML-DATEI** durch den **Index**
- Sie können diesen Code entweder in *viewDidLoad ()* oder *viewDidAppear ()* oder in einer anderen Funktion verwenden

UIWebView online lesen: <https://riptutorial.com/de/ios/topic/1452/uiwebview>

Kapitel 202: Umgang mit URL-Schemata

Syntax

1. // Die **canOpenURL**- Methode überprüft, ob eine App das angegebene URL-Schema verarbeiten kann.

2. // schnell

```
UIApplication.sharedApplication (). CanOpenURL (_ aUrl: NSURL)
```

3. // Ziel c

```
[[UIApplication sharedApplication] canOpenURL: (NSURL *) aUrl];
```

4. // Die **openURL**- Methode versucht, eine Ressource nach URL zu öffnen. JA / wahr, wenn geöffnet, sonst NEIN / falsch.

5. // schnell

```
UIApplication.sharedApplication (). OpenURL (_ aUrl: NSURL)
```

6. // Ziel c

```
[[UIApplication sharedApplication] openURL: (NSURL *) aUrl];
```

Parameter

Parameter	Bedeutung
aUrl	eine NSURL-Instanz, in der eine integrierte oder benutzerdefinierte Schemazeichenfolge gespeichert wird

Bemerkungen

In iOS9 und darüber muss Ihre App alle URL-Schemen auflisten, die abgefragt werden sollen. Dies geschieht durch Hinzufügen von `LSApplicationQueriesSchemes` zu `Info.plist`

iOS verfügt über eine integrierte Unterstützung für die Schemata `tel`, `http` / `https`, `sms`, `mailto` und `facetime`. Es unterstützt auch http-basierte URLs für `Youtube`, `Maps` und `iTunes` Apps.

Beispiele für integrierte URL-Schemata:

tel : `tel://123456890` oder `tel:123456890`

http : `http://www.google.com`

facetime : `facetime://azimov@demo.com`

mailto : `mailto://azimov@demo.com`

SMS: `sms://123456890` oder `sms:123456890`

Youtube : `https://www.youtube.com/watch?v=-eCaif2QKfA`

Karten :

- **Verwenden Sie die Adresse:**
`http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California`
- **Verwenden von Koordinaten:** `http://maps.apple.com/?ll=46.683155557,6.683155557`

iTunes : `https://itunes.apple.com/us/artist/randy-newman/id200900`

Hinweis : Im `tel` Schema werden nicht alle Sonderzeichen unterstützt (z. B. `*` oder `#`). Dies geschieht aus Sicherheitsgründen, um zu verhindern, dass Benutzer unbefugte Anrufe umleiten können. In diesem Fall wird die `Phone` App nicht geöffnet.

Examples

Verwenden des integrierten URL-Schemas zum Öffnen der Mail-App

Schnell:

```
if let url = URL(string: "mailto://azimov@demo.com") {
    if UIApplication.shared.canOpenURL(url) {
        UIApplication.shared.openURL(url)
    } else {
        print("Cannot open URL")
    }
}
```

Ziel c:

```
NSURL *url = [NSURL URLWithString:@"mailto://azimov@demo.com"];
if ([[UIApplication sharedApplication] canOpenURL:url]) {
    [[UIApplication sharedApplication] openURL:url];
} else {
    NSLog(@"Cannot open URL");
}
```

Apple URL-Schemata

Dies sind URL-Schemata, die von nativen Apps unter iOS, OS X und watchOS 2 und höher unterstützt werden.

Eröffnungslink in Safari:

Ziel c

```
NSString *stringURL = @"http://stackoverflow.com/";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Schnell:

```
let stringURL = "http://stackoverflow.com/"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

Ein Telefongespräch starten

Ziel c

```
NSString *stringURL = @"tel:1-408-555-5555";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Schnell:

```
let stringURL = "tel:1-408-555-5555"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="tel:1-408-555-5555">1-408-555-5555</a>
```

Starten einer FaceTime-Konversation

Ziel c

```
NSString *stringURL = @"facetime:14085551234";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Schnell:

```
let stringURL = "facetime:14085551234"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="facetime:14085551234">Connect using FaceTime</a>
<a href="facetime:user@example.com">Connect using FaceTime</a>
```

Nachrichten-App öffnen, um eine SMS an den Empfänger zu verfassen:

Ziel c

```
NSString *stringURL = @"sms:1-408-555-1212";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Schnell:

```
let stringURL = "sms:1-408-555-1212"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="sms:">Launch Messages App</a>
<a href="sms:1-408-555-1212">New SMS Message</a>
```

Mail-App öffnen, um eine E-Mail an den Empfänger zu erstellen:

Ziel c

```
NSString *stringURL = @"mailto:foo@example.com";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Schnell:

```
let stringURL = "mailto:foo@example.com"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="mailto:frank@wwdcdemo.example.com">John Frank</a>
```

Sie können auch ein Betrefffeld, eine Nachricht und mehrere Empfänger in die Felder An, Cc und Bcc aufnehmen. (In iOS wird das from-Attribut ignoriert.) Das folgende Beispiel zeigt eine mailto-URL, die mehrere verschiedene Attribute enthält:

```
mailto:foo@example.com?cc=bar@example.com&subject=Greetings%20from%20Cupertino!&body=Wish%20you%20were
```

Hinweis: Der Dialog zum `MFMailComposeViewController` E-Mails kann auch innerhalb der App mit

MFMailComposeViewController .

Umgang mit URL-Schemata online lesen: <https://riptutorial.com/de/ios/topic/3646/umgang-mit-url-schemata>

Kapitel 203: Universelle Links

Bemerkungen

1. Wenn Sie universelle Links unterstützen, können Benutzer von iOS 9 auf einen Link zu Ihrer Website tippen und nahtlos zu Ihrer installierten App weiterleiten, ohne Safari zu durchlaufen. Wenn Ihre App nicht installiert ist, wird durch Antippen eines Links auf Ihre Website Ihre Website in Safari geöffnet.
2. Im Allgemeinen sollte jeder unterstützte Link, auf den in Safari oder in Instanzen von `UIWebView` / `WKWebView` geklickt wird, die App öffnen.
3. Für iOS 9.2 und weniger funktioniert dies nur auf einem Gerät. iOS 9.3 unterstützt auch den Simulator.
4. iOS speichert die Wahl des Benutzers, wenn er Universal Links öffnet. Wenn Sie auf den oberen rechten Breadcrumb tippen, um den Link in Safari zu öffnen, werden alle weiteren Klicks zu Safari und nicht zur App geführt. Sie können standardmäßig zum Öffnen der App zurückkehren, indem Sie im App-Banner auf der Website die Option Öffnen wählen.

Examples

Setup-Server

Sie müssen einen Server online ausführen. Um Ihre iOS-App sicher mit einem Server zu verknüpfen, müssen Sie eine Konfigurationsdatei (`apple-app-site-association`). Dies ist eine `JSON` Datei, die die Domäne und unterstützte Routen beschreibt.

Die `apple-app-site-association` Datei muss über `HTTPS` ohne Weiterleitungen unter `https:// {domain} / Apple-App-Site-Association` zugänglich sein.

Die Datei sieht so aus:

```
{
  "applinks": {
    "apps": [ ],
    "details": [
      {
        "appID": "{app_prefix}.{app_identifier}",
        "paths": [ "/path/to/content", "/path/to/other/*", "NOT /path/to/exclude" ]
      },
      {
        "appID": "TeamID.BundleID2",
        "paths": [ "*" ]
      }
    ]
  }
}
```

HINWEIS - `.json` Sie `.json` an den Dateinamen der `apple-app-site-association` .

Die Schlüssel lauten wie folgt:

`apps` : Sollte ein leeres Array als Wert haben, und es muss vorhanden sein. So will Apple es.

`details` : Ist ein Array von Wörterbüchern, eines für jede von der Website unterstützte iOS-App. Jedes Wörterbuch enthält Informationen über die App, das Team und die Paket-IDs.

Es gibt drei Möglichkeiten, Pfade zu definieren:

`Static` : Der gesamte unterstützte Pfad ist fest codiert, um einen bestimmten Link zu identifizieren, z. B. / static / terms

`Wildcards` : Ein * kann verwendet werden, um dynamische Pfade abzugleichen, z. B. / books / * kann den Pfad zu jeder Autorensseite angeben. ? innerhalb bestimmter Pfadkomponenten, z. B. Bücher / 1? kann verwendet werden, um alle Bücher abzugleichen, deren ID mit 1 beginnt.

`Exclusions` : Wenn Sie einen Pfad mit NOT voranstellen, wird dieser Pfad nicht abgeglichen.

Die Reihenfolge, in der die Pfade im Array angegeben werden, ist wichtig. Frühere Indizes haben eine höhere Priorität. Sobald ein Pfad übereinstimmt, wird die Auswertung angehalten und andere Pfade werden ignoriert. Jeder Pfad unterscheidet zwischen Groß- und Kleinschreibung.

#Website-Code

Den Website-Code finden Sie unter gh-pages branch unter

<https://github.com/vineetchoudhary/iOS-Universal-Links/tree/gh-pages>

Mehrere Domains unterstützen

Jede in der App unterstützte Domäne muss eine eigene Apple-App-Site-Zuordnungsdatei bereitstellen. Wenn der von jeder Domäne bereitgestellte Inhalt unterschiedlich ist, ändert sich auch der Inhalt der Datei, um die jeweiligen Pfade zu unterstützen. Ansonsten kann dieselbe Datei verwendet werden, sie muss jedoch in jeder unterstützten Domäne verfügbar sein.

App-Site-Association-Datei signieren

Hinweis : *Sie können diesen Teil überspringen, wenn Ihr Server `HTTPS`, um Inhalte bereitzustellen und zum Handbuch für die Anwendungskonfiguration zu springen.*

Wenn Ihre App auf iOS 9 abzielt und Ihr Server `HTTPS` zum Bereitstellen von Inhalten verwendet, müssen Sie die Datei nicht signieren. Wenn nicht (z. B. bei der Unterstützung von Handoff unter iOS 8), muss es mit einem `SSL` Zertifikat von einer anerkannten Zertifizierungsstelle signiert werden.

Hinweis : Dies ist nicht das von Apple bereitgestellte Zertifikat, um Ihre App an den App Store zu übermitteln. Es sollte von einem Drittanbieter bereitgestellt werden. Es wird empfohlen, dasselbe Zertifikat zu verwenden, das Sie für Ihren `HTTPS` Server verwenden (obwohl dies nicht erforderlich ist).

Erstellen Sie zum Signieren der Datei zunächst eine einfache TXT-Version. Führen Sie anschließend im Terminal den folgenden Befehl aus:

```
cat <unsigned_file>.txt | openssl smime -sign -inkey example.com.key -signer example.com.pem -certfile intermediate.pem -noattr -nodetach -outform DER > apple-app-site-association
```

Dadurch wird die signierte Datei im aktuellen Verzeichnis ausgegeben. Bei `example.com.key`, `example.com.pem` und `intermediate.pem` es sich um die Dateien, die Ihnen von Ihrer Zertifizierungsstelle zur Verfügung gestellt würden.

Hinweis : Wenn die Datei nicht signiert ist, sollte sie einen Content-Type von `application/json` . Ansonsten sollte es `application/pkcs7-mime` .

Überprüfen Sie Ihren Server mit dem Apple App-Suchprüfungstool

Testen Sie Ihre Webseite auf iOS 9-Such-APIs. Geben Sie eine URL ein. Applebot durchsucht Ihre Webseite und zeigt, wie Sie die besten Ergebnisse erzielen können.

<https://search.developer.apple.com/appsearch-validation-tool/>

IOS-Anwendung einrichten (Universal Links aktivieren)

Das Setup auf der App-Seite erfordert zwei Dinge:

1. Konfigurieren Sie die Berechtigung der App und aktivieren Sie die universellen Links, indem Sie die Funktion für verknüpfte Domänen im Projekt aktivieren.
2. Eingehende Links in Ihrer `AppDelegate` .

1. Konfigurieren Sie die Berechtigung der App und aktivieren Sie universelle Links.

Der erste Schritt beim Konfigurieren der Berechtigungen Ihrer App ist das Aktivieren der App-ID. Tun Sie dies im Apple Developer Member Center. Klicken Sie auf Zertifikate, Bezeichner und Profile und dann auf Bezeichner. Wählen Sie Ihre App-ID aus (erstellen Sie sie ggf. zuerst), klicken Sie auf Bearbeiten, und aktivieren Sie die Berechtigung Zugeordnete Domänen.

ID: com.Universal-Links		
Application Services:		
Service	Development	Distribution
App Group	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Associated Domains	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Data Protection	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Game Center	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
HealthKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
HomeKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wireless Accessory Configuration	<input type="radio"/> Disabled	<input type="radio"/> Disabled
iCloud	<input type="radio"/> Disabled	<input type="radio"/> Disabled
In-App Purchase	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Inter-App Audio	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Apple Pay	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wallet	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Push Notifications	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Personal VPN	<input type="radio"/> Disabled	<input type="radio"/> Disabled

Rufen Sie als Nächstes das App-ID-Präfix und das Suffix auf, indem Sie auf die entsprechende App-ID klicken.

Das Präfix und das Suffix der App-ID sollten mit dem in der Datei zur Zuordnung von Apple-App-Sites übereinstimmen.

Xcode Sie als Nächstes in Xcode das Ziel Ihrer App aus, klicken Sie auf "Funktionen" und aktivieren Sie "Zugeordnete Domänen". Fügen Sie für jede Domain, die Ihre App unterstützt, einen Eintrag hinzu, dem **App-Links** vorangestellt sind :

Zum Beispiel **Applinks: YourCustomDomainName.com**

Was für die Beispiel-App so aussieht:

General
Capabilities
Resource Tags
Info

PROJECT

Universal Links

TARGETS

Universal Links

- ▶ **Apple Pay**
- ▶ **In-App Purchase**
- ▶ **Personal VPN**
- ▶ **Maps**
- ▶ **Keychain Sharing**
- ▶ **Background Modes**
- ▶ **Inter-App Audio**
- ▼ **Associated Domains**

Domains:

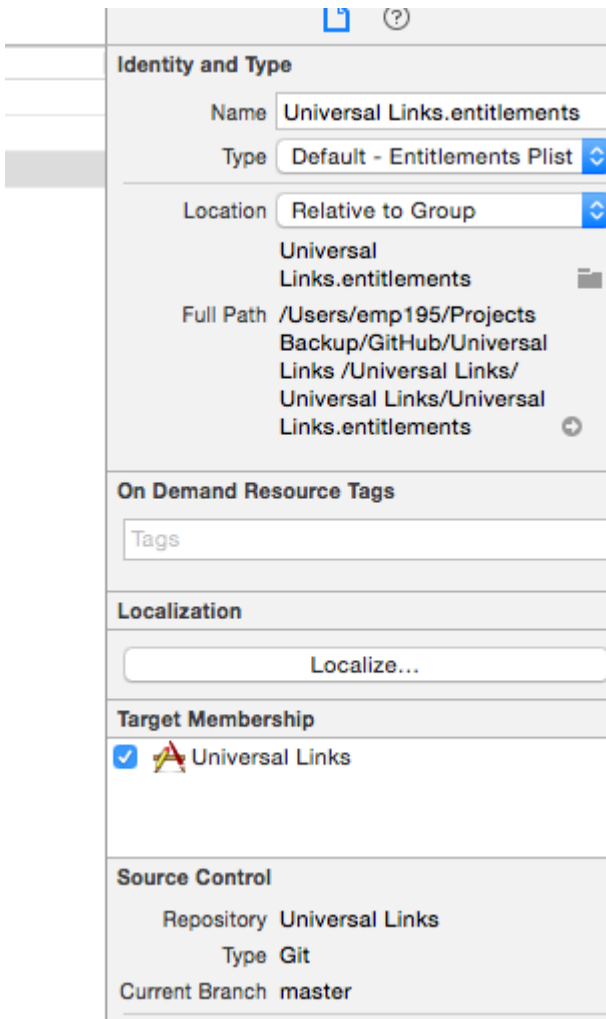
+ -

Steps: ✓ Add the "Associated Domain"

✓ Add the "Associated Domain"

- ▶ **App Groups**
- ▶ **Data Protection**

Hinweis : Stellen Sie sicher, dass Sie dasselbe Team ausgewählt und die gleiche Bundle-ID wie die im App Center registrierte App-ID eingegeben haben. Stellen Sie außerdem sicher, dass die Berechtigungsdatei in Xcode enthalten ist, indem Sie die Datei auswählen. Stellen Sie im Datei-Inspector sicher, dass Ihr Ziel ausgewählt ist.



2. Umgang mit eingehenden Links in Ihrer AppDelegate

Alle Weiterleitungen von Safari zur App für universelle Links werden über die unten stehende Methode in der AppDelegate-Klasse der Anwendung ausgeführt. Sie analysieren diese URL, um die richtige Aktion in der App zu ermitteln.

```
[UIApplicationDelegate application: continueUserActivity: restorationHandler:]
```

Ziel c

```
-(BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity
*)userActivity restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler{
    ///Checking whether the activity was from a web page redirect to the app.
    if ([userActivity.activityType isEqualToString: NSUserActivityTypeBrowsingWeb]) {
        ///Getting the URL from the UserActivity Object.
        NSURL *url = userActivity.webpageURL;
        UIStoryboard *storyBoard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
        UINavigationController *navigationController = (UINavigationController
*)_window.rootViewController;
        if ([url.pathComponents containsObject:@"home"]) {
            [navigationController pushViewController:[storyBoard
instantiateViewControllerWithIdentifier:@"HomeScreenId"] animated:YES];
        }else if ([url.pathComponents containsObject:@"about"]){
            [navigationController pushViewController:[storyBoard
```

```
instantiateViewControllerWithIdentifier:@"AboutScreenId"] animated:YES];  
    }  
}  
return YES;  
}
```

Schnell:

```
func application(application: UIApplication, continueUserActivity userActivity:  
NSUserActivity, restorationHandler: ([AnyObject]?) -> Void) -> Bool {  
    if userActivity.activityType == NSUserActivityTypeBrowsingWeb {  
        let url = userActivity.webpageURL!  
        //handle url  
    }  
    return true  
}
```

iOS-Anwendungscode

Der Code App kann Master - Zweig gefunden werden [hier](#) .

Universelle Links online lesen: <https://riptutorial.com/de/ios/topic/2362/universelle-links>

Kapitel 204: UUID (Universally Unique Identifier)

Bemerkungen

Um die UUID zu speichern, können wir [SSKeychainUtility verwenden](#) . Ein Beispiel finden Sie auf der Github-Seite

Examples

UUID generieren

Zufällige UUID

Schnell

```
func randomUUID() -> NSString{
    return NSUUID.UUID().UUIDString()
}
```

Ziel c

```
+ (NSString *)randomUUID {
    if(NSClassFromString(@"NSUUID")) { // only available in iOS >= 6.0
        return [[NSUUID UUID] UUIDString];
    }
    CFUUIDRef uuidRef = CFUUIDCreate(kCFAllocatorDefault);
    CFStringRef cfuuid = CFUUIDCreateString(kCFAllocatorDefault, uuidRef);
    CFRelease(uuidRef);
    NSString *uuid = [((__bridge NSString *) cfuuid) copy];
    CFRelease(cfuuid);
    return uuid;
}
```

Kennung für Verkäufer

iOS 6

Innerhalb einer Zeile können wir eine UUID wie folgt erhalten:

Schnell

```
let UDIDString = UIDevice.currentDevice().identifierForVendor?.UUIDString
```

Ziel c

```
NSString *UDIDString = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
```

`identifierForVendor` ist eine eindeutige Kennung, die für jede App eines einzelnen Anbieters auf einem einzelnen Gerät gleich bleibt, sofern nicht alle Apps des Herstellers von diesem Gerät gelöscht werden. In der [Apple-Dokumentation](#) finden Sie `UUID` Änderungen dieser `UUID` .

Apples IFA vs. IFV (Apple Identifier für Advertiser vs. Identifier für Anbieter)

- Sie können den IFA zum Messen von Anzeigenklicks und den IFV zum Messen von App-Installationen verwenden.
- IFA verfügt über eingebaute Datenschutzmechanismen, die es ideal für Werbung machen. Im Gegensatz dazu können Entwickler das IFV intern verwenden, um Benutzer zu messen, die ihre Apps installieren.

WENN EINE

- Die `ASIdentifierManager`-Klasse stellt Folgendes bereit
 - **advertisingIdentifier: UUID** : Eine alphanumerische Zeichenfolge, die für jedes Gerät eindeutig ist und nur zum Anzeigen von Werbung verwendet wird.
 - **isAdvertisingTrackingEnabled** : Ein boolescher Wert, der angibt, ob der Benutzer die Anzeigenverfolgung eingeschränkt hat.

IFV

- Die `ASIdentifierManager`-Klasse stellt Folgendes bereit
 - **IdentifierForVendor: UUID** : Eine alphanumerische Zeichenfolge, die ein Gerät für den Hersteller der App eindeutig identifiziert.

Finden Sie Ihr Gerät IFA und IFV [hier](#) .

Erstellen Sie eine UUID-Zeichenfolge für iOS-Geräte

Hier können wir eine `UUID String` in einer Zeile erstellen.

Stellt `UUID`-Zeichenfolgen dar, mit denen Typen, Schnittstellen und andere Elemente eindeutig identifiziert werden können.

Swift 3,0

```
print(UUID().uuidString)
```

Es ist sehr nützlich, um mehrere Geräte mit eindeutiger ID zu identifizieren.

UUID (Universally Unique Identifier) online lesen: <https://riptutorial.com/de/ios/topic/3629/uuid--universally-unique-identifier->

Kapitel 205: Verwenden von Image Aseets

Einführung

Image-Assets werden verwendet, um verschiedene Arten von Image-Assets in unserer iOS-App mithilfe von Xcode zu verwalten und zu organisieren.

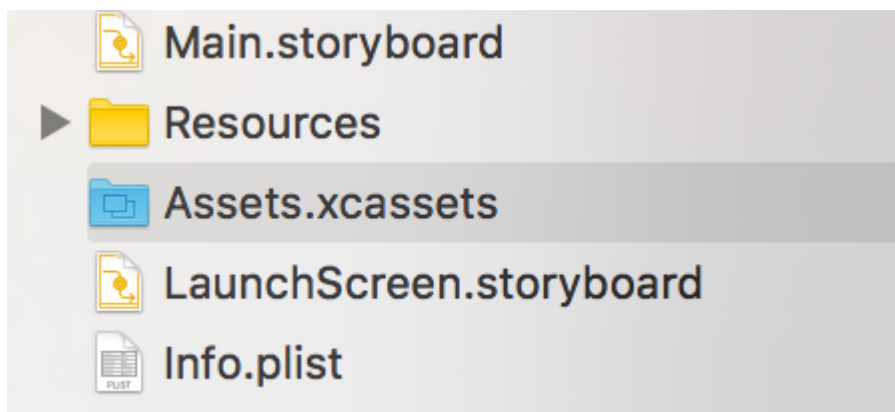
Diese Assets können **App-Symbole, Startbilder, in der App verwendete Bilder, Bilder in voller Größe, Bilder in zufälliger Größe** usw. sein.

Examples

App-Symbol mit Bildelementen

Wenn wir ein neues Projekt in Xcode für unsere neue App erstellen, erhalten Sie verschiedene eingebaute Klassen, Ziele, Tests, Plist-Dateien usw. In ähnlicher Weise erhalten Sie auch die Datei `Assets.xcassets`, die alle `Assets.xcassets` in unserer Projekt.

So sieht diese Datei im Datei-Navigator aus:



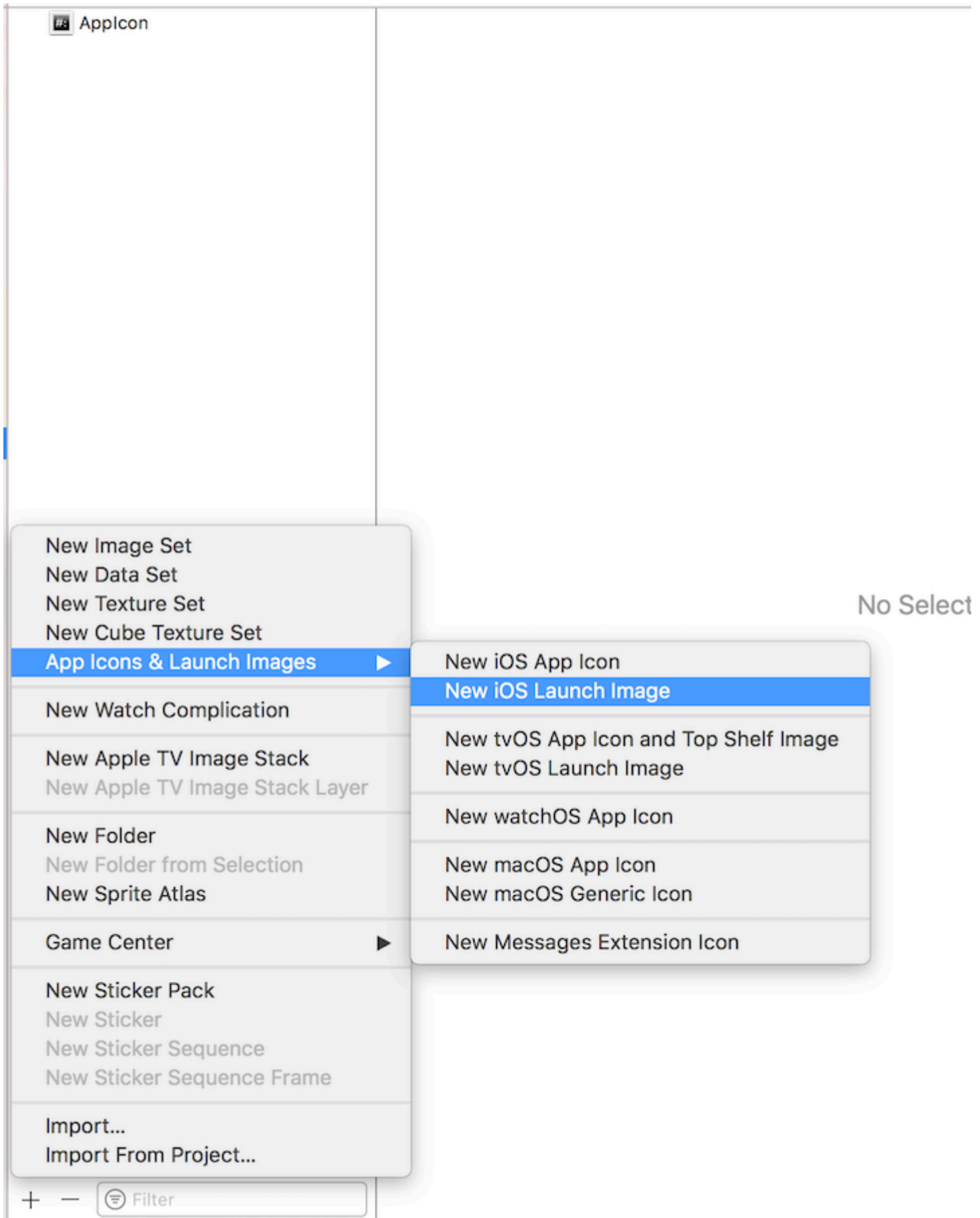
Wenn wir darauf klicken, sieht das so aus:

Wir müssen nur das entsprechende Bild auf jeden leeren quadratischen Block **ziehen und ablegen** . Jeder Schwarze wird uns sagen, wie groß das Bild sein sollte, es wird direkt darunter geschrieben.

Nach dem Ziehen und Ablegen aller Bilder in allen Feldern sieht das so aus:



klicken:



Danach können wir die leeren Kästchen nach Bedarf in Geräte umwandeln, die wir mit dem

Attribut-Inspector unterstützen, indem Sie die Kästchen aktivieren bzw. deaktivieren.

Ich füllte diese Bilder für iPhones von 4 "bis 5,5" und für alle iPads als:



AppIcon



LaunchImage

LaunchImage

Hier sind die Größen aller Startbilder:

```
Retina HD 5.5" iPhone Portrait - iPhone (6, 6S, 7)Plus - 1242x2208px
Retina HD 4.7" iPhone Portrait - iPhone 6, 6S, 7 - 750x1334px
Retina HD 5.5" iPhone Landscape - iPhone (6, 6S, 7)Plus - 2208x1242px
2x iPhone Portrait - (3.5") iPhone 4S - 640x960px
Retina 4 iPhone Portrait - (4") iPhone 5, 5S, 5C, iPod Touch, SE - 640x1136px
2x iPad Portrait - All Retina iPads - 1536x2048px
2x iPad Landscape - All Retina iPads - 2048x1536px
```

Anmerkungen:

1 Nicht-Retina-iPads: Ich habe 1x iPad Portrait and Landscape leer gelassen, da iPads ohne Retina 2x Startbilder durch Skalieren verwenden

2 12,9 "iPad Pro : Es gibt kein Quadrat für dieses iPad, da dieses iPad auch 2x iPad Bilder verwendet, indem es skaliert wird

3 Retina HD 5.5 ": iPads sollten über eine 1920x1080px von 1920x1080px Hochformat und 1080x1920px für das 1080x1920px verfügen. Xcode gibt jedoch Warnen aus und das Startbild wird auf diesen Geräten nicht angezeigt

4 SplitView: Da wir LaunchImage Asset anstelle von LaunchScreen XIB , unterstützt unsere App SplitView auf iPads und 5,5- SplitView iPhones nicht

5 Neu installieren: Wenn unsere App bereits auf dem Gerät installiert ist und wir versuchen, diese neu hinzugefügten Start-Image-Assets auszuführen, zeigt das Gerät beim Start der App manchmal keine Start-Images an. In diesem Fall einfach App vom Gerät löschen, Projekt reinigen und erstellen und ausführen, es werden neue Startbilder angezeigt

Verwenden von Image Aseets online lesen: <https://riptutorial.com/de/ios/topic/10087/verwenden-von-image-aseets>

Kapitel 206: WCSessionDelegate

Einführung

`WCSessionDelegate` arbeitet mit watch OS2 + mit `WatchConnectivity`.
`var watchSession: WCSession? func startWatchSession () {if (WCSession.isSupported ()) {watchSession = WCSession.default () watchSession!.delegate = self watchSession!.activate ()}}` Implementieren Sie die erforderliche Methode: - `didReceiveApplicationContext`

Examples

Kit-Controller beobachten (WKInterfaceController)

```
import WatchConnectivity

var watchSession : WCSession?

override func awake(withContext context: Any?) {
    super.awake(withContext: context)
    // Configure interface objects here.
    startWatchSession()
}

func startWatchSession(){

    if(WCSession.isSupported()){
        watchSession = WCSession.default()
        watchSession!.delegate = self
        watchSession!.activate()
    }
}

//Callback in below delegate method when iOS app triggers event
func session(_ session: WCSession, didReceiveApplicationContext applicationContext: [String : Any]) {
    print("did ReceiveApplicationContext at watch")
}
```

`WCSessionDelegate` online lesen: <https://riptutorial.com/de/ios/topic/8289/wcsessiondelegate>

Kapitel 207: WKWebView

Einführung

WKWebView ist das Herzstück der modernen WebKit-API, die in iOS 8 und OS X Yosemite eingeführt wurde. Es ersetzt UIWebView in UIKit und WebView in AppKit und bietet eine konsistente API für beide Plattformen.

Mit reaktionsschnellem 60-fps-Bildlauf, integrierten Gesten, optimierter Kommunikation zwischen App und Webseite und derselben JavaScript-Engine wie Safari ist WKWebView eine der wichtigsten Ankündigungen der WWDC 2014.

Examples

Einen einfachen WebBrowser erstellen

```
import UIKit
import WebKit

class ViewController: UIViewController, UISearchBarDelegate, WKNavigationDelegate,
WKUIDelegate {

    var searchBar: UISearchBar! //All web-browsers have a search-bar.
    var webView: WKWebView! //The WKWebView we'll use.
    var toolbar: UIToolbar! //Toolbar at the bottom just like in Safari.
    var activityIndicator: UIActivityIndicatorView! //Activity indicator to let the user know
the page is loading.

    override func viewDidLoad() {
        super.viewDidLoad()

        self.initControls()
        self.setTheme()
        self.doLayout()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    func initControls() {
        self.searchbar = UISearchBar()

        //WKUserController allows us to add Javascript scripts to our webView that will
run either at the beginning of a page load OR at the end of a page load.

        let configuration = WKWebViewConfiguration()
        let contentController = WKUserController()
        configuration.userContentController = contentController

        //create the webView with the custom configuration.
```

```

self.webView = WKWebView(frame: .zero, configuration: configuration)

self.toolbar = UIToolbar()
self.layoutToolBar()

self.activityIndicator = UIActivityIndicatorView(activityIndicatorStyle: .gray)
self.activityIndicator.hidesWhenStopped = true
}

func setTheme() {
    self.edgesForExtendedLayout = UIRectEdge(rawValue: 0)
    self.navigationController?.navigationBar.barTintColor = UIColor.white()

    //Theme the keyboard and searchBar. Setup delegates.
    self.searchbar.delegate = self
    self.searchbar.returnKeyType = .go
    self.searchbar.searchBarStyle = .prominent
    self.searchbar.placeholder = "Search or enter website name"
    self.searchbar.autocapitalizationType = .none
    self.searchbar.autocorrectionType = .no

    //Set the WebView's delegate.
    self.webView.navigationDelegate = self //Delegate that handles page navigation
    self.webView.uiDelegate = self //Delegate that handles new tabs, windows, popups,
layout, etc..

    self.activityIndicator.transform = CGAffineTransform(scaleX: 1.5, y: 1.5)
}

func layoutToolBar() {
    //Browsers typically have a back button, forward button, refresh button, and
newTab/newWindow button.

    var items = Array<UIBarButtonItem>()

    let space = UIBarButtonItem(barButtonItemSystemItem: .flexibleSpace, target: nil, action:
nil)

    items.append(UIBarButtonItem(title: "<", style: .plain, target: self, action:
#selector(onBackButtonPressed)))
    items.append(space)
    items.append(UIBarButtonItem(title: ">", style: .plain, target: self, action:
#selector(onForwardButtonPressed)))
    items.append(space)
    items.append(UIBarButtonItem(barButtonItemSystemItem: .refresh, target: self, action:
#selector(onRefreshPressed)))
    items.append(space)
    items.append(UIBarButtonItem(barButtonItemSystemItem: .organize, target: self, action:
#selector(onTabPressed)))

    self.toolbar.items = items
}

func doLayout() {
    //Add the searchBar to the navigationBar.
    self.navigationItem.titleView = self.searchbar

    //Add all other subViews to self.view.
    self.view.addSubview(self.webView)
    self.view.addSubview(self.toolbar)
    self.view.addSubview(self.activityIndicator)
}

```

```

//Setup which views will be constrained.

let views: [String: AnyObject] = ["webView": self.webView, "toolbar": self.toolbar,
"activityIndicator": self.activityIndicator];
var constraints = Array<String>();

constraints.append("H:|-0-[webView]-0-|")
constraints.append("H:|-0-[toolbar]-0-|")
constraints.append("V:|-0-[webView]-0-[toolbar(50)]-0-|")

//constrain the subviews using the above visual constraints.

for constraint in constraints {
    self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views))
}

for view in self.view.subviews {
    view.translatesAutoresizingMaskIntoConstraints = false
}

//constraint the activity indicator to the center of the view.
self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerX, relatedBy: .equal, toItem: self.view, attribute: .centerX, multiplier: 1.0,
constant: 0.0))
self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerY, relatedBy: .equal, toItem: self.view, attribute: .centerY, multiplier: 1.0,
constant: 0.0))
}

//Searchbar Delegates

func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
    self.searchbar.resignFirstResponder()

    if let searchText = self.searchbar.text, url = URL(string: searchText) {
        //Get the URL from the search bar. Create a new NSURLRequest with it and tell the
webView to navigate to that URL/Page. Also specify a timeout for if the page takes too long.
Also handles cookie/caching policy.

        let request = URLRequest(url: url, cachePolicy: .useProtocolCachePolicy,
timeoutInterval: 30)
        self.webView.load(request)
    }
}

//Toolbar Delegates

func onBackButtonPressed(button: UIBarButtonItem) {
    if (self.webView.canGoBack) { //allow the user to go back to the previous page.
        self.webView.goBack()
    }
}

func onForwardButtonPressed(button: UIBarButtonItem) {
    if (self.webView.canGoForward) { //allow the user to go forward to the next page.
        self.webView.goForward()
    }
}

```

```

    }
}

func onRefreshPressed(button: UIBarButtonItem) {
    self.webView.reload() //reload the current page.
}

func onTabPressed(button: UIBarButtonItem) {
    //TODO: Open a new tab or web-page.
}

//WebView Delegates

func webView(_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction,
decisionHandler: (WKNavigationActionPolicy) -> Void) {

    decisionHandler(.allow) //allow the user to navigate to the requested page.
}

func webView(_ webView: WKWebView, decidePolicyFor navigationResponse:
WKNavigationResponse, decisionHandler: (WKNavigationResponsePolicy) -> Void) {

    decisionHandler(.allow) //allow the webView to process the response.
}

func webView(_ webView: WKWebView, didStartProvisionalNavigation navigation:
WKNavigation!) {
    self.activityIndicator.startAnimating()
}

func webView(_ webView: WKWebView, didFailProvisionalNavigation navigation: WKNavigation!,
withError error: NSError) {
    self.activityIndicator.stopAnimating()

    //Handle the error. Display an alert to the user telling them what happened.

    let alert = UIAlertController(title: "Error", message: error.localizedDescription,
preferredStyle: .alert)
    let action = UIAlertAction(title: "OK", style: .default) { (action) in
        alert.dismiss(animated: true, completion: nil)
    }
    alert.addAction(action)
    self.present(alert, animated: true, completion: nil)
}

func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
    self.activityIndicator.stopAnimating()

    //Update our search bar with the webPage's final endpoint-URL.
    if let url = self.webView.url {
        self.searchbar.text = url.absoluteString ?? self.searchbar.text
    }
}

func webView(_ webView: WKWebView, didReceiveServerRedirectForProvisionalNavigation
navigation: WKNavigation!) {
    //When the webview receives a "Redirect" to a different page or endpoint, this is
called.
}

```



```

func webView(_ webView: WKWebView, didCommit navigation: WKNavigation!) {
    //When the content for the webpage starts arriving, this is called.
}

func webView(_ webView: WKWebView, didFail navigation: WKNavigation!, withError error:
NSError) {

}

func webView(_ webView: WKWebView, didReceive challenge: URLAuthenticationChallenge,
completionHandler: (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    completionHandler(.performDefaultHandling, .none) //Handle SSL connections by default.
    We aren't doing SSL pinning or custom certificate handling.

}

//WebView's UINavigationController Delegates

//This is called when a webView or existing loaded page wants to open a new window/tab.
func webView(_ webView: WKWebView, createWebViewWith configuration:
WKWebViewConfiguration, for navigationAction: WKNavigationAction, windowFeatures:
WKWindowFeatures) -> WKWebView? {

    //The view that represents the new tab/window. This view will have an X button at the
    top left corner + a webView.
    let container = UIView()

    //New tabs need an exit button.
    let XButton = UIButton()
    XButton.addTarget(self, action: #selector(onWebViewExit), for: .touchUpInside)
    XButton.layer.cornerRadius = 22.0

    //Create the new webView window.
    let webView = WKWebView(frame: .zero, configuration: configuration)
    webView.navigationDelegate = self
    webView.uiDelegate = self

    //Layout the tab.
    container.addSubview(XButton)
    container.addSubview(webView)

    let views: [String: AnyObject] = ["XButton": XButton, "webView": webView];
    var constraints = Array<String>()

    constraints.append("H:|-(22)-[XButton(44)]")
    constraints.append("H:|-0-[webView]-0-|")
    constraints.append("V:|-(22)-[XButton(44)]-0-[webView]-0-|")

    //constrain the subviews.
    for constraint in constraints {
        container.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views))
    }

    for view in container.subviews {
        view.translatesAutoresizingMaskIntoConstraints = false
    }
}

```

```
        //TODO: Add the containerView to self.view or present it with a new controller. Keep
track of tabs..

        return webView
    }

    func onWebViewExit(button: UIButton) {
        //TODO: Destroy the tab. Remove the new tab from the current window or controller.
    }
}
```

Die benutzerdefinierte `GO` Schaltfläche auf der Tastatur anzeigen:

 <https://stackoverflow.com/>



All Questions 

Show [Interesting](#)

0 

0 

Mysql error, "Specified key was too long; max key length is 767 bytes" need workaround

mysql

6 secs ago [rerat](#)

0 



Error Code: 1305. FUNCTION or PROCEDURE does not exist

q w e r t y u i o p

a s d f g h j k l

 z x c v b n m 

`addScriptMessageHandler:name:` **mehr als einmal wird eine** `NSInvalidArgumentException`
Ausnahme `NSInvalidArgumentException` .

WKWebView online lesen: <https://riptutorial.com/de/ios/topic/3602/wkwebview>

Kapitel 208: Xcode Build & Archive von der Kommandozeile aus

Syntax

- `xcodebuild` `[-project name.xcodeproj]` `-scheme schemename` `[[-destination destinationspecifier] ...]` `[-destination-timeout value]` `[-configuration configurationname]` `[-sdk [sdkfullpath | sdkname]]` `[action ...]` `[buildsetting=value ...]` `[-userdefault=value ...]`

Parameter

Möglichkeit	Beschreibung
-Projekt	Erstellen Sie den Projektnamen.xcodeproj.
-planen	Erforderlich, wenn ein Arbeitsbereich erstellt wird.
-Ziel	Verwenden Sie das Zielgerät
-Aufbau	Verwenden Sie die Build-Konfiguration
-sdk	angegebenes SDK

Bemerkungen

Führen Sie `xcodebuild` aus dem Verzeichnis aus, das Ihr Projekt enthält, um ein Xcode-Projekt zu erstellen. Um einen Xcode-Arbeitsbereich zu erstellen, müssen Sie die Optionen **-workspace** und **-scheme übergeben**, um den Build zu definieren. Die Parameter des Schemas steuern, welche Ziele erstellt werden und wie sie erstellt werden. Sie können jedoch andere Optionen an `xcodebuild` übergeben, um einige Parameter des Schemas zu überschreiben.

Examples

Bauen & Archivieren

Bauen:

```
xcodebuild -exportArchive -exportFormat ipa \  
-archivePath "/Users/username/Desktop/MyiOSApp.xcarchive" \  
-exportPath "/Users/username/Desktop/MyiOSApp.ipa" \  
-exportProvisioningProfile "MyCompany Distribution Profile"
```

Archiv:

```
xcodebuild -project <ProjectName.xcodeproj>  
  -scheme <ProjectName>  
  -sdk iphonesimulator  
  -configuration Debug  
  -destination "platform=iOS Simulator,name=<Device>,OS=9.3"  
  clean build
```

Xcode Build & Archive von der Kommandozeile aus online lesen:

<https://riptutorial.com/de/ios/topic/5027/xcode-build--amp--archive-von-der-kommandozeile-aus>

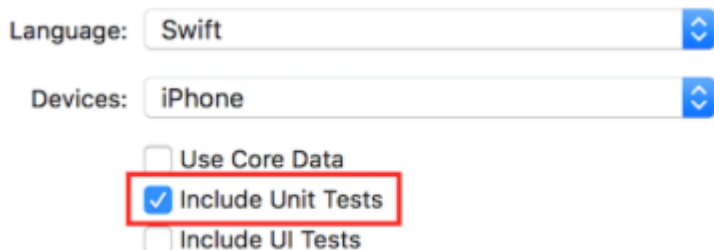
Kapitel 209: XCTest Framework - Unit Testing

Examples

Testdateien zu Xcode Project hinzufügen

Beim Erstellen des Projekts

Sie sollten im Projekterstellungsdialog "Komponententests einbeziehen" aktivieren.



The screenshot shows the Xcode project creation dialog. The 'Language' dropdown is set to 'Swift' and the 'Devices' dropdown is set to 'iPhone'. Below these, there are three checkboxes: 'Use Core Data' (unchecked), 'Include Unit Tests' (checked and highlighted with a red box), and 'Include UI Tests' (unchecked).

Nachdem Sie das Projekt erstellt haben

Wenn Sie dieses Element beim Erstellen des Projekts nicht überprüft haben, können Sie später jederzeit Testdateien hinzufügen. Um dies zu tun:

- 1- Gehen Sie in Xcode zu Ihren Projekteinstellungen
- 2- Gehe zu "Ziele"
- 3- Klicken Sie auf "Ziel hinzufügen".
- 4- Wählen Sie unter "Andere" die Option "Testpaket für Kakao-Touch-Einheiten testen" aus.

Am Ende sollte eine Datei mit dem Namen `[Your app name]Tests.swift`. In Objective-C sollten Sie stattdessen zwei Dateien namens `[Your app name]Tests.h` und `[Your app name]Tests.m`.

`[Your app name]Tests.swift` or `.m` Datei enthält standardmäßig `[Your app name]Tests.swift` or `.m`:

- Ein `XCTest`
- A `[Your app name]Tests XCTestCase` Klasse zur `XCTestCase`
- `setUp`, `tearDown`, `testExample`, `testPerformanceExample` Methoden

Schnell

```
import XCTest
```

```

class MyProjectTests: XCTestCase {

override func setUp() {
    super.setUp()
    // Put setup code here. This method is called before the invocation of each test method in
the class.
}

override func tearDown() {
    // Put teardown code here. This method is called after the invocation of each test method
in the class.
    super.tearDown()
}

func testExample() {
    // This is an example of a functional test case.
    // Use XCTAssert and related functions to verify your tests produce the correct results.
}

func testPerformanceExample() {
    // This is an example of a performance test case.
    self.measure {
        // Put the code you want to measure the time of here.
    }
}
}

```

Ziel c

```

#import <XCTest/XCTest.h>

@interface MyProjectTests : XCTestCase

@end

@implementation MyProjectTests

- (void)setUp {
    [super setUp];
    // Put setup code here. This method is called before the invocation of each test method in the
class.
}

- (void)tearDown {
    // Put teardown code here. This method is called after the invocation of each test method in
the class.
    [super tearDown];
}

- (void)testExample {
    // This is an example of a functional test case.
    // Use XCTAssert and related functions to verify your tests produce the correct results.
}

- (void)testPerformanceExample {
    // This is an example of a performance test case.
    [self measureBlock:^(

```



```
// Put the code you want to measure the time of here.
    }];
}
@end
```

Storyboard und View Controller als Instanzen zur Testdatei hinzufügen

Um mit dem Komponententest zu beginnen, der in der Testdatei ausgeführt wird, werden View Controller und Storyboard getestet. Wir sollten diese beiden Dateien in die Testdatei einfügen.

Definieren des View Controllers

Schnell

```
var viewController : ViewController!
```

Vorstellung des Storyboards und Initialisieren des View Controllers

Fügen Sie der `setUp()` -Methode diesen Code `setUp()` :

Schnell

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
viewController = storyboard.instantiateInitialViewController() as! ViewController
```

Ziel c

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:"Main" bundle:nil];
viewController = (ViewController *) [storyboard instantiateInitialViewController];
```

Auf diese Weise könnten Sie Testmethoden schreiben, und sie wissen, wo sie nach Fehlern suchen sollen. In diesem Fall gibt es View Controller und das Storyboard.

Testmethoden hinzufügen

Laut Apple:

Testmethoden

Eine Testmethode ist eine Instanzmethode einer Testklasse, die mit dem Präfix `test` beginnt, keine Parameter übernimmt und `void` zurückgibt, z. B. `(void) testColorIsRed ()`. Eine Testmethode übt Code in Ihrem Projekt aus. Wenn dieser Code nicht das erwartete Ergebnis liefert, werden Fehler unter Verwendung einer Reihe von Assertions-APIs gemeldet. Beispielsweise kann der Rückgabewert einer Funktion mit einem erwarteten Wert verglichen werden, oder Ihr Test kann die Annahme ausdrücken, dass eine unzulässige Verwendung einer Methode in einer Ihrer Klassen eine Ausnahme auslöst.

Also fügen wir eine Testmethode hinzu, die "test" als Präfix der Methode verwendet, wie zum Beispiel:

Schnell

```
func testSomething() {  
}
```

Ziel c

```
- (void)testSomething {  
}
```

Um die Ergebnisse tatsächlich zu testen, verwenden wir die `XCTAssert()` Methode, die einen booleschen Ausdruck `XCTAssert()` `true` ist, wird der Test als erfolgreich markiert, andernfalls wird er als fehlgeschlagen markiert.

Nehmen wir an, wir haben in View Controller eine Methode namens `sum()` die die Summe zweier Zahlen berechnet. Um es zu testen, verwenden wir diese Methode:

Schnell

```
func testSum(){  
    let result = viewController.sum(4, and: 5)  
    XCTAssertEqual(result, 9)  
}
```

Ziel c

```
- (void)testSum {  
    int result = [viewController sum:4 and:5];  
    XCTAssertEqual(result, 9);  
}
```

Hinweis

Standardmäßig können Sie nicht auf Beschriftungen, Textfelder oder andere UI-Elemente der View Controller-Klasse über die Testklasse zugreifen, wenn sie zuerst in der Storyboard-Datei erstellt wurden. Dies liegt daran, dass sie in der `loadView()` -Methode der View Controller-Klasse initialisiert werden und beim Testen nicht aufgerufen werden. Die beste Möglichkeit, `loadView()` und alle anderen erforderlichen Methoden `loadView()` , ist der Zugriff auf die `view` `viewController` unserer `viewController` . Sie sollten diese Zeile hinzufügen, bevor Sie Elemente der Benutzeroberfläche testen:

```
XCTAssertNotNil(viewController.view)
```

Starten Sie den Test

Testen einer bestimmten Methode

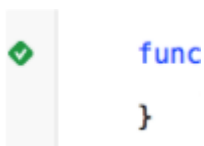
Klicken Sie zum Testen einer bestimmten Methode auf das Quadrat neben der Methodendefinition.

Alle Methoden testen

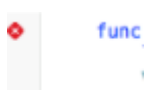
Um alle Methoden zu testen, klicken Sie auf das Quadrat neben der Klassendefinition.

Sehen Sie das Testergebnis

Wenn neben der Definition ein grünes Häkchen angezeigt wird, ist der Test erfolgreich abgeschlossen.



Wenn sich neben der Definition ein rotes Kreuz befindet, ist der Test fehlgeschlagen.



Alle Tests werden ausgeführt

```
Product -> Test OR Cmd + U
```

Es werden alle Tests von allen Testzielen ausgeführt!

Importieren Sie ein Modul, das getestet werden kann

Klassen, Strukturen, Aufzählungen und alle ihre Methoden sind standardmäßig `internal`. Dies bedeutet, dass nur von demselben Modul aus auf sie zugegriffen werden kann. Die Testfälle befinden sich in einem anderen Ziel, dh sie befinden sich in einem anderen Modul. Um auf die zu testende Methode zugreifen zu können, müssen Sie das zu testende Modul mit dem Schlüsselwort `@testable`.

Nehmen wir an, wir haben ein Hauptmodul namens `ToDo` und wollen Tests dafür schreiben. Wir würden dieses Modul so importieren:

```
@testable import ToDo
```

Alle Testmethoden in der Datei mit dieser Importanweisung können jetzt auf alle `internal` Klassen, Strukturen, Enums und alle ihre `internal` Methoden des `ToDo` Moduls `ToDo`.

Sie sollten niemals die Dateien mit den Elementen, die Sie testen möchten, zum Testziel hinzufügen, da dies zu schwer zu debuggenden Fehlern führen kann.

Triggeransicht laden und Aussehen

Laden anzeigen

In einem Test für einen View-Controller möchten Sie manchmal die Ausführung von `loadView()` oder `viewDidLoad()` auslösen. Dies kann durch Zugriff auf die Ansicht erfolgen. Nehmen wir an, Sie haben in Ihrem Test eine View-Controller-Instanz namens `sut` (System unter Test). Dann würde der Code folgendermaßen aussehen:

```
XCTAssertNotNil(sut.view)
```

Aussehen anzeigen

Sie können die Methoden `viewWillAppear(_:)` und `viewDidAppear(_:)` auch auslösen, indem Sie folgenden Code hinzufügen:

```
sut.beginAppearanceTransition(true, animated: true)
sut.endAppearanceTransition()
```

Testklasse schreiben

```
import XCTest
@testable import PersonApp
```

```
class PersonTests: XCTestCase {
    func test_completeName() {
        let person = Person(firstName: "Josh", lastName: "Brown")
        XCTAssertEqual(person.completeName(), "Josh Brown")
    }
}
```

Lass uns jetzt besprechen, was hier los ist. Die `import XCTest` Zeile ermöglicht uns die Erweiterung von `XCTestCase` und die Verwendung von `XCTAssertEqual` (neben anderen Assertions). Wenn Sie `XCTestCase` und unseren `XCTestCase` mit `test XCTestCase` wird sichergestellt, dass Xcode diesen Test automatisch **ausführt**, wenn Sie die Tests im Projekt **ausführen** (**U** oder **Product > Test**). Die `@testable import PersonApp` Zeile importiert unser `PersonApp` Ziel, damit wir Klassen testen und verwenden können, wie zum Beispiel die `Person` in unserem obigen Beispiel. Und schließlich stellt unser `XCTAssertEqual` sicher, dass `person.completeName()` der Zeichenfolge "Josh Brown" .

XCTest Framework - Unit Testing online lesen: <https://riptutorial.com/de/ios/topic/5075/xctest-framework---unit-testing>

Kapitel 210: Zugänglichkeit

Einführung

Dank der Barrierefreiheit von iOS können Benutzer mit Hörbehinderungen und Sehstörungen auf iOS und Ihre Anwendung zugreifen, indem sie verschiedene Funktionen wie VoiceOver, Sprachsteuerung, Weiß auf Schwarz, Mono-Audio, Sprache zu Text usw. unterstützen. Barrierefreiheit in der iOS-App bereitzustellen bedeutet, die App für alle nutzbar zu machen.

Examples

Machen Sie eine Ansicht zugänglich

UINavigationController Ihre UINavigationController Unterklasse als barrierefreies Element, damit sie für VoiceOver sichtbar ist.

```
myView.isAccessibilityElement = YES;
```

Stellen Sie sicher, dass die Ansicht eine aussagekräftige Bezeichnung, einen Wert und einen Hinweis enthält. Weitere Informationen zur Auswahl guter Beschreibungen finden Sie im [Handbuch](#) zur [Barrierefreiheit](#).

Eingabehilfen

Der Eingabehilfenrahmen wird von VoiceOver für Trefferprüfungen verwendet, um den VoiceOver-Cursor zu zeichnen und die Stelle im fokussierten Element zu berechnen, um ein Tippen zu simulieren, wenn der Benutzer den Bildschirm doppelt antippt. Beachten Sie, dass der Rahmen in Bildschirmkoordinaten ist!

```
myElement.accessibilityFrame = frameInScreenCoordinates;
```

Wenn sich Ihre Elemente oder Bildschirmlayouts häufig ändern, sollten Sie das Überschreiben von `accessibilityFrame` in Betracht ziehen, um immer ein aktuelles Problem zu lösen. Die Berechnung des bildschirmrelevanten Rahmens für Unteransichten der Bildlaufansicht kann fehleranfällig und langwierig sein. iOS 10 führt eine neue API ein, um dies zu erleichtern:

```
accessibilityFrameInContainerSpace .
```

Bildschirmwechsel

VoiceOver funktioniert die meiste Zeit über hervorragend und liest Bilder mit Inhalten laut vor und folgt dem Benutzer intuitiv. Leider ist keine allgemeine Lösung perfekt. Manchmal wissen nur Sie als App-Entwickler, wo VoiceOver für eine optimale Benutzererfahrung konzentriert werden sollte. Glücklicherweise überwacht VoiceOver Systemzugriffsbenachrichtigungen auf Hinweise, wo der Fokus liegt. Um den VoiceOver-Cursor manuell zu verschieben, buchen Sie eine Benachrichtigung über einen Eingabehilfen mit geänderter Anzeige:

```
UIAccessibilityPostNotification(UIAccessibilityScreenChangedNotification, firstElement);
```

Wenn diese Benachrichtigung veröffentlicht wird, werden die Benutzer über eine kurze Reihe von Tönen über die Änderung informiert. Der zweite Parameter kann entweder das nächste zu fokussierende Element oder eine Zeichenfolge sein, die die Änderung ankündigt. Veröffentlichen Sie eine Bildschirmänderungsbenachrichtigung nur dann, wenn das VoiceOver-Erlebnis ohne sie schlecht ist und keine andere Problemumgehung vorliegt. Das Bewegen des VoiceOver-Cursors ist wie das Stoßen auf dem Bildschirm eines sehenden Benutzers. Es kann ärgerlich und desorientierend sein, um auf diesem Weg herumgeführt zu werden.

Layout ändern

In vielen Fällen werden Inhalte auf einem einzigen Bildschirm mit neuen oder anderen Inhalten aktualisiert. Stellen Sie sich beispielsweise ein Formular vor, das zusätzliche Optionen enthält, die auf der Antwort des Benutzers auf eine vorherige Frage basieren. In diesem Fall können Sie durch eine Benachrichtigung über eine Layoutänderung entweder die Änderung ankündigen oder sich auf ein neues Element konzentrieren. Diese Benachrichtigung akzeptiert die gleichen Parameter wie die Bildschirmänderungsbenachrichtigung.

```
UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification, firstElement);
```

Ankündigung

Ankündigungen sind nützlich, um Benutzer auf Ereignisse hinzuweisen, für die keine Interaktion erforderlich ist, z. B. „Bildschirm gesperrt“ oder „Laden abgeschlossen“. Verwenden Sie eine speziellere Ankündigung, um Benutzer über Bildschirmänderungen oder kleinere Layoutänderungen zu informieren.

```
UIAccessibilityPostNotification(UIAccessibilityAnnouncementNotification, @"The thing happened!");
```

Elemente bestellen

VoiceOver navigiert unabhängig von der Ansichtshierarchie von links oben nach rechts unten. In der Regel wird der Inhalt in Sprachen von links nach rechts angeordnet, da sehende Personen den Bildschirm in einem „F-förmigen Muster“ scannen. VoiceOver-Benutzer werden erwarten, auf dieselbe Weise wie normale Benutzer zu navigieren. Vorhersagbarkeit und Konsistenz sind sehr wichtig für die Zugänglichkeit. Bitte nehmen Sie keine Anpassungen vor, die das Standardverhalten „verbessern“ (z. B. die Tabulatorleiste zuerst in der Wischreihenfolge bestellen). Wenn Sie jedoch ein Feedback erhalten haben, dass die Reihenfolge der Elemente in Ihrer App überraschend ist, gibt es mehrere Möglichkeiten, wie Sie das Erlebnis verbessern können.

Wenn VoiceOver die Unteransichten einer Ansicht nacheinander lesen soll, dies jedoch nicht ist, müssen Sie VoiceOver möglicherweise mitteilen, dass die in einer einzelnen Ansicht enthaltenen Elemente miteinander verbunden sind. Sie können dies tun, indem Sie

shouldGroupAccessibilityChildren :

```
myView.shouldGroupAccessibilityChildren = YES;
```

Um komplexe Navigationsstrukturen zu unterstützen, die sich über mehrere Container erstrecken oder Schnittstellen enthalten, die ohne UIKit gerendert werden, sollten Sie das Containerprotokoll in der übergeordneten Ansicht implementieren.

Container für Barrierefreiheit

VoiceOver kann in vielen Apps unter iOS navigieren, da die meisten UIKit Klassen UIAccessibilityProtocol implementieren. Funktionen, die keine Bildelemente mit UIView , einschließlich Apps, die Core Graphics oder Metal zum Zeichnen verwenden, müssen diese Elemente für den barrierefreien UIView beschreiben. Ab iOS 8.0 kann dies durch Zuweisen einer Eigenschaft in UIView die nicht zugängliche Elemente enthält:

```
myInaccessibleContainerView.accessibilityElements = @[elements, that, should, be, accessible];
```

Jedes Objekt im Array kann eine Instanz von UIAccessibilityElement oder eine beliebige andere Klasse sein, die UIAccessibilityProtocol . Die untergeordneten Elemente sollten in der Reihenfolge zurückgegeben werden, in der der Benutzer sie navigieren soll. Als Anwendungsautor können Sie Barrierefreiheitscontainer verwenden, um die Standardreihenfolge der VoiceOver-Swipe-Navigation von links nach rechts zu überschreiben. Da UIView implementiert UIAccessibilityProtocol , können Sie Instanzen kombinieren UIAccessibilityElement und UIView in der gleichen Reihe von Kinder Zugänglichkeit Elemente. Wenn Sie Elemente manuell zuweisen, müssen Sie keine dynamischen Protokollmethoden für das Eingabehilfen implementieren. Möglicherweise müssen Sie jedoch eine Bildschirmänderungsbenachrichtigung für die von VoiceOver erkannten Elemente ausgeben.

Modale Ansicht

Modale Ansichten erfassen die Aufmerksamkeit des Benutzers vollständig, bis eine Aufgabe abgeschlossen ist. iOS macht dies für Benutzer klarer, indem alle anderen Inhalte abgedunkelt und deaktiviert werden, wenn eine modale Ansicht, beispielsweise eine Warnung oder ein Popover, sichtbar ist. Eine App, die eine benutzerdefinierte modale Benutzeroberfläche implementiert, muss VoiceOver darauf hinweisen, dass diese Ansicht die ungeteilte Aufmerksamkeit des Benutzers verdient, indem accessibilityViewIsModal . Beachten Sie, dass diese Eigenschaft nur für die Ansicht festgelegt werden sollte, die modalen Inhalt enthält, nicht für Elemente, die in einer modalen Ansicht enthalten sind.

```
myModalView.accessibilityViewIsModal = YES;
```

Durch das Markieren einer Ansicht als modal wird VoiceOver aufgefordert, die Ansichten von Geschwistern zu ignorieren. Wenn Sie nach dem Festlegen dieser Eigenschaft feststellen, dass VoiceOver noch in anderen Elementen in Ihrer App navigiert, versuchen Sie, die Problemansichten auszublenden, bis der modale Modus verworfen wird.

Elemente ausblenden

Die meisten UIKit-Klassen, einschließlich UIView, halten sich an `UIAccessibilityProtocol` und geben standardmäßig korrekte Werte zurück. Es ist leicht `UIView`, dass ein `UIView`, der auf "hidden" gesetzt ist, auch nicht in der Hierarchie für `UIView` enthalten ist und von VoiceOver nicht navigiert werden kann. Während dieses Standardverhalten normalerweise ausreicht, gibt es Zeiten, in denen eine Ansicht in der Ansichtshierarchie vorhanden, jedoch nicht sichtbar oder navigierbar ist. Beispielsweise kann eine Sammlung von Schaltflächen von einer anderen Ansicht überlappt werden, wodurch sie für einen sehenden Benutzer unsichtbar wird. VoiceOver versucht jedoch immer noch, durch sie zu navigieren, da sie technisch nicht für `UIKit` verborgen `UIKit` und daher immer noch in der `UIKit` vorhanden sind. In solchen Fällen müssen Sie VoiceOver mitteilen, dass auf die übergeordnete Ansicht nicht zugegriffen werden kann. Sie können dies tun, indem Sie die Ansicht explizit von `UIKit` ausblenden, indem Sie `hidden` festlegen, wenn die Ansicht aus dem Bildschirm verschwindet:

```
myViewFullofButtons.hidden = YES;
```

Alternativ können Sie die übergeordnete Ansicht sichtbar lassen und die untergeordneten Elemente einfach vor der Eingabehierarchie verbergen:

```
myViewFullofButtons.accessibilityElementsHidden = YES;
```

Temporäre Ansichten sind ein weiterer Ort, an dem Sie Elemente aus der Hierarchie für Eingabehilfen ausblenden möchten, während sie für Benutzer sichtbar bleiben. Beispielsweise ist die Ansicht, die angezeigt wird, wenn Sie die Lautstärketaste drücken, für sehende Benutzer sichtbar, erfordert jedoch keine Aufmerksamkeit wie bei einer normalen Warnung. Sie möchten nicht, dass VoiceOver den Benutzer unterbricht und den Cursor von der aktuellen Lautstärke wegbewegt, um die neue Lautstärke anzukündigen, zumal die Einstellung der Lautstärke bereits ein hörbares Feedback durch das Klickgeräusch liefert. In solchen Fällen möchten Sie die Ansicht mithilfe von `accessibilityElementsHidden` ausblenden.

Zugänglichkeit online lesen: <https://riptutorial.com/de/ios/topic/773/zuganglichkeit>

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit iOS	Ali Beadle , Allan Burlison , Anand Nimje , Anatoliy , Ashutosh Dave , Bhadresh Kathiriya , bjtitus , Blachshma , bmike , Charlie H , Cin316 , Community , Dair , dan , deyanm , Efraim Weiss , Erik Godard , FelixSFD , Fogmeister , Hudson Taylor , Irfan , J F , Jack Ngai , James , Josh Brown , jrf , Kampai , Kevin , Losiowaty , M. Galban , Maddyツヅ , Matthew Cawley , Md. Ibrahim Hassan , Midhun MP , Miguel Cabezas , Muhammad Zohaib Ehsan , Pro Q , PSN , RamenChef , Sam Fischer , Seyyed Parsa Neshaei , shim , Skeleton Bow , Stephen Leppik , Steve Moser , Suragch , SuzGupta , The_Curry_Man , ThrowingSpoon , Undo , user3480295 , user6939352 , Vignan
2	3D Touch	4444 , Harshal Bhavsar , LinusGeffarth , Md. Ibrahim Hassan , Onur Tuna , Stephen Leppik , tobeiosdeveloper
3	AFNetworking	4444 , Mayank Patel , OhadM , Ruby
4	AirDrop	FelixSFD , Md. Ibrahim Hassan
5	AirPrint-Tutorial in iOS	Md. Ibrahim Hassan
6	Alamofire	Alex Koshy , Josh Caswell , Sour LeangChhean , yogesh wadhwa
7	Ändern der Größe von UIImage	Rahul
8	Antrag auf Antragsbewertung / Überprüfung	Abhijit
9	AppDelegate	CodeChanger , Oleh Zayats , Saamil Shah
10	App-Einreichungsprozess	Nermin Sehic
11	App-ID erstellen	yogesh wadhwa
12	App-Transportsicherheit (ATS)	breakingobstacles , D4ttatraya , esthepiking , FelixSFD , Mehul Chuahan , nathan
13	App-weiten Operationen	midori

14	ARC (automatische Referenzzählung)	4444 , Irfan , John Militer , Ketan P , Tricertops
15	attributedText in UILabel	vp2698
16	Auf Netzwerkverbindung prüfen	ajmccall , breakingobstacles , Mick MacCallum , pableiros , sushant jagtap
17	Automatisches Layout	alaphao , amar , Anuj Joshi , Bean , Bhumit Mehta , BlackDeveraux , dasdom , Dennis , Dima Deplov , Dinesh Raja , Đông An , Harshal Bhavsar , Hasintha Janka , Irfan , Jano , juanjo , keithbhunter , Mahesh , Mert Buran , Mr. Xcoder , NSNoob , ozgur , Pärserk , Rajesh , Sally , Sandy , Stephen Leppik , Suragch , Undo , user3480295 , Vignan
18	AVPlayer und AVPlayerViewController	Bonnie , Chirag Desai , Gazi Alankus , Harshal Bhavsar , Konda Yadav , Stephen Leppik
19	AVSpeechSynthesizer	Ali Beadle , Bhumit Mehta , Harshal Bhavsar , Midhun MP , Stephen Leppik
20	AWS SDK	OhadM
21	Beacons mit CoreBluetooth konfigurieren	Beto Caldas
22	Behandeln Sie mehrere Umgebungen mit Makro	Tien
23	Benutzerdefinierte Auswahlmethoden für UITableViewCells	Kamil Harasimowicz
24	Benutzerdefinierte Schriftarten	Alexi , Dima Deplov , Harshal Bhavsar , Maddy ヅヅ , njuri , Stephen Leppik , Tommie C.
25	Benutzerdefinierte Tastatur	Md. Ibrahim Hassan
26	Benutzerdefinierte UIViews aus XIB-Dateien	backslash-f , Code.Warrior , Harshal Bhavsar , idocode , Nirav Bhatt , Sharpkits Innovations , Stephen Leppik
27	Benutzerdefiniertes UITextField	D4ttatraya
28	Bilder asynchron laden	J.Paravicini
29	Block	4444 , animuson , Joshua , Mehul Chuahan , Ruby , Tamarous , user459460

30	CAAnimation	Bhavin Ramani , James P , Mr. Xcoder , Narendra Pandey , Rahul , Rob , Undo
31	CAGradientLayer	Bhavin Ramani , Harshal Bhavsar , Sam Fischer , Stephen Leppik , Undo
32	CALayer	Alistra , Dunja Lalic , HariKrishnan.P , Harshal Bhavsar , ignotusverum , iOS BadBoy , Kamil Harasimowicz , Luiz Henrique Guimaraes , Stephen Leppik , Suragch , Viktor Simkó , william205
33	Carthage iOS-Setup	Md. Ibrahim Hassan
34	CAShapeLayer	Filip Radelic , HariKrishnan.P , Harshal Bhavsar , Narendra Pandey , Stephen Leppik
35	CGContext-Referenz	4444 , Narendra Pandey
36	CLLocation	amar , Duly Kinsky , FelixSFD , Siddharth Sunil , Sujania , That lazy iOS Guy , void , Zee
37	CloudKit	Seyyed Parsa Neshaei
38	Code-Signatur	HaemEternal
39	Codierbar	Ashish Kakkad
40	Content Hugging / Inhaltskomprimierung im Autolayout	Mehul Chuahan
41	Core Graphics	Dunja Lalic , Josh Caswell , Seyyed Parsa Neshaei , Sunil Sharma , Unheilig
42	Core Spotlight in iOS	Md. Ibrahim Hassan
43	CoreImage-Filter	Md. Ibrahim Hassan
44	CTCallCenter	MANI , Md. Ibrahim Hassan , OhadM
45	CydiaSubstrate Tweak	gkpln3
46	Daten zwischen View Controllern übergeben (mit MessageBox-Konzept)	StackUnderflow
47	Daten zwischen View- Controllern übergeben	Arulkumar , Ashish Kakkad , BorisE , Bright Future , Dima Deplov , dispute , FelixSFD , Honey , ignotusverum , Irfan , Jake Runzer , juanjo , Kasun Randika , Kendall Lister , Kyle KIM , Luca D'Alberti , muazhud , OhadM , RamenChef ,

		rustproofFish , salabaha , StackUnderflow , Steve Moser , Suragch , Tamarous , timbroder , Undo , WMios , Yagnesh Dobariya
48	Debuggen von Abstürzen	NobodyNada
49	Deep Linking in iOS	bryanjclark , Dunja Lalic , FelixSFD , sanman
50	DispatchGroup	Brandon , Fonix
51	Dynamischer Typ	Alvin Abia , H. M. Madrone , Harshal Bhavsar , James P , Stephen Leppik
52	Erstellen Sie ein benutzerdefiniertes Framework in iOS	Saeed-rz
53	Erstellen Sie ein Video aus Bildern	Tiko
54	Erstellen Sie eine .ipa-Datei, die mit Apploadloader im Appstore hochgeladen wird	Anuj Joshi
55	Erweiterung für umfassende Push-Benachrichtigung - iOS 10.	Oleh Zayats
56	EventKit	Seyyed Parsa Neshaei
57	FacebookSDK	Brian , Harshal Bhavsar , Irfan , Mehul Chuahan , OhadM , Ravi Prakash Verma , Stephen Leppik
58	Farbe der Statusleiste ändern	Alex Rouse , danshevluk , Harshal Bhavsar , Mr. Xcoder , shim , Stephen Leppik , Steve Moser , william205 , WMios
59	FCM Messaging in Swift	Saeed-rz
60	FileHandle	Nikhlesh Bagdiya
61	GameCenter-Bestenlisten	4444 , Cyril Ivar Garcia , Harshal Bhavsar , Stephen Leppik
62	GameplayKit	BennX , Seyyed Parsa Neshaei
63	GCD (Grand Central Dispatch)	Andrea Antonioni , DS Dharma , Fonix , Md. Ibrahim Hassan , skyline75489
64	Gesichtserkennung mit	Md. Ibrahim Hassan

CoreImage / OpenCV		
65	Graph (Coreplot)	MarmiK , Md. Ibrahim Hassan
66	Größenklassen und Adaptivität	Tim
67	Grundlegende Textdatei-E / A	Idan
68	Healthkit	Md. Ibrahim Hassan
69	Hintergrund festlegen	Adriana Carelli , Andreas , Bhadresh Kathiriya , Harshal Bhavsar , Md. Ibrahim Hassan , user459460
70	Hintergrundmodi	Seyyed Parsa Neshaei
71	Hintergrundmodi und Ereignisse	Ashish Kakkad
72	HINZUFÜGEN EINES SWIFT BRIDGING HEADER	yogesh wadhwa
73	iBeacon	amar , Arefly , Harshal Bhavsar , Stephen Leppik
74	IBOutlets	Fabio , SharkbaitWhohaha
75	In-App-Kauf	Cyril Ivar Garcia , Martin , rigdonmr , WMios
76	Initialisierungs-Idiome	Jano
77	Integration von SqlCipher	Nirav
78	iOS - Implementierung von XMPP mit dem Robbie Hanson-Framework	Saheb Roy
79	iOS 10- Spracherkennungs-API	rohit90 , Stephen Leppik
80	iOS Google Places-API	Cyril Ivar Garcia , Vignan
81	iOS TTS	Ali Abbas , Stephen Leppik
82	IOS-Version überprüfen	Bhavin Ramani , byJeevan , James P , Joshua , njuri , Samuel Teferra , Sandy
83	Kategorien	Faran Ghani , simple_code
84	Kernbewegung	Md. Ibrahim Hassan , RamenChef

85	Kerndatei	Ankit chauhan , Md. Ibrahim Hassan
86	Kernstandort	Harshal Bhavsar , Mayuri R Talaviya , Mehul Chuahan , mtso , quant24 , Stephen Leppik , sushant jagtap , william205
87	Kettenblöcke in einer Warteschlange (mit MKBlockQueue)	StackUnderflow
88	Kontakte-Framework	Md. Ibrahim Hassan , Seyyed Parsa Neshaei
89	Konvertieren Sie HTML in NSAttributedString-Zeichenfolge und umgekehrt	Md. Ibrahim Hassan
90	Konvertieren Sie NSAttributedString in UIImage	Md. Ibrahim Hassan
91	Laufzeit in Objective-C	halil_g
92	Leitfaden zur Auswahl der besten iOS-Architekturmuster	Phani Sai
93	Lokalisierung	4444 , animuson , Joshua , Ruby , WMios
94	MeinLayout	
95	Mitteilungen	Amanpreet , Anh Pham , Ashish Kakkad , Bhadresh Kathiriya , BloodWook , Bonnie , Honey , Hossam Ghareeb , iOS BadBoy , J F , Patrick Beard , Pavel Gurov , sanman , Seyyed Parsa Neshaei , tilo
96	MKDistanceFormatter	Harshal Bhavsar , Md. Ibrahim Hassan , Stephen Leppik , Undo
97	MKMapView	Arnon Rodrigues , Brian , FelixSFD , Harshal Bhavsar , Kosuke Ogawa , Mahesh , Mehul Thakkar , Ortwin Gentz , Reinier Melian , Stephen Leppik
98	ModelPresentationStyles	Dishant Kapadiya
99	Momentaufnahme von UIView	Bright Future , Darshit Shah , Md. Ibrahim Hassan , SpaceDog
100	MPMediaPickerDelegate	FelixSFD , George Lee
101	MPVolumeView	lostAtSeaJoshua

102	Multicast-Delegierte	Rahul
103	MVP-Architektur	Oleh Zayats
104	MVVM	JPetric
105	Navigationsleiste	Md. Ibrahim Hassan , Mehul Chuahan
106	NSArray	Krunal , user5553647
107	NSAttributedString	Bhavin Ramani , Harshal Bhavsar , Jinhuan Li , Kirit Modi , Luiz Henrique Guimaraes , Mansi Panchal , Stephen Leppik , Tim , Tim Ebenezer , Undo
108	NSBundle	wdywayne
109	NSData	Felipe Cypriano , maxkonovalov , Seyyed Parsa Neshaei
110	NSDate	Bonnie , Charles , dasdom , Dunja Lalic , ERbittuu , FelixSFD , Harshal Bhavsar , Jon Snow , Josh Caswell , lostAtSeaJoshua , maxkonovalov , Mehul Thakkar , NSNoob , Nykholas , OhadM , Sally , Samuel Teferra , Sandy , Seyyed Parsa Neshaei , Stephen Leppik , tharkay , tobeiosdeveloper
111	NSHTTPCookieStorage	balagurubaran
112	NSInvocation	Md. Ibrahim Hassan
113	NSNotificationCenter	Alex Kallam , Alex Koshy , Anand Nimje , Bence Pattogato , Bright Future , Ichthyocentaurs , Jacopo Penzo , James P , Kirit Modi , Tarun Seera
114	NSPredicate	Brendon Roberto , Joshua , Mehul Chuahan
115	NSTimer	AJ9 , James P , Maddy`ヅ`ヅ` , Samuel Teferra , tfrank377 , That lazy iOS Guy , Undo , william205
116	NSURL	Adnan Aftab , ApolloSoftware , tharkay
117	NSURLConnection	byJeevan
118	NSURLSession	bluey31 , dasdom , dgatwood , Duly Kinsky , Harshal Bhavsar , Narendra Pandey , Otávio , R P , sage444 , Stephen Leppik
119	NSUserActivity	Samuel Spencer
120	NSUserDefaults	Anand Nimje , Emptyless , Harshal Bhavsar , Husein Behboodi Rad , J F , James P , Josh Caswell , Kirit Modi ,

		Mr. Xcoder , Roland Keesom , Seyyed Parsa Neshaei , user3760892 , william205
121	Objective-C Zugeordnete Objekte	Noam
122	Online-Bilder zwischenspeichern	Md. Ibrahim Hassan
123	OpenGL	Fonix
124	Parallelität	Doc , Fonix , Juan Campa , Kevin DiTraglia , Tien
125	PDF-Erstellung in iOS	Mansi Panchal , Narendra Pandey
126	plist iOS	SNarula
127	Profil mit Instrumenten	Vinod Kumar
128	QR Code Scanner	Bluewings , Efraim Weiss
129	Reich	subv3rsion
130	Rich Benachrichtigungen	Koushik
131	Safari-Dienstleistungen	Arnon Rodrigues , Harshal Bhavsar , Kilian Koeltzsch , Md. Ibrahim Hassan , Stephen Leppik
132	Schlüsselbund	abjurato , avojak , Matthew Seaman , Mehul Chuahan
133	Schlüsselwert-Schlüsselwertbeobachtung	D4ttatraya , Harshal Bhavsar , Mehul Chuahan , Mihriban Minaz , Mithrandir , Muhammad Zohaib Ehsan , Pärserk , sanman
134	Schneiden Sie einen UIImage in einen Kreis	Md. Ibrahim Hassan
135	Schnelle und Objective-C-Interoperabilität	Harshal Bhavsar , njuri , Stephen Leppik
136	Segues	Daniel Ormeño
137	Selektive UIView-Ecken abrunden	Md. Ibrahim Hassan
138	Sicherheit	D4ttatraya
139	Simulator	Seyyed Parsa Neshaei
140	Simulator-Builds	Durai Amuthan.H

141	SiriKit	Seyyed Parsa Neshaei
142	SLComposeViewController	Md. Ibrahim Hassan
143	Standort mit GPX-Dateien simulieren iOS	Uma
144	StoreKit	askielboe
145	Storyboard	Harshal Bhavsar , Kirit Vaghela , Stephen Leppik , Tommie C.
146	Swift: Ändern des rootViewControllers in AppDelegate, um den Haupt- oder Login- / Onboarding-Ablauf anzuzeigen	cleverbit
147	SWRevealViewController	Reinier Melian , tharkay
148	Tastatur verwalten	Alexander Tkachenko , Greg , Harshal Bhavsar , Mr. Xcoder , Richard Ash , Shog9 , slxl , Stephen Leppik , Steve Moser , Suragch , V1P3R , william205 , WMios
149	Überholspur	J F , KrauseFx , SM18 , tharkay
150	UIA Auftritt	azimov , Harshal Bhavsar , Stephen Leppik , Undo
151	UIActivityViewController	Amandeep , Harshal Bhavsar , Stephen Leppik , Vivek Molkar
152	UIAlertController	Andrii Chernenko , Arefly , Bhavin Ramani , FelixSFD , Harshal Bhavsar , Irfan , juliand665 , Kirit Modi , Muhammad Zohaib Ehsan , Narendra Pandey , Nikita Kurtin , NSNoob , pableiros , Senseful , Seyyed Parsa Neshaei , shim , Stephen Leppik , Sunil Sharma , Suragch , user3480295
153	UIBarButtonItem	Ahmed Khalaf , Dunja Lalic , hgwhittle , Suragch , william205
154	UIBezierPath	Bean , Igor Bidiniuc , Suragch , Teja Nandamuri
155	UIButton	Aleksei Minaev , Arefly , dasdom , ddb , Fabio Berger , FelixSFD , fredpi , James , James P , Jojodmo , Joshua , mattblessed , Mr. Xcoder , mtso , Nate Lee , NSNoob , P. Pawluś , Quantm , RamenChef , Roland Keesom , Sachin S P , tharkay , Viktor Simkó , william205 , WMios
156	UICollectionView	Adam Eberbach , AJ9 , Alex Koshy , Anand Nimje , Anh

		Pham, Bhavin Ramani, Bhumit Mehta, Brian, Dalija Prasnkar, ddb, Dima Deplov, Harshal Bhavsar, Kevin DiTraglia, Koushik, Mark, Rodrigo de Santiago, Stephen Leppik, Suragch, Undo
157	UIColor	Amanpreet, Anh Pham, Avineet Gupta, Brett Ponder, Cin316, Community, dasdom, DeyaEldeen, Douglas Hill, Elias Datler, Fabio Berger, FelixSFD, Gary Riches, Harshal Bhavsar, Honey, ing0, iphonic, Irfan, JAL, Jaleel Nazir, Jojodmo, Luca D'Alberti, maxkonovalov, mtso, nielsbot, NSNoob, pableiros, Reinier Melian, Rex, Sally, Samer Murad, Sandy, shim, The_Curry_Man, Tommie C., Viktor Simkó, WMios, Yagnesh Dobariya
158	UIControl - Ereignisbehandlung mit Blöcken	Brandon
159	UIDatePicker	Pavel Gatilov
160	UIDevice	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mehul Chuahan, Nef10, pableiros, Ramkumar chintala
161	UIFeedbackGenerator	beyowulf
162	UIFont	Mr. Xcoder
163	UIGestureRecognizer	Adam Preble, dannyzlo, Dunja Lalic, Harshal Bhavsar, John Leonardo, Josh Caswell, Md. Ibrahim Hassan, Ruby, Stephen Leppik, Sujania, Suragch, Undo
164	UIImage	Adrian Schönig, Alexander Tkachenko, Bean, Bhavin Ramani, Dipen Panchasara, Dunja Lalic, Emptyless, FelixSFD, Harshal Bhavsar, Heberti Almeida, Jimmy James, Mahmoud Adam, maxkonovalov, Md. Ibrahim Hassan, Muhammad Zeeshan, RamenChef, Reinier Melian, Rex, rob180, Ronak Chaniyara, sage444, Sandy, Seyyed Parsa Neshaei, Sujania, Sunil Sharma, The_Curry_Man, user3480295, Vineet Choudhary
165	UIImagePickerController	Brian, stonybrooklyn, william205
166	UIImageView	Adam Eberbach, Anh Pham, Bean, Caleb Kleveter, DeyaEldeen, Dunja Lalic, FelixSFD, il Malvagio Dottor Prosciutto, Irfan, Joshua, mattblessed, Md. Ibrahim Hassan, njuri, Quantm, Reinier Melian, Rex, Rob, Samuel Spencer, Sunil Sharma, Suragch, william205
167	UIKit Dynamics	beyowulf, Mark Stewart, Md. Ibrahim Hassan

168	UIKit Dynamics mit UICollectionView	beyowulf
169	UILabel	4oby , Akilan Arasu , Alex Koshy , alvarolopez , Andres Canella , Andrii Chernenko , Anh Pham , Ashwin Ramaswami , AstroCB , Barlow Tucker , bentford , Bhumit Mehta , Brian , byJeevan , Caleb Kleveter , Chathuranga Silva , Chris Brandsma , Cin316 , Code.Warrior , Community , Daniel Bocksteger , Daniel Stradowski , danshevluk , dasdom , ddb , DeyaEldeen , Dunja Lalic , Eric , Erwin , esthepiking , Fabio Berger , Fahim Parkar , Felix , FelixSFD , Franck Dernoncourt , gadu , ggrana , GingerHead , gvuksic , HaemEternal , hankide , Hans Sjunnesson , Harshal Bhavsar , Hossam Ghareeb , idobn , Imanou Petit , iOS BadBoy , iphonic , Irfan , J F , Jacky , Jacobanks , johnpenning , Jojodmo , Josh Brown , Joshua , Joshua J. McKinnon , jtbandes , juanjo , kabioberai , Kai Engelhardt , KANGKANG , Khanh Nguyen , Kireyin , leni , Luca D'Alberti , lufritz , Lukas , Luke Patterson , Lumialxk , Mad Burea , Mahmoud Adam , Md. Ibrahim Hassan , Moshe , Nadzeya , Narendra Pandey , Nathan Levitt , Nirav D , njuri , noelicus , NSNoob , Ollie , Quantm , Radagast the Brown , Rahul Vyas , RamenChef , ramsserio , rfarry , sage444 , Scotow , Seyyed Parsa Neshaei , Shahabuddin Vansiwala , solidcell , Sravan , stackptr , Sunil Sharma , Suragch , sushant jagtap , TDM , tharkay , The_Curry_Man , Tibor Molnár , Tyler , Undo , user3480295 , vasili111 , Vignan , Viktor Simkó , william205 , WMios , Yagnesh Dobariya
170	UILabel-Text unterstrichen	Md. Ibrahim Hassan
171	UILocalNotification	Bhumit Mehta , Brian , Byte1518 , D4ttatraya , David , ElonChan , Harshal Bhavsar , hgwhittle , kamwysoc , KrishnaCA , rajesh sukumaran , Rex , Samuel Spencer , themathsrobot , tksubota , william205 , Wolverine , Xenon
172	UINavigationController	dasdom , Oleh Zayats , sage444 , Suragch , william205 , WMios
173	UIPageViewController	azimov , Bright Future , Harshal Bhavsar , Mayuri R Talaviya , Stephen Leppik , stonbrooklyn , Victor M
174	UIPheonix - einfaches, flexibles, dynamisches und hoch skalierbares UI-Framework	StackUnderflow

175	UIPickerView	FelixSFD , Hasintha Janka , MCMatan , Md. Ibrahim Hassan , Moritz , NinjaDeveloper
176	UIRefreshControl UITableView	Md. Ibrahim Hassan , Mohammad Rana
177	UIScrollView	Bhavin Ramani , LinusGeffarth , maxkonovalov , Rex , sanman , Sujania , Sunil Sharma , Suragch , tharkay , torinpitchers
178	UIScrollView AutoLayout	Aaron , Brandon , Shrikant K
179	UIScrollView mit StackView-Kind	mourodrigo
180	UISearchController	Harshal Bhavsar , Mehul Chuahan , mtso , Stephen Leppik , Tarvo Mäesepp
181	UISegmentedControl	Kamil Harasimowicz
182	UISlider	Andreas , Md. Ibrahim Hassan
183	UISplitViewController	Cerbrus , Koushik
184	UIStackView	Anuj Joshi , danshevluk , Harshal Bhavsar , Kof , Lior Pollak , Sally , sasquatch , Stephen Leppik , william205
185	UIStackView dynamisch aktualisieren	Harshal Bhavsar , Rahul , Stephen Leppik
186	UIStoryboard	Adriana Carelli , Mr. Xcoder , Vignan
187	UISwitch	Bhavin Ramani , FelixSFD , Md. Ibrahim Hassan , Mr. Xcoder , RamenChef , Sujay
188	UITabBarController	Alexi , Anand Nimje , Cristina , Mehul Chuahan , Quantm , Srinija
189	UITableView	AJ9 , Alex Koshy , Andres Kievsky , Anh Pham , animuson , Bean , Brendon Roberto , Brian , dasdom , DeyaEldeen , Dima Deplov , Dunja Lalic , Erik Godard , Glorfindel , Harshal Bhavsar , Jojodmo , Kof , Luca D'Alberti , Luis , Meng Zhang , Nathan , Nirav Bhatt , Nirav D , RamenChef , Rex , RodolfoAntonici , Ruby , Samuel Spencer , Seslyn , simple_code , Srinija , Steve Moser , Sujania , Sujay , Suragch , Tamarous , user3480295
190	UITableViewCell	Rahul
191	UITableViewController	Aju

192	UI-Test	P. Pawluś
193	UITextField	Alex Koshy , Ali Elsokary , Ashvinkumar , Duly Kinsky , Fabio Berger , FelixSFD , J F , Joshua , Kof , Luiz Henrique Guimaraes , Maddy`ヰ`ヰ , P. Pawluś , RamenChef , Reinier Melian , Ruby , samwize , sasquatch , shim , SourabhV , Suragch , sushant jagtap , tharkay , william205 , WMios
194	UITextField-Delegat	Andreas , animuson , Md. Ibrahim Hassan , midori , Ruby
195	UITextView	Anh Pham , animuson , Bole Tzar , Bright Future , Cris , Dunja Lalic , Eonil , gadu , Harshal Bhavsar , Hejazi , Md. Ibrahim Hassan , njuri , Roland Keesom , Ruby , Suragch , sushant jagtap , william205 , WMios
196	UIView	Adam Preble , alaphao , Anh Pham , Caleb Kleveter , Community , Cory Wilhite , D4ttatraya , ddb , DeyaEldeen , Douglas Starnes , hgwhittle , iphonic , Irfan , James , Jojodmo , Jota , Kotha Sai Ram , Luca D'Alberti , maxkonovalov , Md. Ibrahim Hassan , muazhud , Narendra Pandey , Nikhil Manapure , NSNoob , pableiros , pckill , Peter DeWeese , Rahul Vyas , sasquatch , shallowThought , Sunil Sharma , That lazy iOS Guy , The_Curry_Man , Viktor Simkó , william205
197	UIViewController	dasdom , Dunja Lalic , shim , Suragch , tassinari , william205
198	UIWebView	Allan Burleson , dchar4life80X , iOS BadBoy , J F , Julian135 , KANGKANG , Kevin DiTraglia , maxkonovalov , Md. Ibrahim Hassan , Ortwin Gentz , Ramkumar chintala , Sunil Sharma
199	Umgang mit URL-Schemata	azimov , Brian , Dunja Lalic , Harshal Bhavsar , James P , Stephen Leppik
200	Universelle Links	Harshal Bhavsar , Irfan , satheeshwaran , Stephen Leppik , Vineet Choudhary
201	UUID (Universally Unique Identifier)	Anand Nimje , FelixSFD , Harshal Bhavsar , James P , Mehul Chuahan , Rahul Vyas , Seyyed Parsa Neshaei , shim , Stephen Leppik , sushant jagtap
202	Verwenden von Image Aseets	D4ttatraya
203	WCSessionDelegate	pkc456
204	WKWebView	Brandon , byJeevan , Mahmoud Adam , Yevhen Dubinin

205	Xcode Build & Archive von der Kommandozeile aus	Kyle Decot , Shardul
206	XCTest Framework - Unit Testing	D4ttatraya , dasdom , Jan ATAC , Josh Brown , msohng , Raphael Silva , Seyyed Parsa Neshaei , Tarun Seera
207	Zugänglichkeit	Harshal Bhavsar , Justin , Ruby , Stephen Leppik , Zev Eisenberg