Kostenloses eBook

LERNEN ionic2

Free unaffiliated eBook created from
Stack Overflow contributors.

#ionic2

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ionic2

It is an unofficial and free ionic2 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 ionic2.

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 ionic2

Bemerkungen

Ionic 2 ist eine plattformübergreifende mobile Entwicklungstechnologie. Dieses Framework ist für die Erstellung von mobilen Hybridanwendungen konzipiert und kann auch für Desktopanwendungen verwendet werden. Es ist ein einmal schreiben, laufen überall Technologie. Es verwendet Webtechnologien wie JavaScript / Typescript, Angular 2, HTML und CSS (SCSS / LESS). Ionic2-Apps funktionieren gut auf `>=android 4.4` , aber Sie möchten mit `android 4.1` bis `android 4.3` laufen und Cross Walk verwenden .

Examples

Installation oder Setup

Da Ionic 2 von Tag zu Tag besser wird, überprüfen Sie immer die **offizielle Dokumentation** , um die neuesten Änderungen und Verbesserungen zu verfolgen.

Voraussetzungen: Sie benötigen NodeJS, um Ionic 2-Projekte zu erstellen. Sie können Knoten herunterladen und installieren hier und mehr über npm lernen und die Pakete Ionic 2 verwendet hier .

---

1. Installieren von Ionic 2

Wie bei Ionic 1 können Sie die Ionic CLI oder GUI verwenden, um Apps direkt im Browser zu erstellen und zu testen. Es verfügt sogar über alle Funktionen, um mit Ihren Ionic 1-Apps zu arbeiten, sodass Sie nichts ändern müssen!

Um Ionic 2 zu verwenden, installieren Sie einfach ionic von npm:

```
$ npm install -g ionic
```

Wenn Sie einen EACCES-Fehler erhalten, befolgen Sie die Anweisungen hier , um dem Knoten die erforderlichen Berechtigungen zu erteilen.

2. Erstellen Sie Ihre erste App

Führen Sie nach der Installation der CLI den folgenden Befehl aus, um Ihre erste App zu starten:

```
$ ionic start MyIonic2Project
```

Die Tabs-Vorlage wird standardmäßig verwendet. Sie können jedoch eine andere Vorlage auswählen, indem Sie ein Flag übergeben. Zum Beispiel:

```
$ ionic start MyIonic2Project tutorial
$ cd MyIonic2Project
$ npm install
```

Dies wird die Tutorialvorlage verwenden .

Um Ihre App auszuführen, wechseln Sie in Ihr `ionic serve -lc` und führen Sie `ionic serve -lc` :

```
$ ionic serve -lc
```

Das -l aktiviert das Live-Reload der Seite, das -c zeigt die Konsolenprotokolle an. Wenn Sie Probleme beim Erstellen Ihrer App haben, stellen Sie sicher, dass Ihre package.json mit der in der ionic2-app-Basis übereinstimmt

Sie können mit Ihrer neuen App direkt im Browser spielen!

3. Aufbau auf einem Gerät

Sie können Ihre neue App auch auf einem physischen Gerät oder einem Geräteemulator erstellen. Sie brauchen Cordova, um fortzufahren.

Um Cordova zu installieren, führen Sie Folgendes aus:

```
$ npm install -g cordova
```

Lesen Sie die iOS-Simulatordokumente zum Erstellen von iOS-Anwendungen (HINWEIS: Sie können keine iOS-Geräte oder Emulatoren unter einem anderen Betriebssystem als OSX erstellen) oder die Genymotion- Dokumente zum Erstellen einer Android-Anwendung.

Läuft auf iOS-Gerät:

Um eine iOS-App zu erstellen, müssen Sie auf einem OSX-Computer arbeiten, da Sie das Kakao-Framework benötigen, um für ios erstellen zu können. In diesem Fall müssen Sie zuerst die Plattform zu Cordova hinzufügen, indem Sie die folgenden Befehl:

```
$ ionic cordova platform add ios
```

Sie benötigen Xcode , um ein iOS-Gerät zu kompilieren.

Führen Sie schließlich Ihre App mit dem folgenden Befehl aus:

```
$ ionic cordova run ios
```

Auf einem Android-Gerät ausführen:

Die Schritte für Android sind fast identisch. Fügen Sie zuerst die Plattform hinzu:

```
$ ionic cordova platform add android
```

Installieren Sie dann das Android SDK , mit dem Sie auf einem Android-Gerät kompilieren können. Obwohl das Android-SDK mit einem Emulator geliefert wird, ist es wirklich langsam. Genymotion ist viel schneller. Nach der Installation führen Sie einfach den folgenden Befehl aus:

```
$ ionic cordova run android
```

Und das ist es! Herzlichen Glückwunsch zum Bau Ihrer ersten Ionic 2-App!

Ionic hat auch Live-Nachladen. Wenn Sie also Ihre App entwickeln und Änderungen live auf dem Emulator / Gerät sehen möchten, können Sie dies mit den folgenden Befehlen ausführen:

Für iOS:

```
$ ionic cordova emulate ios -lcs
```

Seien Sie vorsichtig, unter iOS 9.2.2 funktioniert die Belastung nicht. Wenn Sie mit livereload arbeiten möchten, bearbeiten Sie die Datei config.xml, indem Sie Folgendes hinzufügen:

```
<allow-navigation href="*"/>
```

Dann in der `<platform name="ios">` :

```
<config-file parent="NSAppTransportSecurity" platform="ios" target="*-Info.plist">
  <dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
  </dict>
</config-file>
```

Für Android:

```
$ ionic cordova run android -lcs
```

Das `l` steht für live-reload, `c` für Konsolenprotokolle und `s` für Serverprotokolle. So können Sie während der Ausführung feststellen, ob Fehler / Warnungen vorliegen.

Bauen für Windows

Wenn Sie Ihr Projekt für Windows erstellen möchten, müssen Sie auf einem Windows-Computer arbeiten. Installieren Sie zunächst die Windows-Plattform in Ihrem ionic2-Projekt, indem Sie den folgenden Befehl ausführen:

```
$ionic cordova platform add windows
```

Dann führen Sie einfach den folgenden Befehl aus:

```
$ionic cordova run windows
```

Im Browser ausführen

```
$ionic serve
```

für Chrome-Browserinspektionsgerät (geben Sie die Adressleiste des Chrome-Browsers ein)

```
chrome://inspect/#devices
```

Erste Schritte mit ionic2 online lesen: https://riptutorial.com/de/ionic2/topic/3632/erste-schritte-mit-ionic2

Kapitel 2: Angularfire2 mit Ionic2

Einführung

Hier erfahren Sie, wie Sie AngularFire2 integrieren und diese Echtzeitdatenbank in unserer Ionic App verwenden.

Examples

AngularFire-Initialisierung

Zunächst müssen Sie die Winkelfeuer-Module in Ihrem App-Modul folgendermaßen initialisieren:

```
  const firebaseConfig = {
  apiKey: 'XXXXXXXXXX',
  authDomain: 'XXXXXXXXXX',
  databaseURL: 'XXXXXXXXXX',
  storageBucket: 'XXXXXXXXXX',
  messagingSenderId: 'XXXXXXXXXX'
};
```

Sie erhalten diese Schlüssel, indem Sie sich bei firebase anmelden und ein neues Projekt erstellen.

```
imports: [
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule,
    AngularFireAuthModule
  ],
```

AngularFire2 verwenden

Sobald Sie es in Ihrer App haben, importieren Sie es einfach:

```
import { AngularFireDatabase } from 'angularfire2/database';
constructor (private _af: AngularFireDatabase) {}
```

Mit dieser Beobachtungsliste können Sie auf eine Liste von Objekten unter einem Pfad zugreifen. Wenn Sie beispielsweise root / items / food haben, können Sie Nahrungsmittel wie diese erhalten:

```
this._af.list('root/items/food');
```

Sie können einfach ein neues Element hier einfügen und werden in Ihrer Firebase-Datenbank angezeigt, oder Sie können ein Element aktualisieren und es wird in Ihrer Datenbank aktualisiert. Sie können wie folgt pushen und aktualisieren:

```
this._af.list('root/items/food').push(myItemData);
```

```
this._af.list('root/items/food').update(myItem.$key, myNewItemData);
```

Oder Sie können sogar Gegenstände aus Ihrer Nahrungsliste entfernen:

```
this._af.list('root/items/food').remove(myItem.$key);
```

Angularfire2 mit Ionic2 online lesen: https://riptutorial.com/de/ionic2/topic/10918/angularfire2-mit-ionic2

Kapitel 3: Dienste verwenden

Bemerkungen

Eine sehr wichtige Sache bei der Verwendung gemeinsam genutzter Dienste ist, dass sie im `providers` Array der obersten Komponente enthalten sein müssen, in der sie freigegeben werden müssen.

Warum das? Nehmen wir an, wir nehmen den `MyService` Verweis von jeder `Component` in das `providers` Array auf. So etwas wie:

```
@Component({
  templateUrl:"page1.html",
  providers: [MyService]
})
```

Und

```
@Component({
  templateUrl:"page2.html",
  providers: [MyService]
})
```

Auf diese Weise **wird für jede Komponente eine neue Instanz des Diensts erstellt.** Die Instanz, in der eine Seite die Daten speichert, unterscheidet sich von der Instanz, die zum Abrufen der Daten verwendet wird. Das geht also nicht.

Damit die gesamte App dieselbe Instanz verwendet (damit der Dienst als *Singleton*- Dienst funktioniert), können wir die Referenz in der `App Component` hinzufügen:

```
@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>',
  providers: [MyService]
})
```

Sie können den `MyService` Verweis auch in `ionicBootstrap(MyApp, [MyService]);` aber entsprechend Angular2 Style Guides

> Bieten Sie dem Angular 2-Injektor an der obersten Komponente Dienste an, wo sie gemeinsam genutzt werden.
>
> Warum? Der Angular 2 Injektor ist hierarchisch.
>
> Warum? Bei der Bereitstellung des Dienstes für eine Komponente der obersten Ebene wird diese Instanz gemeinsam genutzt und steht allen untergeordneten Komponenten dieser Komponente der obersten Ebene zur Verfügung.
>
> Warum? Dies ist ideal, wenn ein Dienst Methoden oder Status freigibt.

> Warum? Dies ist nicht ideal, wenn zwei verschiedene Komponenten unterschiedliche Instanzen eines Dienstes benötigen. In diesem Szenario ist es besser, den Service auf der Komponentenebene bereitzustellen, die die neue und separate Instanz benötigt.

Und

> Es wird klappen. Es ist einfach keine Best Practice. **Die Bootstrap-Provider-Option dient zum Konfigurieren und Überschreiben der vorregistrierten Dienste von Angular** , z. B. der Routing-Unterstützung.

... die `App Component` wäre die beste Wahl.

Examples

Teilen Sie Informationen zwischen verschiedenen Seiten

Eines der einfachsten Beispiele für die Verwendung *gemeinsam genutzter Dienste* ist, wenn wir einige Daten von einer bestimmten Seite unserer Anwendung speichern möchten und diese Daten dann aber wieder von einer anderen Seite erhalten möchten.

Eine Option könnte darin bestehen, diese Daten als Parameter zu senden (z. B. wenn eine Seite die andere Seite aufruft). Wenn wir diese Daten jedoch aus einem völlig anderen Teil der Anwendung verwenden möchten, scheint dies nicht die beste Methode zu sein es. Dann kommen *Shared Services* ins Spiel.

In diesem Beispiel verwenden wir einen einfachen Dienst namens `MyService` der nur zwei einfache Methoden hat: `saveMessage()` , um einen String zu speichern, und `getMessage()` , um ihn erneut `getMessage()` . Dieser Code ist Teil dieses funktionierenden Plunkers, wo Sie ihn in Aktion sehen können.

```
import {Injectable} from '@angular/core';

@Injectable()
export class MyService {

  private message: string;

  constructor(){ }

  public saveMessage(theMessage: string): void {
    this.message = theMessage;
  }

  public getMessage(): string {
    return this.message;
  }
}
```

Wenn wir dann eine neue Nachricht speichern möchten, können wir einfach die `saveMessage(theMessageWeWantToSave);` Methode aus der `MyService` Instanz (nur als `service`).

```
import { Component } from "@angular/core";
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  message: string;

  // ...

  public saveSecretMessage(): void {
    this.service.saveMessage(this.message);
  }
}
```

Auf dieselbe Weise können wir, wenn wir diese Daten `getMessage()` möchten, die Methode `getMessage()` von der `getMessage()` wie `getMessage()` :

```
import { Component } from "@angular/core";
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page2.html"
})
export class Page2 {

  enteredMessage: string;

  constructor(private service: MyService) {
    this.enteredMessage = this.service.getMessage();
  }

  // ...
}
```

Bitte beachten Sie den Abschnitt *Hinweise* zu überprüfen , um zu sehen , wo sollte die Referenz für den `MyService` Dienst aufgenommen werden und warum.

Dienste verwenden online lesen: https://riptutorial.com/de/ionic2/topic/4407/dienste-verwenden

Kapitel 4: Geolocation

Examples

Einfache Benutzung

`package.json` sicher, dass Sie die Abhängigkeiten in Ihre `package.json` :

```
{
    ...
    "dependencies": {
        ...
        "ionic-native": "^1.3.10",
        ...
    },
    ...
}
```

Geolocation verwenden:

```
// custom-component.ts

import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
    selector: 'custom-component',
    template: template
})
export class CustomComponent {

    constructor() {

        // get the geolocation through a promise
        Geolocation.getCurrentPosition().then((position:Geoposition)=> {
            console.log(
                position.coords.latitude,
                position.coords.longitude);
        });
    }
}
```

Die Position beobachten

Für eine Echtzeitlösung können Sie die watchPosition-Funktion in Geolocation verwenden, die benachrichtigt, wenn ein Fehler oder eine Positionsänderung auftritt. Im Gegensatz zu getCurrentPosition gibt watchPosition ein Observable zurück

```
import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
```

```
selector: 'custom-component',
template: template
})
export class CustomComponent {
constructor() {

    // get the geolocation through an observable
        Geolocation.watchPosition(<GeolocationOptions>{
            maximumAge: 5000, // a maximum age of cache is 5 seconds
            timeout: 10000, // time out after 10 seconds
            enableHighAccuracy: true // high accuracy
        }).subscribe((position) => {
            console.log('Time:' + position.timestamp);
            console.log(
                'Position:' + position.coords.latitude + ',' +
                position.coords.longitude);
            console.log('Direction:' position.coords.heading);
            console.log('Speed:' position.coords.speed);

        });
}
```

Geolocation online lesen: https://riptutorial.com/de/ionic2/topic/5840/geolocation

Kapitel 5: Hinzufügen einer ionischen App zur ionischen Ansicht

Einführung

ionic view ist eine mobile App, die Sie auf Ihrem Handy installieren müssen, damit Sie Ihre App anzeigen können, ohne APK-Dateien zu erstellen. Indem Sie Ihre App-ID freigeben, können andere Benutzer Ihre App auch mithilfe der Ionic-Ansicht auf ihrem Mobilgerät anzeigen.

Site: https://view.ionic.io/

Examples

Schritte zum Hinzufügen Ihrer App zur Ionenansicht

Die folgenden Schritte müssen in app.ionic.io ausgeführt werden

1. Erstellen Sie ein Konto oder melden Sie sich bei Ihrem Ionic-Konto an
2. Klicken Sie im Dashboard auf "Neue App" und geben Sie den Namen für Ihre App an

   ```
   I named my app as 'MyIonicApp'
   ```

3. Im Übersichtsbereich dieser neu erstellten App befindet sich unter dem App-Namen eine ID.

   ```
   MyIonicApp ID is 4c5051c1
   ```

Die folgenden Schritte werden in der Eingabeaufforderung **Node.js** ausgeführt

1. Melden Sie sich in Ihrem ionischen Konto an, indem Sie ausführen

   ```
   $ ionic login
   ```

2. Wurzeln Sie Ihren App-Ordner.
3. Um Ihre App in die ionische Ansicht hochzuladen, müssen Sie Ihre App zunächst mit der ID verknüpfen, die Sie in ionic site erstellt haben. Führen Sie den folgenden Befehl aus, um eine Verknüpfung herzustellen.

   ```
   $ ionic link [your-app-id]
   ```

   Für MyIoincApp lautet der Befehl:

   ```
   $ ionic link 4c5051c1
   ```

Der obige Befehl aktualisiert die App-ID in der Konfigurationsdatei von MyIonicApp.

4. Sobald die Verknüpfung fertig ist, laden Sie die App hoch, indem Sie sie ausführen

```
$ ionic upload
```

Hinweis

Wenn der Upload erfolgreich ist, öffnen Sie die ionische Ansicht in Ihrem Handy, um die App anzuzeigen.

Andere können Ihre App anzeigen, indem Sie die App-ID unter "Vorschau einer App" in der Ionenansicht senden.

Hinzufügen einer ionischen App zur ionischen Ansicht online lesen:
https://riptutorial.com/de/ionic2/topic/10542/hinzufugen-einer-ionischen-app-zur-ionischen-ansicht

Kapitel 6: InAppBrowser

Einführung

Manchmal muss der Client nur eine Web-App in der mobilen App öffnen. Dazu können wir InAppBrowser so verwenden, dass es wie eine App aussieht. Stattdessen öffnen wir eine Website / WebApp in Mobile, sobald der Benutzer auf das App-Symbol tippt Anstatt die erste App-Ansicht zu öffnen, können Sie InAppBrowser direkt öffnen.

Examples

Ein Live-Beispiel für diese Verwendung ist diese App:

In dieser App öffne ich direkt InAppBrowser, wenn der Benutzer auf das App-Symbol tippt, anstatt die erste Seite der App zu laden. Das würde also für den Benutzer so aussehen, dass er die App derselben Website / Webapp anzeigt.

Codebeispiel zur Verwendung von InAppBrowser

```
  platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      var url= "https://blog.knoldus.com/";
      var browserRef = window.cordova.InAppBrowser.open(url, "_self", "location=no",
"toolbar=no");
      browserRef.addEventListener("exit", (event) => {
          return navigator["app"].exitApp();
        }
      );
```

InAppBrowser online lesen: https://riptutorial.com/de/ionic2/topic/9801/inappbrowser

Kapitel 7: Ionic2-CSS-Komponenten

Examples

Gitter

Das Rastersystem von Ionic basiert auf Flexbox, einer CSS-Funktion, die von allen von Ionic unterstützten Geräten unterstützt wird. Das Gitter besteht aus drei Einheiten-Gittern, Zeilen und Spalten. Spalten werden erweitert, um ihre Zeile zu füllen, und die Größe wird für zusätzliche Spalten angepasst.

Klasse	Breite
Breite-10	10%
Breite-20	20%
Breite-25	25%
Breite-33	33.3333%
Breite-50	50%
Breite-67	66.6666%
Breite-75	75%
Breite-80	80%
Breite-90	90%

Beispiel.

```
    <ion-grid>
      <ion-row>
        <ion-col width-10>This column will take 10% of space</ion-col>
      </ion-row>
    </ion-grid>
```

Karten

Karten sind eine großartige Möglichkeit, wichtige Inhalte anzuzeigen und entwickeln sich schnell zu einem zentralen Entwurfsmuster für Apps. Sie sind eine großartige Möglichkeit, Informationen zu enthalten und zu organisieren, und stellen gleichzeitig vorhersehbare Erwartungen für den Benutzer auf. Mit so vielen Inhalten, die auf einmal angezeigt werden können, und oft so wenig Bildschirmfläche, sind Karten für viele Unternehmen schnell zum Muster der Wahl geworden.

Beispiel.

```
<ion-card>
  <ion-card-header>
        Header
  </ion-card-header>
  <ion-card-content>
        The British use the term "header", but the American term "head-shot" the English
simply refuse to adopt.
  </ion-card-content>
</ion-card>
```

Ionic2-CSS-Komponenten online lesen: https://riptutorial.com/de/ionic2/topic/8011/ionic2-css-komponenten

Kapitel 8: Konstruktor und OnInit

Einführung

In Bezug auf ionic2 der `constructor` : In einfachen Worten erstellen wir zum Beispiel Instanzen unserer Plugins, Services usw.: Sie haben eine Seite (Ansicht), auf der Sie die Liste aller Schüler anzeigen möchten, und Sie haben eine Json-Datei Das enthält alle Schüler (diese Datei ist Ihre Datendatei). Sie müssen in diesem Dienst einen Dienst erstellen. Sie erstellen eine Methode und klicken auf eine http.get-Anforderung, um die Json-Daten abzurufen. Was brauchen Sie also? http geht einfach so:

Examples

Student Service Method-Beispiel für die Verwendung von Http im Konstruktor

```
import {Http} from '@angular/http';
@Injectable()
export class StudentService{
    constructor(public http: Http){}
    getAllStudents(): Observable<Students[]>{
        return this.http.get('assets/students.json')
        .map(res => res.json().data)
        }
    }
```

Beachten Sie den Konstruktor jetzt wieder, wenn Sie diese Servicemethode verwenden möchten, gehen wir zu unserer Ansicht / Seite und:

```
import {StudentService} from './student.service';
import { SocialSharing } from '@ionic-native/social-sharing';
export class HomePage implements OnInit {

  constructor(public _studentService: StudentService, public socialSharing: SocialSharing) {
   }
```

Beachten Sie auch hier den Konstruktor, wir erstellen eine Instanz von StudentService in Konstruktor und noch etwas: Wir verwenden das socialSharing-Plugin, um es so zu verwenden, dass wir auch eine Instanz des Konstruktors erstellen.

ngOnInit-Methode, um die Liste der Schüler beim Laden der Ansicht abzurufen

`OnInit` : Das ist wirklich erstaunlich in ionic2 oder man kann es in AngularJs2 sagen. Mit dem gleichen Beispiel können wir sehen, was ngOnInit ist. Damit Sie mit der Servicemethode fertig sind, möchten Sie nun in Ihrer Ansicht / Seite, dass die Schülerlistendaten verfügbar sind, sobald Ihre Ansicht angezeigt wird. Dies sollte die erste Operation sein, die automatisch beim Laden ausgeführt wird, da die Ansicht den Schüler lädt Liste sollte sichtbar sein. Die Klasse

implementiert also OnInit und Sie definieren ngOnInit. Beispiel:

ngOnInit-Beispiel, um die Liste der Schüler auf Seite / Ansicht anzuzeigen

```
export class HomePage implements OnInit {
...
....
constructor(....){}

ngOnInit(){
    this._studentService.getAllStudents().subscribe(
     (students: Students[]) => this.students = students,
    )
```

Konstruktor und OnInit online lesen: https://riptutorial.com/de/ionic2/topic/9907/konstruktor-und-oninit

Kapitel 9: Modals

Examples

Modals verwenden

Modals werden auf dem Bildschirm angezeigt, um eine temporäre Benutzeroberfläche anzuzeigen, die häufig für Anmelde- oder Anmeldeseiten, Nachrichtenzusammensetzung und Auswahl von Optionen verwendet wird.

```
import { ModalController } from 'ionic-angular';
import { ModalPage } from './modal-page';

export class MyPage {
  constructor(public modalCtrl: ModalController) {
  }

  presentModal() {
    let modal = this.modalCtrl.create(ModalPage);
    modal.present();
  }
}
```

HINWEIS: Ein Modal ist ein Inhaltsbereich, der über die aktuelle Seite des Benutzers geleitet wird.

Daten durch einen Modal weiterleiten

Daten können über `Modal.create()` als zweites Argument an ein neues Modal `Modal.create()` . Auf die Daten kann dann von der geöffneten Seite aus durch `NavParams` von `NavParams` . Beachten Sie, dass die Seite, die als Modal geöffnet wurde, keine spezielle "Modal" `NavParams` , `NavParams` nicht anders als eine Standardseite verwendet.

Erste Seite:

```
import { ModalController, NavParams } from 'ionic-angular';

export class HomePage {

 constructor(public modalCtrl: ModalController) {

 }

 presentProfileModal() {
   let profileModal = this.modalCtrl.create(Profile, { userId: 8675309 });
   profileModal.present();
 }

}
```

Zweite Seite:

```
import { NavParams } from 'ionic-angular';
export class Profile {

 constructor(params: NavParams) {
   console.log('UserId', params.get('userId'));
 }

}
```

Modals online lesen: https://riptutorial.com/de/ionic2/topic/6415/modals

Kapitel 10: Modals

Examples

Einfach Modal

Modal ist eine temporäre Benutzeroberfläche, die über Ihrer aktuellen Seite angezeigt wird. Dies wird häufig zum Anmelden, Anmelden, Bearbeiten vorhandener Optionen und Auswählen von Optionen verwendet.

Schauen wir uns ein einfaches Beispiel mit den verwendeten Modalen an. Zunächst erstellen wir ein Projekt für ionische Leere. Lassen Sie uns ein einfaches Modal erstellen, in dem eine Nachricht angezeigt wird, und beenden Sie das Klicken auf die Schaltfläche. Zu diesem Zweck erstellen wir eine Ansicht für unser Modal.

Message.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Modal
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h1>Modal Without Params is created successfully.</h1>
  <button full (click)="dismiss()"> Exit </button>
</ion-content>
```

Message.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/message/message.html',
})
export class MessagePage {
  viewCtrl;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

In diesem Modal wird eine Nachricht angezeigt. Die Modal kann mit Hilfe des View - Controller **entlassen** Verfahren geschlossen oder „entlassen“ werden.

Home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Modal Example
    </ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <button full (click)="openModal()">ModalWithoutParams-Message</button>
</ion-content>
```

Home.ts

```
import { Component } from '@angular/core';
import { ModalController } from 'ionic-angular';
import {MessagePage} from '../message/message';
@Component({
  templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
  modalCtrl;
  data;
  constructor(modalCtrl: ModalController) {
    this.modalCtrl = modalCtrl;
    this.data = [{name: "aaa", email: "aaa.a@som.com", mobile: "1234567890", nickname: "zzz"},
      {name: "bbb", email: "bbb.a@som.com", mobile: "1234567890", nickname: "yyy"},
      {name: "ccc", email: "ccc.a@som.com", mobile: "1234567890", nickname: "xxx"}]
  }
  openModal() {
    let myModal = this.modalCtrl.create(MessagePage);
    myModal.present();
  }
}
```

Jetzt erstellen wir unsere Homepage, auf der der **ModalController** und unser Datenmodell MessagePage importiert werden. Die Methode **create von** ModalController erstellt modal für unser Datenmodell MessagePage, das zur Steuerung der Variablen myModal gespeichert wird. **Die** aktuelle Methode öffnet das Modal oben auf unserer aktuellen Seite.

Modal mit Parametern bei Entlassung:

Wir wissen jetzt, wie man einen Modal erstellt. Was aber, wenn wir einige Daten von Modal auf unsere Homepage übertragen möchten. Lassen Sie uns dazu ein Beispiel mit Modal als Register-Seitenübergabeparameter an die übergeordnete Seite betrachten.

Register.html

```
<ion-header>
  <ion-toolbar>
```

```
    <ion-title>
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <ion-item>
      <ion-label>Name</ion-label>
      <ion-input type="text" [(ngModel)]="name"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" [(ngModel)]="email"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" [(ngModel)]="mobile"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" [(ngModel)]="nickname"></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="add()">Add</button>
</ion-content>
```

Register.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/register/register.html',
})
export class ResisterPage {
  viewCtrl;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
    this.name = "";
    this.email = "";
    this.mobile = "";
    this.nickname = "";
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
  add(){
    let data = {"name": this.name, "email": this.email, "mobile": this.mobile, "nickname":
this.nickname};
    this.viewCtrl.dismiss(data);
```

```
  }
}
```

Register modal ruft ein Datenobjekt mit vom Benutzer eingegebenen Werten ab, und die Parameter werden an unsere aktuelle Seite übergeben, wenn die ViewControllers-Methode verworfen wird. Nun werden die Parameter gesendet.

Wie werden die Parameter auf der Startseite abgerufen? Dazu erstellen wir eine Schaltfläche auf der Startseite und rufen bei Klick das Register modal auf. Um den Benutzer anzuzeigen, zeigen wir eine Liste an.

Home.html

```
<ion-list>
  <ion-item *ngFor="let datum of data">
    <h1>{{datum.name}}</h1>
  </ion-item>
</ion-list>
<button full secondary (click)="openModalParams()">ModalWithParams-Register</button>
```

Home.ts

```
import {ResisterPage} from '../register/register';

  openModalParams(){
    let modalWithParams = this.modalCtrl.create(ResisterPage);
    modalWithParams.present();

    modalWithParams.onDidDismiss((result) =>{
      if(result){
        this.data.unshift(result);
      }
    });
  }
```

Die **onDidDismiss-** Methode von **ViewController** wird ausgeführt, wenn ein Modal geschlossen wird. Wenn Daten von modal als Parameter übergeben werden, können wir sie mit der Methode onDidDismiss abrufen. Hier werden die vom Benutzer eingegebenen Daten an die vorhandenen Daten angehängt. Wenn keine Daten als Parameter übergeben werden, ist der zurückgegebene Wert null.

Modal mit Parametern beim Erstellen:

Die Übergabe von Parametern an einen Modal ähnelt der Übergabe von Werten an einen NavController. Dazu ändern wir unsere Liste in home.html, um beim Anklicken eines Listenelements eine Modalität zu öffnen und die erforderlichen Parameter als zweites Argument an die **create-** Methode zu übergeben.

Home.html

```
 <ion-list>
```

```
    <ion-item *ngFor="let datum of data" (click)="openModalwithNavParams(datum)">
      <h1>{{datum.name}}</h1>
    </ion-item>
  </ion-list>
```

Home.ts

```
import {EditProfilePage} from '../edit-profile/edit-profile';

  openModalwithNavParams(data){
    let modalWithNavParams = this.modalCtrl.create(EditProfilePage,{Data: data});
    modalWithNavParams.present();
  }
```

Ähnlich wie bei anderen Ansichten verwenden wir NavParams, um die aus der vorherigen Ansicht gesendeten Daten abzurufen.

Edit-Profile.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h2>Welcome {{name}}</h2>
  <ion-list>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" value={{email}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" value={{mobile}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" value={{nickname}}></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="dismiss()">Close</button>
</ion-content>
```

Edit-Profile.ts

```
import { Component } from '@angular/core';
import { ViewController, NavParams } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/edit-profile/edit-profile.html',
```

```
})
export class EditProfilePage {
  viewCtrl;
  navParams;
  data;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController, navParams: NavParams) {
    this.viewCtrl = viewCtrl;
    this.navParams = navParams;
    this.data = this.navParams.get('Data');
    this.name = this.data.name;
    this.email = this.data.email;
    this.mobile = this.data.mobile;
    this.nickname = this.data.nickname;

  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

Modals online lesen: https://riptutorial.com/de/ionic2/topic/6612/modals

Kapitel 11: Problemumgehung für 'show-delete' in Missbilligung

Examples

Lösung

Ich entwickle eine mobile App mit Ionic 2 mit Angular 2.

Ich habe eine Liste mit Ionenlisten gefüllt. Ich möchte, dass diese Ionenelemente bei Bedarf gelöscht werden, wie hier auf der ionischen Website dargestellt.

Allerdings haben viele in **ionischen 2** seit der ersten Version und der oben Stil einer Taste geändert all **Ionen Elemente** an einer Öffnung ist nicht mehr möglich , da die **Show-löschen** und **Show-Neuordnungs** werden nicht mehr unterstützt. Die einzige verfügbare Option ist das **Ionenartikelgleiten** als Ionenartikel. Dies gibt uns die Möglichkeit, jeden Artikel einzeln zu verschieben, um die Schaltfläche "Löschen" anzuzeigen.

Das wollte ich nicht. Ich wollte eine Schaltfläche, mit der alle Ionenelemente gleichzeitig geöffnet werden.

Nachdem ich einige Zeit damit verbracht hatte, habe ich eine funktionierende Lösung gefunden und mit ionic 2 das gewünschte Ergebnis erzielt. Ich werde es mit Ihnen teilen.

Hier ist meine Lösung:

In der .html-Datei:

```
<ion-header>
  <ion-navbar>
    <ion-buttons start (click)="manageSlide()">
      <button>
        <ion-icon name="ios-remove"></ion-icon>
      </button>
    </ion-buttons>
    <ion-title>PageName</ion-title>
  </ion-navbar>
</ion-header>
```

und für die Liste:

```
<ion-list #list1>
  <ion-item-sliding #slidingItem *ngFor="let contact of contacts | sortOrder">
    <button #item ion-item>
      <p>{{ item.details }}</p>
      <ion-icon id="listIcon" name="arrow-forward" item-right></ion-icon>
    </button>
    <ion-item-options side="left">
      <button danger (click)="doConfirm(contact, slidingItem)">
```

```
        <ion-icon name="ios-remove-circle-outline"></ion-icon>
      Remove
      </button>
    </ion-item-options>
  </ion-item-sliding>
</ion-list>
```

Führen Sie in der **.ts-** Datei zuerst Ihre Importe aus:

```
import { ViewChild } from '@angular/core';
import { Item } from 'ionic-angular';
import { ItemSliding, List } from 'ionic-angular';
```

Verweisen Sie dann auf das HTML-Element, indem Sie ein ViewChild deklarieren:

```
@ViewChild(List) list: List;
```

Fügen Sie schließlich Ihre Klassen hinzu, um die Arbeit zu erledigen:

```
public manageSlide() {

    //loop through the list by the number retreived of the number of ion-item-sliding in the
list
    for (let i = 0; i < this.list.getElementRef().nativeElement.children.length; i++) {

        // retreive the current ion-item-sliding
        let itemSlide = this.list.getElementRef().nativeElement.children[i].$ionComponent;

        // retreive the button to slide within the ion-item-sliding
        let item = itemSlide.item;

        // retreive the icon
        let ic = item._elementRef.nativeElement.children[0].children[1];

        if (this.deleteOpened) {
            this.closeSlide(itemSlide);
        } else {
            this.openSlide(itemSlide, item, ic);
        }
    }

    if (this.deleteOpened) {
        this.deleteOpened = false;
    } else {
        this.deleteOpened = true;
    }
}
```

Dann der Eröffnungskurs:

```
private openSlide(itemSlide: ItemSliding, item: Item, inIcon) {
  itemSlide.setCssClass("active-sliding", true);
  itemSlide.setCssClass("active-slide", true);
  itemSlide.setCssClass("active-options-left", true);
  item.setCssStyle("transform", "translate3d(72px, 0px, 0px)")
}
```

Und die Abschlussklasse:

```
private closeSlide(itemSlide: ItemSliding) {
  itemSlide.close();
  itemSlide.setCssClass("active-sliding", false);
  itemSlide.setCssClass("active-slide", false);
  itemSlide.setCssClass("active-options-left", false);
```

}

Ich hoffe, es wird einigen da draußen helfen.

Viel Spaß und gute Programmierung ...

Problemumgehung für 'show-delete' in Missbilligung online lesen: https://riptutorial.com/de/ionic2/topic/6620/problemumgehung-fur--show-delete--in--ion-list--missbilligung

Kapitel 12: Push-Benachrichtigung gesendet und empfangen

Bemerkungen

Die im Initialisierungsbeispiel vorhandene SenderID ist eine gcm-Absender-ID, die Sie von Google erhalten. Es sollte auch vorhanden sein, wenn Sie das Plugin installieren

```
ionic plugin add phonegap-plugin-push --variable SENDER_ID="XXXXXXX"
```

Wenn Sie Ihren Push-Benachrichtigungen weitere Daten hinzufügen möchten, lesen Sie diesen Link, um weitere Typisierungen hinzuzufügen: https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/TYPESCRIPT.md

Examples

Initialisierung

Das Push-Benachrichtigungs-Plugin erfordert eine Init-Initialisierung, die das Plugin mit der angegebenen Absender-ID anweist, zu laufen.

```
 let push = Push.init({
     android: {
       senderID: "------------",
     },
     ios: {
       alert: "true",
       badge: true,
       sound: "false",
     },
     windows: {},
   });
```

Anmeldung

Der Registrierungsschritt registriert die App beim System des Geräts und gibt eine Registrierungs-ID zurück

```
import { Push, RegistrationEventResponse} from "ionic-native";

       //the push element is created in the initialization example
       push.on("registration", async (response: RegistrationEventResponse) => {
                   //The registration returns an id of the registration on your device
                   RegisterWithWebApi(response.registrationId);

       });
```

Empfangen einer Push-Benachrichtigung

Um Push-Benachrichtigungen zu erhalten, müssen wir dem Plugin mitteilen, eingehende Push-Benachrichtigungen abzuhören. Dieser Schritt wird nach der Initialisierung und Registrierung durchgeführt

```
import { Push, NotificationEventResponse} from "ionic-native";

        //the push element is created in the initialization example
        push.on("notification", (response: NotificationEventResponse) => {
            let chatMessage: ChatMessage = <ChatMessage>{
              title: response.title,
              message: response.message,
              receiver: response.additionalData.replyTo,
              image: response.image
            };
            DoStuff(chatMessage));
       });
```

Push-Benachrichtigung gesendet und empfangen online lesen:
https://riptutorial.com/de/ionic2/topic/5874/push-benachrichtigung-gesendet-und-empfangen

Kapitel 13: Setup und Debugging von Ionic 2 in Visual Studio Code

Einführung

Visual Studio ist eine Open Source IDE, die Intellisense- und Bearbeitungsfunktionen für Code bietet. Diese IDE unterstützt viele Sprachen wie (Ionic, C, C #, AngularJs, TypeScript, Android usw.). Diese Sprachen können ihren Code ausführen, indem sie ihre Erweiterungen in VSCode hinzufügen. Mit VSCode können wir den Code verschiedener Sprachen ausführen und debuggen.

Examples

Installation von VSCode

Zunächst müssen Sie den VSCode herunterladen und installieren. Diese neueste Version von VSCode steht auf der offiziellen Website zum Download zur Verfügung. Nach dem Download des VSCode sollten Sie ihn installieren und öffnen.

```
Introduction of Extensions in VSCode
```

VSCode ist ein offener Editor, also Editor für alle Sprachen. Um einen Code auszuführen, müssen Sie die Erweiterung für diese bestimmte Sprache hinzufügen. Um Ihren ionischen Code **auszuführen** und zu bearbeiten, sollten Sie in **Ihrem VSCode** die **ionic2-vscode-** Erweiterung hinzufügen. Auf der linken Seite des VSCode-Editors befinden sich 5 Symbole, in denen das unterste Symbol für die Erweiterung verwendet wird. Die Erweiterungen erhalten Sie mit der **Tastenkombination (Strg + Umschalttaste + X)** .

```
Add Extension for Ionic2 in VsCode
```

Durch Drücken von **Strg + Umschalttaste + X haben** Sie den Teil der Erweiterung angezeigt, an dem oben **drei Punkte** angezeigt werden **.** Diese Punkte werden als mehr Symbole bezeichnet. Wenn Sie darauf klicken, wird ein Dialogfeld geöffnet, in dem die Anzahl der Optionen angezeigt wird Wählen Sie die gewünschte Option. Um jedoch alle Erweiterungen zu erhalten, wählen Sie die Option **Empfohlene Erweiterung** anzeigen. In der Liste aller Erweiterungen, in denen Sie die **Erweiterung** installieren können `(ionic2-vscode),npm`

Erstellen und fügen Sie Ihr Ionenprojekt in VSCode hinzu

VsCode ist nicht in der Lage , das ionische Projekt zu erstellen , weil es ein Code ist editor.So Sie Ihr ionisches Projekt von **CLI** oder **cmd** erstellen können. Erstellen Sie Ihr Projekt mit dem folgenden Befehl

```
$ ionic start appName blank
```

Mit dem obigen Befehl können Sie eine leere Vorlagenanwendung erstellen. Ionic2 bietet drei Arten von Vorlagen **, Registerkarten und Seitenmenüs** . So. Sie können die leere Vorlage durch zwei andere Vorlagen ersetzen.

Nun ist Ihr Ionenprojekt erstellt worden. So können Sie Ihr Projekt in VSCode zur Bearbeitung hinzufügen. Um Ihr Projekt hinzuzufügen, folgen Sie den folgenden Punkten.

1. Gehen Sie Menü in VScode **Datei.**
2. Klicken Sie im Menü Datei auf den **Ordner öffnen** .
3. Suchen und öffnen Sie Ihren Projektordner.

   Sie können den Ordner mit der Tastenkombination **Strg + O** oder **Strg + K** direkt öffnen

`

Führen Sie Ihr Ionenprojekt aus und debuggen Sie es

> In Chrome ausführen und debuggen

Um das ionische Projekt **auszuführen** , verwenden Sie den folgenden Befehl in **terminal oder cmd oder CLI**

```
$ ionic serve
```

Um das ionische Projekt zu **debuggen** , sollten Sie zuerst die Erweiterung **(Debugger für Chrome)** hinzufügen und anschließend die Datei launch.json so konfigurieren.

```
{
        "version": "0.2.0",
        "configurations": [

            {
          "name": "Launch in Chrome",
          "type": "chrome",
          "request": "launch",
          "url": "http://localhost:8100",
          "sourceMaps": true,
          "webRoot": "${workspaceRoot}/src"
        }
]
}
```

> ***In Android ausführen und debuggen***

Zum Ausführen eines ionischen Projekts in Android sollten Sie die Android-Plattform mit dem folgenden Befehl in terminal oder cmd oder CLI hinzufügen:

```
$ ionic cordova platform add android
```

Erstellen Sie Android mit diesem Befehl

```
$ ionic cordova build android
```

Befehl für Android-Plattform ausführen

```
$ ionic cordova run android
```

Jetzt läuft Ihre Anwendung auf einem echten Android-Gerät.

Zum Debuggen in Android-Geräten müssen Sie **Cordova oder Android Extension** in VSCode hinzufügen. und konfigurieren Sie die Datei launch.json wie folgt.

```
{
    "version": "0.2.0",
    "configurations": [
    {
                "name": "Run Android on device",
                "type": "cordova",
                "request": "launch",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Run iOS on device",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Attach to running android on device",
                "type": "cordova",
                "request": "attach",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Attach to running iOS on device",
                "type": "cordova",
                "request": "attach",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Run Android on emulator",
```

```
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Run iOS on simulator",
            "type": "cordova",
            "request": "launch",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Attach to running android on emulator",
            "type": "cordova",
            "request": "attach",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Attach to running iOS on simulator",
            "type": "cordova",
            "request": "attach",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Serve to the browser (ionic serve)",
            "type": "cordova",
            "request": "launch",
            "platform": "serve",
            "cwd": "${workspaceRoot}",
            "devServerAddress": "localhost",
            "sourceMaps": true,
            "ionicLiveReload": true
        },
        {
            "name": "Simulate Android in browser",
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "chrome",
            "simulatePort": 8000,
            "livereload": true,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
```

```
            {
                "name": "Simulate iOS in browser",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "chrome",
                "simulatePort": 8000,
                "livereload": true,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            }
]
}
```

Nach der Konfiguration folgen Sie den folgenden Schritten oder kurzen Schlüsseln zum Debuggen:

1. Gehen Sie zum **Debug-** Menü.
2. Klicken Sie auf **Debuggen starten** .

   oder

```
Short keys
```

- Debugging - F5
- StepOver - F10
- Einsteigen und Aussteigen - F11
- Beenden Sie das Debuggen - Umschalt + F5
- Starten Sie Debugging -ctrl + shift_F5 neu

Setup und Debugging von Ionic 2 in Visual Studio Code online lesen:
https://riptutorial.com/de/ionic2/topic/10559/setup-und-debugging-von-ionic-2-in-visual-studio-code

Kapitel 14: Social Login mit Angularfire2 / Firebase

Examples

Native Facebook-Anmeldung mit Angularfire2 / Firebase

App.ts

```
import {Component} from '@angular/core';
import {Platform, ionicBootstrap} from 'ionic-angular';
import {StatusBar} from 'ionic-native';
import {LoginPage} from './pages/login/login';
import {FIREBASE_PROVIDERS, defaultFirebase, AuthMethods, AuthProviders, firebaseAuthConfig}
from 'angularfire2';

@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>'
})

export class MyApp {

  private rootPage: any;

  constructor(private platform: Platform) {
    this.rootPage = LoginPage;

    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      StatusBar.styleDefault();
    });
  }
}

ionicBootstrap(MyApp, [
  FIREBASE_PROVIDERS,
  defaultFirebase({
    apiKey: myAppKey,
    authDomain: 'myapp.firebaseapp.com',
    databaseURL: 'https://myapp.firebaseio.com',
    storageBucket: 'myapp.appspot.com',
  }),
  firebaseAuthConfig({})
]);
```

login.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Home</ion-title>
  </ion-navbar>
</ion-header>
```

```
<ion-content padding class="login">
  <button (click)="facebookLogin()">Login With Facebook</button>
</ion-content>
```

login.ts

```
import {Component} from '@angular/core';
import {Platform} from 'ionic-angular';
import {AngularFire, AuthMethods, AuthProviders} from 'angularfire2';
import {Facebook} from 'ionic-native';

declare let firebase: any; // There is currently an error with the Firebase files, this will
fix it.

@Component({
  templateUrl: 'build/pages/login/login.html'
})
export class LoginPage {

  constructor(private platform: Platform, public af: AngularFire) {

  }

  facebookLogin() {
    Facebook.login(['public_profile', 'email', 'user_friends'])
      .then(success => {
        console.log('Facebook success: ' + JSON.stringify(success));
        let creds =
firebase.auth.FacebookAuthProvider.credential(success.authResponse.accessToken);
        this.af.auth.login(creds, {
          provider: AuthProviders.Facebook,
          method: AuthMethods.OAuthToken,
          remember: 'default',
          scope: ['email']
        }).then(success => {
          console.log('Firebase success: ' + JSON.stringify(success));
        }).catch(error => {
          console.log('Firebase failure: ' + JSON.stringify(error));
        });
      }).catch(error => {
        console.log('Facebook failure: ' + JSON.stringify(error));
      });
  }
}
```

Social Login mit Angularfire2 / Firebase online lesen:
https://riptutorial.com/de/ionic2/topic/5518/social-login-mit-angularfire2---firebase

Kapitel 15: Tabs verwenden

Bemerkungen

Denken Sie immer daran, die Ionic 2 Tab-Dokumente zu lesen , um sich über die neuesten Änderungen und Updates zu informieren.

Examples

Ändern Sie die ausgewählte Registerkarte programmgesteuert von der untergeordneten Seite

Sie können den vollständigen Code in diesem funktionierenden Plunker nachlesen .

In diesem Beispiel verwende ich einen gemeinsam genutzten Dienst, um die Kommunikation zwischen den Seiten innerhalb der Registerkarte (untergeordnete Seiten) und dem Registerkartencontainer (der Komponente, die die Registerkarten enthält) zu handhaben. Auch wenn Sie dies mit Events wahrscheinlich tun könnten, gefällt mir der Shared-Service-Ansatz, denn er ist einfacher zu verstehen und auch zu verwalten, wenn die Anwendung wächst.

TabService

```
import {Injectable} from '@angular/core';
import {Platform} from 'ionic-angular/index';
import {Observable} from 'rxjs/Observable';

@Injectable()
export class TabService {

  private tabChangeObserver: any;
  public tabChange: any;

  constructor(private platform: Platform){
    this.tabChangeObserver = null;
    this.tabChange = Observable.create(observer => {
        this.tabChangeObserver = observer;
    });
  }

  public changeTabInContainerPage(index: number) {
    this.tabChangeObserver.next(index);
  }
}
```

Der `TabService` erstellt also im `TabService` nur ein `Observable` , damit der Tabs-Container ihn abonnieren kann, und deklariert außerdem die `changeTabInContainerPage()` Methode, die von den `changeTabInContainerPage()` Seiten aufgerufen wird.

Dann fügen wir auf jeder untergeordneten Seite (die innerhalb der Registerkarten) nur eine Schaltfläche hinzu und binden das `click` Ereignis an eine Methode, die den Dienst aufruft:

Seite1.html

```
<ion-content class="has-header">
    <h1>Page 1</h1>
    <button secondary (click)="changeTab()">Select next tab</button>
</ion-content>
```

Seite1.ts

```
import { Component } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  constructor(private tabService: TabService) { }

  public changeTab() {
    this.tabService.changeTabInContainerPage(1);
  }
}
```

Und schließlich abonnieren wir in der `TabsPage` nur den Dienst und ändern dann die ausgewählte Registerkarte mit `this.tabRef.select(index);`

```
import { Component, ViewChild } from "@angular/core";
import { Page1 } from './page1.ts';
import { Page2 } from './page2.ts';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl: 'tabs.html'
})
export class TabsPage {
  @ViewChild('myTabs') tabRef: Tabs;

  tab1Root: any = Page1;
  tab2Root: any = Page2;

  constructor(private tabService: TabService){
    this.tabService.tabChange.subscribe((index) => {
      this.tabRef.select(index);
    });
  }
}
```

`#myTabs` Sie, dass wir einen Verweis auf die Tabs-Instanz erhalten, indem `#myTabs` im Element `ion-tabs #myTabs` hinzufügen. Wir erhalten ihn von der Komponente mit `@ViewChild('myTabs') tabRef: Tabs;`

```
<ion-tabs #myTabs>
  <ion-tab [root]="tab1Root" tabTitle="Tab 1"></ion-tab>
  <ion-tab [root]="tab2Root" tabTitle="Tab 2"></ion-tab>
```

```
</ion-tabs>
```

Registerkarte mit selectIndex wechseln

Anstatt einen Verweis auf das DOM zu erhalten, können Sie einfach den Index der Registerkarte mithilfe des selectedIndex-Attributs auf den Ionenregistern ändern

HTML:

```
<ion-tabs [selectedIndex]="tabIndex" class="tabs-icon-text" primary >
     <ion-tab tabIcon="list-box"   [root]="tabOne"></ion-tab>
     <ion-tab tabIcon="contacts"  [root]="tabTwo"></ion-tab>
     <ion-tab tabIcon="chatboxes"  [tabBadge]="messagesReceived" [root]="tabFive"></ion-tab>
</ion-tabs>
```

TS:

```
import { Events} from "ionic-angular";

export class tabs {
 public tabIndex: number;
 constructor(e: Events) {
   tabs.mySelectedIndex = navParams.data.tabIndex || 0;
   e.subscribe("tab:change", (newIndex) => this.tabIndex = newIndex);
 }
}
```

Wenn Sie es von einem anderen Controller-Dienst ändern möchten, können Sie ein Ereignis senden:

```
e.publish("tab:change",2);
```

Tabs verwenden online lesen: https://riptutorial.com/de/ionic2/topic/5569/tabs-verwenden

Kapitel 16: Unit Testing

Einführung

Komponententests geben einem Produkt im Allgemeinen zusätzliche Sicherheit, um Probleme beim Ändern / Hinzufügen von Funktionen zu vermeiden. Ein Sicherheitsnetz mit der Aufschrift "ALLES NOCH ARBEITET". Unit-Tests ersetzen in keiner Weise die tatsächlichen Benutzertests, die eine ordnungsgemäße Qualitätssicherung durchführen kann.

In diesem Dokument werden die Beispiele auf diesem Repository basieren:
https://github.com/driftyco/ionic-unit-testing-example

Examples

Unit-Tests mit Karma / Jasmin

Unit-Tests in Ionic sind die gleichen wie in jeder Winkel-App.

Wir werden dazu ein paar Frameworks verwenden.

Karma - ein Rahmen für die Durchführung von Tests

Jasmine - ein Rahmen zum Schreiben von Tests

PhantomJS - eine Anwendung, die Javascript ohne Browser ausführt

Zunächst einmal können wir alles installieren. Vergewissern Sie sich daher, dass package.json diese Zeilen in den dev-Abhängigkeiten enthält. Ich denke, es ist wichtig zu wissen, dass die Abhängigkeiten der Entwickler Ihre App überhaupt nicht beeinflussen und dem Entwickler lediglich helfen.

```
"@ionic/app-scripts": "1.1.4",
"@ionic/cli-build-ionic-angular": "0.0.3",
"@ionic/cli-plugin-cordova": "0.0.9",
"@types/jasmine": "^2.5.41",
"@types/node": "^7.0.8",
"angular2-template-loader": "^0.6.2",
"html-loader": "^0.4.5",
"jasmine": "^2.5.3",
"karma": "^1.5.0",
"karma-chrome-launcher": "^2.0.0",
"karma-jasmine": "^1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"karma-sourcemap-loader": "^0.3.7",
"karma-webpack": "^2.0.3",
"null-loader": "^0.1.1",
"ts-loader": "^2.0.3",
"typescript": "2.0.9"
```

Pakete ein bisschen durchgehen

```
"angular2-template-loader": "^0.6.2", - will load and compile the angular2 html files.

"ts-loader": "^2.0.3", - will compile the actual typescript files

"null-loader": "^0.1.1", - will not load the assets that will be missing, such as fonts and
images. We are testing, not image lurking.
```

Wir sollten dieses Skript auch unseren package.json-Skripten hinzufügen:

```
"test": "karma start ./test-config/karma.conf.js"
```

Beachten Sie auch in tsconfig, dass Sie die spec.ts-Dateien von der Kompilierung ausschließen:

```
 "exclude": [
    "node_modules",
    "src/**/*.spec.ts"
  ],
```

Ok, jetzt nehmen wir die eigentliche Testkonfiguration. Erstellen Sie einen `test-config` in Ihrem Projektordner. (So wie es im package.json-Skript erwähnt wurde) Erstellen Sie im Ordner 3 Dateien:

`webpack.test.js` - gibt dem Webpack an, welche Dateien für den Testvorgang geladen werden sollen

```
var webpack = require('webpack');
var path = require('path');

module.exports = {
  devtool: 'inline-source-map',

  resolve: {
    extensions: ['.ts', '.js']
  },

  module: {
    rules: [
      {
        test: /\.ts$/,
        loaders: [
          {
            loader: 'ts-loader'
          } , 'angular2-template-loader'
        ]
      },
      {
        test: /\.html$/,
        loader: 'html-loader'
      },
      {
        test: /\.(png|jpe?g|gif|svg|woff|woff2|ttf|eot|ico)$/,
        loader: 'null-loader'
      }
    ]
  },
```

```
  plugins: [
    new webpack.ContextReplacementPlugin(
      // The (\\|\/) piece accounts for path separators in *nix and Windows
      /angular(\\|\/)core(\\|\/)(esm(\\|\/)src|src)(\\|\/)linker/,
      root('./src'), // location of your src
      {} // a map of your routes
    )
  ]
};

function root(localPath) {
  return path.resolve(__dirname, localPath);
}
```

`karma-test-shim.js` - `karma-test-shim.js` die `karma-test-shim.js` Bibliotheken wie Zonen- und Testbibliotheken sowie das Modul zum Testen.

```
Error.stackTraceLimit = Infinity;

require('core-js/es6');
require('core-js/es7/reflect');

require('zone.js/dist/zone');
require('zone.js/dist/long-stack-trace-zone');
require('zone.js/dist/proxy');
require('zone.js/dist/sync-test');
require('zone.js/dist/jasmine-patch');
require('zone.js/dist/async-test');
require('zone.js/dist/fake-async-test');

var appContext = require.context('../src', true, /\.spec\.ts/);

appContext.keys().forEach(appContext);

var testing = require('@angular/core/testing');
var browser = require('@angular/platform-browser-dynamic/testing');

testing.TestBed.initTestEnvironment(browser.BrowserDynamicTestingModule,
browser.platformBrowserDynamicTesting());
```

`karma.conf.js` - `karma.conf.js` die Konfiguration für das Testen mit Karma. Hier können Sie von Chrome zu PhantomJS wechseln, um diesen Prozess unter anderem unsichtbar und schneller zu machen.

```
var webpackConfig = require('./webpack.test.js');

module.exports = function (config) {
  var _config = {
    basePath: '',

    frameworks: ['jasmine'],

    files: [
      {pattern: './karma-test-shim.js', watched: true}
    ],

    preprocessors: {
      './karma-test-shim.js': ['webpack', 'sourcemap']
```

```
    },

    webpack: webpackConfig,

    webpackMiddleware: {
      stats: 'errors-only'
    },

    webpackServer: {
      noInfo: true
    },

    browserConsoleLogOptions: {
      level: 'log',
      format: '%b %T: %m',
      terminal: true
    },

    reporters: ['kjhtml', 'dots'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
  };

  config.set(_config);
};
```

Nun, da wir alles konfiguriert haben, können wir einige Tests schreiben. Für dieses Beispiel schreiben wir eine app.component-Spezifikationsdatei. Wenn Sie Tests für eine Seite und nicht für die Hauptkomponente anzeigen möchten, können Sie hier nachsehen: https://github.com/driftyco/ionic-unit-testingexample/blob/master/src/pages/page1/page1. spez.ts

Zuerst müssen wir unseren Konstruktor testen. Dadurch wird der Konstruktor unserer app.component erstellt und ausgeführt

```
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));
```

Die Erklärung wird unsere Haupt-App für ionische Anwendungen enthalten. Die Importe werden für diesen Test benötigt. Nicht alles.

Die Provider enthalten die Elemente, die in den Konstruktor eingefügt werden, jedoch nicht Teil des Imports sind. Zum Beispiel fügt app.component den Platform-Dienst ein, aber da es ein Teil

des IonicModule ist, muss es in den Anbietern nicht erwähnt werden.

Für die nächsten Tests benötigen wir eine Instanz unserer Komponente:

```
  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });
```

Als nächstes ein paar Tests, um zu sehen, dass alles in Ordnung ist:

```
  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });
```

Am Ende haben wir also so etwas:

```
import { async, TestBed } from '@angular/core/testing';
import { IonicModule } from 'ionic-angular';

import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';

import { MyApp } from './app.component';

describe('MyApp Component', () => {
  let fixture;
  let component;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });

  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });
```

```
});
```

Führen Sie die Tests durch

```
npm run test
```

Und das ist schon alles beim Basis-Test. Es gibt verschiedene Möglichkeiten, um das Schreiben von Tests zu verkürzen, z. B. das Schreiben Ihres eigenen TestBed und die Vererbung von Tests, die Ihnen auf lange Sicht helfen können.

Unit Testing online lesen: https://riptutorial.com/de/ionic2/topic/9561/unit-testing

Kapitel 17: Vom Code zum App Store - Android

Einführung

Schrittweise Anweisungen zum Vorbereiten und Hochladen der ionischen Produktions-App auf Google Play.

Examples

Produktion fertig

App-Projekt erstellen

Wenn Sie eine für den App Store bereite Android-App erstellen, müssen Sie bei der Verwendung von `ionic start --appname|-a` und `--id|-i` Flags hinzufügen. `--appname|-a` werden für Google Play verwendet, um Ihre App anhand anderer Apps zu identifizieren.

Wenn Sie ein neues Projekt für eine mobile App starten, können Sie das folgende cli-Beispiel verwenden.

```
$ ionic start --v2 -a "App Example" -i "com.example.app" -t "tabs"
```

1. App-Konfigurationsdatei

Wenn Sie diese Informationen in einer vorhandenen App `config.xml` möchten, können Sie `config.xml` ändern. Ich empfehle denen, die den Befehl oben verwendet haben, ebenfalls die `config.xml` zu ändern.

Bestätigen / Bearbeiten von `widget id` , `name` , `description` und `author` .

Beispiel:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widget id="com.example.app" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>Example App</name>
  <description>Example app for stackoverflow users</description>
  <author email="admin@example.com" href="http://example.com/">Your name or team</author>
  ...
</widget>
```

2. Symbol und Startbildschirm

Die unterstützten Dateitypen für das Symbol und das Splash-Image sind png, psd oder ai und müssen einen Dateinamen haben, der dem `icon` oder dem `splash` entspricht, das unter dem

Ressourcenverzeichnis im Stammverzeichnis Ihres Projekts abgelegt ist. Die Mindestabmessungen des Symbols sollten 192x192 px betragen und dürfen keine abgerundeten Ecken haben. und der Startbildschirm ist viel komplizierter. Klicken Sie hier, um mehr zu erfahren. Die Mindestabmessungen sollten jedoch 2208 x 2208 Pixel betragen.

Wenn Sie eine Icon-Datei zum Generieren haben, verwenden Sie diesen Befehl `ionic resources --icon` wenn Sie eine Splash-Datei zum Generieren des Befehls `ionic resources --splash`

3. Gebäudeproduktion App

Bevor Sie Ihre Produktions-App erstellen, entfernen Sie alle vertraulichen Protokolldaten.

Um eine Release-Version mit allen Standardoptimierungen zu erstellen, verwenden Sie das Tag --release & --prod

```
ionic build android --release --prod
```

Eine vollständige Liste der verfügbaren Optimierungen finden Sie im @ ionic / app-scripts-Repository

4. Erstellen Sie einen privaten Schlüssel

Jetzt müssen wir die nicht signierte APK (`android-release-unsigned.apk`) signieren und ein Ausrichtungsprogramm ausführen, um sie zu optimieren und für den App Store vorzubereiten. Wenn Sie bereits über einen Signaturschlüssel verfügen, überspringen Sie diese Schritte und verwenden Sie stattdessen diesen.

Suchen Sie als Nächstes Ihre nicht signierte APK-Datei `android-release-unsigned.apk` im Projektverzeichnis `/platforms/android/build/outputs/apk/ keytools /platforms/android/build/outputs/apk/` und verwenden `keytools` Befehl `keytools` , mit dem unsere apk-Datei signiert wird. Sie können das Beispiel unten verwenden:

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias androidKey -keyalg RSA -keysize
2048 -validity 10000
```

Sie finden `my-release-key.keystore` in Ihrem aktuellen Verzeichnis.

Lassen Sie uns unseren privaten Schlüssel mit dem im JDK enthaltenen Befehl keytool generieren. Wenn dieses Tool nicht gefunden wird, schlagen Sie in der Installationsanleitung nach:

Sie werden zunächst aufgefordert, ein Kennwort für den Keystore zu erstellen. Beantworten Sie dann die restlichen Fragen der Tools und wenn alles erledigt ist, sollten Sie eine Datei namens my-release-key.keystore im aktuellen Verzeichnis erstellen.

Hinweis: Speichern Sie diese Datei an einem sicheren Ort. Wenn Sie sie verlieren, können Sie keine Aktualisierungen an Ihre App senden.

5. Unterzeichnen Sie APK

Führen Sie zum Signieren der nicht signierten APK das Jarsigner-Tool aus, das auch im JDK enthalten ist:

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore
HelloWorld-release-unsigned.apk alias_name
```

Dies kennzeichnet den apk an Ort und Stelle. Schließlich müssen wir das Zip-Ausrichtungs-Tool ausführen, um die APK zu optimieren. Das zipalign-Tool kann in / path / to / Android / sdk / build-tools / VERSION / zipalign gefunden werden.

```
$ zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```

Jetzt haben wir unsere letzte Binärversion namens HelloWorld.apk, die wir im Google Play Store veröffentlichen können, damit alle die Welt genießen können!

Veröffentlichen Sie Ihre App im Google Play Store. Jetzt, da wir unsere Release-APK für den Google Play Store bereit haben, können wir einen Play Store-Eintrag erstellen und unsere APK hochladen. Um zu beginnen, müssen Sie die Google Play Store Developer Console besuchen und ein neues Entwicklerkonto erstellen. Es kostet einmalig $ 25.

Wenn Sie über ein Entwicklerkonto verfügen, klicken Sie auf "Android-App bei Google Play veröffentlichen" und folgen Sie den Anweisungen auf dem Bildschirm.

Vom Code zum App Store - Android online lesen: https://riptutorial.com/de/ionic2/topic/9659/vom-code-zum-app-store---android

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit ionic2	Akilan Arasu, Cameron637, carstenbaumhoegger, Community, FreeBird72, Guillaume Le Mière, Ian Pinto, Ketan Akbari, misha130, Raymond Ativie, sebaferreras, tymspy, Will.Harris
2	Angularfire2 mit Ionic2	Fernando Del Olmo
3	Dienste verwenden	sebaferreras
4	Geolocation	Matyas, misha130
5	Hinzufügen einer ionischen App zur ionischen Ansicht	Saravanan Sachi
6	InAppBrowser	niks
7	Ionic2-CSS-Komponenten	Ketan Akbari
8	Konstruktor und OnInit	niks
9	Modals	Raymond Ativie
10	Problemumgehung für 'show-delete' in Missbilligung	Amr ElAdawy, Roman Lee
11	Push-Benachrichtigung gesendet und empfangen	misha130
12	Setup und Debugging von Ionic 2 in Visual Studio Code	misha130, PRIYA PARASHAR
13	Social Login mit Angularfire2 / Firebase	Cameron637, Gianfranco P.

14	Tabs verwenden	misha130, sebaferreras
15	Unit Testing	misha130
16	Vom Code zum App Store - Android	Luis Estevez, misha130