Бесплатная электронная книга

учусь ionic2

Free unaffiliated eBook created from
Stack Overflow contributors.

#ionic2

Около

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

глава 1: Начало работы с ionic2

замечания

Ionic 2 - это кросс-платформенная мобильная технология разработки. Эта структура создана для создания гибридных мобильных приложений, и ее также можно использовать и для настольных приложений. Это однократная запись, работающая везде по технологиям. Он использует веб-технологии, такие как JavaScript / Typcript, Angular 2, HTML и CSS (SCSS / LESS). Приложения Ionic2 хорошо работают на `>=android 4.4` , но вы хотите запустить `android 4.1` для `android 4.3` вам нужно использовать перекрестный ход .

Examples

Установка или настройка

Поскольку Ionic 2 становится все лучше и лучше с каждым днем, всегда проверяйте **официальную документацию,** чтобы отслеживать последние изменения и улучшения.

Предварительные требования: вам понадобится NodeJS для создания проектов Ionic 2. Вы можете загрузить и установить узел здесь и узнать больше о npm и использовать здесь Ionic 2.

---

1. Установка Ionic 2

Как и Ionic 1, вы можете использовать Ionic CLI или GUI, чтобы быстро создавать и тестировать приложения прямо в браузере. У него даже есть все возможности для работы с вашими приложениями Ionic 1, поэтому вам не нужно ничего менять!

Чтобы использовать Ionic 2, просто установите ионный из npm:

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

Если вы получаете ошибку EACCES, следуйте инструкциям здесь, чтобы предоставить узлу необходимые ему разрешения.

2. Создание первого приложения

После того, как CLI установлен, выполните следующую команду, чтобы запустить свое

первое приложение:

```
$ ionic start MyIonic2Project
```

Шаблон табуляции используется по умолчанию, но вы можете выбрать другой шаблон, передав флаг. Например:

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

Это будет использовать шаблон учебника .

Чтобы запустить приложение, перейдите в каталог своих проектов и запустите приложение- `ionic serve -lc` :

```
$ ionic serve -lc
```

-L активирует текущую перезагрузку страницы, -c отображает журналы консоли. Если у вас возникли проблемы с созданием вашего приложения, убедитесь, что ваш пакет.json соответствует тому, который находится в базе данных ionic2-app-base

Вы можете играть с новым приложением прямо в браузере!

3. Построение устройства

Вы также можете создать новое приложение на физическом устройстве или эмуляторе устройства. Кордове понадобится продолжить.

Чтобы установить Кордову, запустите:

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

Ознакомьтесь с документами iOS simulator для создания приложений iOS (ПРИМЕЧАНИЕ: вы не можете создавать на устройствах iOS или эмуляторах в любой операционной системе, отличной от OSX), или в документах Genymotion для создания приложения для Android.

Работа на устройстве iOS:

Чтобы создать приложение iOS, вам необходимо работать на компьютере OSX, потому что вам понадобится каркас какао для сборки для ios, если это так, вам сначала нужно добавить платформу к кордове, запустив следующая команда:

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

Вам понадобится Xcode для компиляции на устройство iOS.

Наконец, запустите приложение со следующей командой:

```
$ ionic cordova run ios
```

Запуск на устройстве Android:

Шаги для Android почти идентичны. Во-первых, добавьте платформу:

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

Затем установите Android SDK, который позволяет скомпилировать Android-устройство. Хотя Android SDK поставляется с эмулятором, он работает очень медленно. Genymotion намного быстрее. После установки просто выполните следующую команду:

```
$ ionic cordova run android
```

И это все! Поздравляем с созданием вашего первого приложения Ionic 2!

Ионная перегрузка тоже. Поэтому, если вы хотите разработать приложение и увидеть изменения, происходящие в реальном времени на эмуляторе / устройстве, вы можете сделать это, выполнив следующие команды:

Для iOS:

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

Будьте осторожны, на iOS 9.2.2 работа с печью не работает. Если вы хотите работать с функцией livereload, отредактируйте файл config.xml, добавив следующее:

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

Затем в `<platform name="ios">` :

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

Для Android:

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

`l` означает перезагрузку, `c` для консольных журналов и `s` для журналов сервера. Это позволит вам увидеть, есть ли какие-либо ошибки / предупреждения во время выполнения.

Строительство для Windows

Если вы хотите создать свой проект для окон, вам нужно работать на компьютере с Windows. Чтобы начать, установите платформу windows в проект ionic2, выполнив следующую команду:

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

Затем просто выполните следующую команду:

```
$ionic cordova run windows
```

Запуск в браузере

```
$ionic serve
```

для устройства проверки браузера Chrome (введите адресную строку браузера Chrome)

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

Прочитайте Начало работы с ionic2 онлайн: https://riptutorial.com/ru/ionic2/topic/3632/начало-работы-с-ionic2

глава 2: Angularfire2 с Ionic2

Вступление

Здесь плохо показано, как интегрировать AngularFire2 и использовать эту базу данных в реальном времени в нашем приложении Ionic.

Examples

Инициализация AngularFire

Прежде всего, вам необходимо инициализировать модули угловых огней в вашем модуле приложения следующим образом:

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

Вы можете получить эти ключи, подписавшись на firebase и создав новый проект.

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

Использование AngularFire2

Если у вас есть это приложение, просто импортируйте его:

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

С помощью этого списка наблюдений вы можете получить доступ к списку предметов под контуром, например, если у вас есть root / items / food, вы можете получить такие продукты:

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

И вы можете просто поставить новый элемент здесь и появиться в базе данных firebase, или вы можете обновить один элемент, и вы увидите его обновление в своей базе данных. Вы можете нажать и обновить так:

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

Или вы можете даже удаленные элементы из списка продуктов:

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

Прочитайте Angularfire2 с Ionic2 онлайн:
https://riptutorial.com/ru/ionic2/topic/10918/angularfire2-c-ionic2

глава 3: InAppBrowser

Вступление

Иногда клиенту просто нужно открыть веб-приложение в мобильном приложении, для этого мы можем использовать InAppBrowser таким образом, чтобы оно выглядело как приложение, вместо этого мы открываем веб-сайт / webApp на мобильных устройствах, как только пользователь нажимает значок приложения вместо того, чтобы открывать первое представление приложения, мы можем напрямую открыть InAppBrowser.

Examples

Живым примером этого использования является это приложение:

В этом приложении я сразу открываю InAppBrowser, когда пользователь нажимает значок приложения вместо загрузки первой страницы приложения. Таким образом, это будет выглядеть для пользователя, что они просматривают приложение того же веб-сайта / webapp.

Пример кода для использования 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();
        }
      );
```

Прочитайте InAppBrowser онлайн: https://riptutorial.com/ru/ionic2/topic/9801/inappbrowser

глава 4: Push-уведомление отправлено и получено

замечания

Идентификатор SenderID, присутствующий в примере инициализации, представляет собой идентификатор отправителя gcm, который предоставляется вам Google. Он также должен присутствовать при установке плагина

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

Если вы хотите добавить дополнительные данные в свои push-уведомления, посмотрите на эту ссылку, объяснив, как добавить больше типичных сообщений
https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/TYPESCRIPT.md

Examples

инициализация

Плагин push-уведомления требует инициализации инициализации, которая сообщает плагину, что он запускается с использованием предоставленного идентификатора отправителя.

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

Постановка на учет

Шаг регистрации регистрирует приложение с помощью системы устройства и возвращает идентификатор регистрации

```
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);

        });
```

Получение push-уведомления

Чтобы получать push-уведомления, мы должны сообщить плагину прослушивать входящие push-уведомления. Этот шаг выполняется после инициализации и регистрации

```
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));
       });
```

Прочитайте Push-уведомление отправлено и получено онлайн:
https://riptutorial.com/ru/ionic2/topic/5874/push-уведомление-отправлено-и-получено

глава 5: геолокации

Examples

Простое использование

В вашем `package.json` обязательно включите зависимости:

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

Использовать геолокацию:

```
// 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);
        });
    }
}
```

Наблюдение за положением

Для более реального времени вы можете использовать функцию watchPosition в Geolocation, которая уведомляет, когда происходит ошибка или изменение позиции. В отличие от getCurrentPosition, watchPosition возвращает Observable

```
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);

        });
}
```

Прочитайте геолокации онлайн: https://riptutorial.com/ru/ionic2/topic/5840/геолокации

глава 6: Добавить ионное приложение для ионного просмотра

Вступление

ionic view - это мобильное приложение, которое вы должны установить на своем мобильном устройстве, чтобы вы могли просматривать свое приложение, не создавая файлы .apk. Делитесь своим идентификатором приложения, другие также могут просматривать ваше приложение на своем мобильном телефоне с помощью ионного представления.

Сайт: https://view.ionic.io/

Examples

Шаги для добавления приложения в ионный вид

Следующие шаги необходимо выполнить в app.ionic.io

1. Создайте учетную запись или войдите в свою ионную учетную запись
2. Нажмите «Новое приложение» в панели мониторинга и укажите имя для своего приложения.

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

3. В разделе обзора этого недавно созданного приложения появится идентификатор под именем приложения.

```
MyIonicApp ID is 4c5051c1
```

Ниже приведены шаги в командной строке **Node.js**

1. Войдите в свою ионную учетную запись, запустив

```
$ ionic login
```

2. Корните папку приложения.
3. Чтобы загрузить приложение в ионный режим, сначала нужно связать свое приложение с идентификатором, созданным на ионном сайте. Выполните следующую команду для соединения,

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

Для MyIoincApp команда будет,

```
$ ionic link 4c5051c1
```

Вышеупомянутая команда обновит идентификатор приложения в конфигурационном файле MyIonicApp.

4. После того, как связь будет выполнена, загрузите приложение, выполнив

```
$ ionic upload
```

Заметка

Как только загрузка будет успешной, откройте ионный просмотр на мобильном устройстве, чтобы просмотреть приложение.

Другие могут просматривать ваше приложение, отправив идентификатор приложения в разделе «Предварительный просмотр приложения» в ионном представлении.

Прочитайте Добавить ионное приложение для ионного просмотра онлайн: https://riptutorial.com/ru/ionic2/topic/10542/добавить-ионное-приложение-для-ионного-просмотра

глава 7: Использование вкладок

замечания

Всегда помните, чтобы проверить документы Ionic 2 Tab, чтобы знать о последних изменениях и обновлениях.

Examples

Измените выбранную вкладку программно с дочерней страницы

Вы можете посмотреть полный код этого рабочего Plunker .

В этом примере я использую общую службу для обработки связи между страницами внутри вкладки (дочерние страницы) и контейнером табуляции (компонент, который содержит вкладки). Несмотря на то, что вы, вероятно, могли бы это сделать с помощью событий, мне нравится подход к общему сервису, потому что его легче понять, а также поддерживать, когда приложение начинает расти.

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);
  }
}
```

Таким образом, в основном `TabService` создает только `Observable` чтобы позволить табуляции подписываться на него, а также объявляет метод `changeTabInContainerPage()` который будет вызываться из дочерних страниц.

Затем на каждой дочерней странице (внутри вкладки) мы добавляем только кнопку и

привязываем событие `click` к методу, который вызывает службу:

page1.html

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

Page1.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);
  }
}
```

И, наконец, в `TabsPage` мы подписываемся только на услугу, а затем меняем выбранную вкладку с `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);
    });
  }
}
```

Обратите внимание, что мы получаем ссылку на экземпляр Tabs, добавляя `#myTabs` в элемент `ion-tabs` , и мы получаем его из компонента с помощью `@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>
```

Заменить вкладку с помощью selectedIndex

Вместо того, чтобы получать ссылку на DOM, вы можете просто изменить индекс вкладки, используя атрибут selectedIndex на вкладках ионов

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);
  }
}
```

Если вы хотите изменить его из какой-либо другой службы контроллера, вы можете отправить событие:

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

Прочитайте Использование вкладок онлайн: https://riptutorial.com/ru/ionic2/topic/5569/использование-вкладок

глава 8: Использование сервисов

замечания

Одна очень важная вещь об использовании общих служб заключается в том, что они должны быть включены в массив `providers` самого верхнего компонента, где они должны использоваться совместно.

Это почему? Ну, предположим, что мы `MyService` ссылку `MyService` в массив `providers` из каждого `Component` . Что-то вроде:

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

А также

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

Таким образом, **для каждого компонента будет создан новый экземпляр службы,** поэтому экземпляр, где одна страница будет сохранять данные, будет отличаться от экземпляра, используемого для получения данных. Так что это не сработает.

Чтобы все приложение использовало один и тот же экземпляр (чтобы служба работала как *однопользовательская* служба), мы можем добавить ее ссылку в `App Component` следующим образом:

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

Вы также можете добавить ссылку `MyService` в `ionicBootstrap(MyApp, [MyService]);` но в соответствии с инструкциями по стилю Angular2

> Предоставляйте услуги инжектору Angular 2 в самом верхнем компоненте, где они будут использоваться совместно.
>
> Зачем? Инжектор Angular 2 является иерархическим.
>
> Зачем? При предоставлении услуги компоненту верхнего уровня этот экземпляр является общим и доступен для всех дочерних компонентов этого компонента

верхнего уровня.

Зачем? Это идеально, когда служба использует методы или состояние.

Зачем? Это не идеально, когда двум различным компонентам нужны разные экземпляры службы. В этом случае было бы лучше предоставить службу на уровне компонента, для которого нужен новый и отдельный экземпляр.

А также

> Это будет работать. Это не лучшая практика. **Опция поставщика бутстрапов предназначена для настройки и переопределения собственных предварительно зарегистрированных сервисов Angular** , таких как поддержка маршрутизации.

... `App Component` был бы лучшим выбором.

Examples

Разделять информацию между разными страницами

Один из самых простых примеров использования *общих служб* - это когда мы хотим хранить некоторые данные с данной страницы нашего приложения, а затем снова получать эти данные, но с другой страницы.

Один из вариантов может состоять в том, чтобы отправить эти данные в качестве параметра (например, если одна страница вызывает другую), но если мы хотим использовать эти данные из совершенно другой части приложения, это, по-видимому, не лучший способ сделать Это. Вот когда приходят *совместные сервисы* .

В этом примере мы собираемся использовать простой сервис под названием `MyService` который имеет только два простых метода: `saveMessage()` чтобы сохранить строку и `getMessage()` чтобы получить ее снова. Этот код является частью этого рабочего плунжера, где вы можете увидеть его в действии.

```
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;
  }
}
```

Затем, когда мы хотим сохранить новое сообщение, мы можем просто использовать `saveMessage(theMessageWeWantToSave);` метод из экземпляра `MyService` (называемый просто `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);
  }
}
```

Точно так же, когда мы хотим получить эти данные, мы можем использовать метод `getMessage()` из экземпляра службы следующим образом:

```
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();
  }

  // ...
}
```

Не забудьте проверить раздел « *Примечания* », чтобы узнать, где должна быть включена ссылка для службы `MyService` и почему.

Прочитайте Использование сервисов онлайн: https://riptutorial.com/ru/ionic2/topic/4407/использование-сервисов

глава 9: Компоненты Ionic2 CSS

Examples

сетка

Сетчатая система Ionic основана на flexbox, CSS-функции, поддерживаемой всеми устройствами, поддерживаемыми Ionic. Сетка состоит из трех единиц - сетки, строк и столбцов. Столбцы будут расширяться, чтобы заполнить их строку, и изменит размер, чтобы они соответствовали дополнительным столбцам.

Учебный класс	ширина
ширина-10	10%
ширина-20	20%
ширина-25	25%
ширина-33	33,3333%
ширина-50	50%
ширина-67	66,6666%
ширина-75	75%
ширина-80	80%
ширина-90	90%

Пример.

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

Карты

Карты - отличный способ отображать важные фрагменты контента и быстро появляются в качестве основного шаблона проектирования для приложений. Они - отличный способ сдерживать и упорядочивать информацию, а также настраивать ожидаемые ожидания для пользователя. С таким количеством контента, чтобы отображать сразу, и часто так

мало экранной недвижимости, карты быстро стали образцом дизайна для многих компаний.

Пример.

```
<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>
```

Прочитайте Компоненты Ionic2 CSS онлайн: https://riptutorial.com/ru/ionic2/topic/8011/компоненты-ionic2-css

глава 10: Конструктор и OnInit

Вступление

В отношении ionic2 `constructor` : в простых терминах мы используем его для создания экземпляра наших плагинов, сервисов и т. Д., Например: у вас есть страница (представление), где вы хотите показать список всех учеников, и у вас есть json-файл который содержит всех студентов (этот файл является вашим файлом данных), вам нужно создать службу в этой службе, вы создадите метод и получите запрос http.get, чтобы получить данные json, так что вам нужно что? http просто так:

Examples

Пример метода студенческой службы для использования Http в конструкторе

```
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)
        }
    }
```

снова обратите внимание на конструктор, если мы хотим использовать этот метод сервиса, мы перейдем к нашему представлению / странице и:

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

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

снова обратите внимание на конструктор здесь, мы создаем экземпляр StudentService в конструкторе, и еще одна вещь, мы используем плагин socialSharing, чтобы использовать то, что мы создаем экземпляр этого в конструкторе.

Метод ngOnInit, чтобы получить список студентов в режиме просмотра

`OnInit` : это действительно потрясающая вещь в ionic2, или мы можем сказать в AngularJs2. В этом же примере мы видим, что такое ngOnInit. Таким образом, вы готовы с помощью метода обслуживания, теперь, когда на вашем представлении / странице вы хотите, чтобы

данные списка учащихся были доступны, как только появится ваше представление, это должна быть первая операция, выполняемая автоматически при загрузке, потому что, поскольку просмотр загружает ученика список должен быть видимым. Таким образом, класс реализует OnInit, и вы определяете ngOnInit. Пример:

Пример ngOnInit, чтобы получить список студентов на странице / просмотре

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

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

Прочитайте Конструктор и OnInit онлайн: https://riptutorial.com/ru/ionic2/topic/9907/конструктор-и-oninit

глава 11: модальности

Examples

Использование модалов

Модальные слайды выходят за пределы экрана, чтобы отображать временный интерфейс, часто используемый для входа или регистрации страниц, составления сообщений и выбора опций.

```
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();
  }
}
```

ПРИМЕЧАНИЕ. Модаль - это область содержимого, которая просматривает текущую страницу пользователя.

Передача данных через Modal

В качестве второго аргумента данные могут быть переданы новому `Modal.create()` через `Modal.create()` . Затем данные можно получить с открытой страницы, введя `NavParams` . Обратите внимание, что страница, открытая как модальная, не имеет в ней специальной «модальной» логики, но использует `NavParams` не иначе, как стандартную страницу.

Первая страница:

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

export class HomePage {

 constructor(public modalCtrl: ModalController) {

 }

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

}
```

Вторая страница:

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

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

}
```

Прочитайте модальности онлайн: https://riptutorial.com/ru/ionic2/topic/6415/модальности

глава 12: модальности

Examples

Простые модальные

Modal - это временный интерфейс, который отображается поверх текущей страницы. Это часто используется для входа в систему, регистрации, редактирования существующих параметров и выбора параметров.

Давайте рассмотрим простой пример использования модалов. Для начала мы создаем ионный пустой проект. Создадим простой способ отображения сообщения и выхода из него. Для этого мы создаем представление для нашего модального.

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();
  }
}
```

Этот модальный отображает сообщение. Модаль может быть закрыт или «отклонен» с помощью метода **уклонения** диспетчера View.

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();
  }
}
```

Теперь мы создаем нашу домашнюю страницу, импортируя **ModalController** и нашу модель MessagePage. Способ **создания** ModalController создает модальный для нашей модели данных MessagePage, который сохраняется для управления переменной myModal. **Существующий** метод открывает модальность поверх нашей текущей страницы.

Modal с параметрами при увольнении:

Теперь мы знаем, как создать модальный. Но что, если мы хотим передать некоторые данные из модального на нашу домашнюю страницу. Для этого давайте рассмотрим пример с модальными параметрами передачи страницы регистрации на родительскую страницу.

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);
  }
}
```

Регистрация modal получает объект данных со значениями, введенными пользователем, и параметры передаются на нашу текущую страницу при увольнении с помощью метода отклонения viewControllers. Теперь параметры отправляются.

Итак, как мы собираемся восстановить параметры на домашней странице? Для этого мы создаем кнопку на домашней странице и вызываем «Зарегистрировать модальный клик». Чтобы отобразить пользователя, мы показываем список.

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);
      }
    });
  }
```

Метод ViewController **onDidDismiss** выполняется, когда модальный объект закрыт. Если данные передаются как параметр из модального, то мы можем получить его с помощью метода onDidDismiss. Здесь данные, введенные пользователем, добавляются к существующим данным. Если данные не передаются как параметр, возвращаемое значение будет равно нулю.

Модальный с параметрами для создания:

Передача параметров в модальную форму аналогична тому, как мы передаем значения в NavController. Для этого мы изменяем наш список в home.html, чтобы открыть модальный щелчок по элементу списка и передать необходимые параметры в качестве второго аргумента методу **create** .

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();
  }
```

Подобно другим представлениям, мы используем NavParams для извлечения данных, отправленных из предыдущего представления.

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();
  }
}
```

Прочитайте модальности онлайн: https://riptutorial.com/ru/ionic2/topic/6612/модальности

глава 13: Настройка и отладка Ionic 2 в коде Visual Studio

Вступление

Visual Studio - это среда с открытым исходным кодом, которая предоставляет средство intellisense и редактирования для кода. Эта среда IDE поддерживает многие языки, такие как (Ionic, C, C #, AngularJs, TypeScript, Android и т. Д.). Эти языки могут выполнять там код, добавляя свои расширения в VSCode. Используя VSCode, мы можем запускать и отлаживать код на разных языках.

Examples

Установка VSCode

Во-первых, вам нужно загрузить и установить VSCode. Последняя версия VSCode доступна для загрузки на официальном сайте . После загрузки VSCode вы должны установить и открыть его.

```
Introduction of Extensions in VSCode
```

VSCode - это открытый редактор, поэтому он предоставляет редактор для всех языков, но для выполнения кода, необходимого для добавления расширения для этого конкретного языка. Для запуска и редактирования вашего ионного кода вы должны добавить **расширение ionic2-vscode** в свой VSCode. В левой части редактора VSCode имеется 5 значков, в которых для расширения используется самый низкий значок. Расширения, которые вы можете получить, используя **комбинацию клавиш (ctrl + shift + X)** .

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

При нажатии **Ctrl + Shift + X** вы показали часть расширения , где на показаны первые **три точки ...** эти точки известны как более icon.On щелчком него диалог открыт и показывает число вариантов выбора .Вы может выберите вариант в соответствии с вашими потребностями, но для получения всего расширения вы должны выбрать « **Рекомендуемое расширение».** iN список всех eXtension вы можете установить расширение `(ionic2-vscode),npm`

Создание и добавление вашего Ионного проекта в VSCode

VsCode не может создать ионный проект, потому что это редактор кода. Таким образом, вы можете создать свой ионный проект с помощью **CLI** или **cmd** . создайте свой проект по

команде ниже

```
$ ionic start appName blank
```

Выше команда используется для создания чистого ионного приложения шаблона. Ionic2 предоставляет три типа шаблонов: **пустые, вкладки и sidemenu** . Так. вы можете заменить пустой шаблон на любые другие два шаблона в соответствии с вашими потребностями.

Теперь ваш Ионный проект был создан. Таким образом, вы можете добавить свой проект в VSCode для редактирования. Чтобы добавить свой проект, следуйте ниже.

1. Перейдите в меню « **Файл»** в VScode.
2. Нажмите « **Открыть папку»** в меню «Файл».
3. Найдите и откройте папку проекта.

   Вы можете напрямую открыть папку с помощью сочетания клавиш **Ctrl + O** или **Ctrl + K**

`

Запуск и отладка вашего Ионного проекта

> Запуск и отладка в Chrome

Для запуска использования ионного проекта ниже команды в **терминале или CMD или CLI**

```
$ ionic serve
```

Для отладки ионного проекта сначала необходимо добавить расширение **(Debugger for chrome),** а затем настроить файл launch.json следующим образом.

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

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

> Запуск и отладка в Android

Для запуска ионного проекта в Android вы должны добавить платформу Android под командой ниже в терминале или cmd или CLI:

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

Создайте Android по этой команде

```
$ ionic cordova build android
```

Запустить команду для платформы Android

```
$ ionic cordova run android
```

Теперь ваше приложение работает на реальном устройстве Android.

Для отладки в Android-устройстве вам нужно добавить **Cordova или Android Extension** в VSCode. и настройте файл launch.json следующим образом.

```
{
    "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}"
            }
]
}
```

После настройки вы выполните следующие шаги или короткие клавиши для отладки:

1. Перейдите в меню **отладки** .
2. Нажмите кнопку « **Отладка**» .

   или же

```
Short keys
```

- Отладка - F5
- StepOver - F10
- Шаг вперед и выход - F11
- Stop Debugging - Shift + F5
- Перезапустить отладку -ctrl + shift_F5

Прочитайте Настройка и отладка Ionic 2 в коде Visual Studio онлайн:
https://riptutorial.com/ru/ionic2/topic/10559/настройка-и-отладка-ionic-2-в-коде-visual-studio

глава 14: Обходной путь для 'show-delete' в неодобрение

Examples

Решение

Я разрабатываю мобильное приложение с использованием ионного 2 с угловым 2.

У меня есть ионные списки, заполненные ионом. Я хочу, чтобы у этих ионных предметов была возможность быть удалена, если это необходимо, как представлено здесь на ионном веб-сайте.

Однако многое изменилось в **ионной** версии **2,** так как первая версия и вышеупомянутый стиль одной кнопки, открывающей все **ионный элемент** на одном, больше не возможны, так как **show-delete** и **show-reorder** больше не поддерживаются. Единственный доступный вариант - это использовать **ионный предмет,** как ионный элемент, который дает возможность скользить по каждому пункту по одному, чтобы открыть кнопку удаления.

Это не то, что я хотел. Я хотел, чтобы одна кнопка открывала все ионные предметы одновременно.

Потратив некоторое время на это, я придумал рабочее решение и смог добиться желаемого результата с помощью ionic 2, и я собираюсь поделиться им с вами.

Вот мое решение:

В файле .html:

```
<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>
```

и для списка:

```
<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>
```

В файле **.ts** сначала сделайте свой импорт:

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

затем обратитесь к элементу html, объявив ViewChild:

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

Наконец, добавьте свои классы для обработки работы:

```
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;
    }
}
```

Тогда класс открытия:

```
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)")
}
```

И закрывающий класс:

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

}

Надеюсь, это поможет некоторым вам там.

Наслаждайтесь и хорошо кодируйте ...

Прочитайте Обходной путь для 'show-delete' в неодобрение онлайн: https://riptutorial.com/ru/ionic2/topic/6620/обходной-путь-для--show-delete--в--ion-list--неодобрение

глава 15: От кода до магазина приложений - Android

Вступление

Вы найдете пошаговые инструкции о том, как подготовить и загрузить производственное ионное приложение в Google Play.

Examples

Производство готово

Создание проекта приложения

При создании Android-приложения, готового для магазина приложений, при использовании `ionic start` важно добавить `--appname|-a` и `--id|-i` которые используются для игры Google для идентификации вашего приложения из других приложений.

Если вы начинаете новый проект для мобильных приложений, вы можете использовать приведенный ниже пример cli.

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

1. Файл конфигурации приложения

если вы хотите установить эту информацию внутри существующего приложения, вы можете изменить `config.xml` . Я рекомендую тем, кто использовал приведенную выше команду, чтобы изменить `config.xml` .

Подтверждение / редактирования `widget id` , `name` , `description` и `author` атрибуты.

Пример:

```
<?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. значок и заставка

Поддерживаемые типы файлов и изображений и изображений с брызгами - это png, psd или ai и должны иметь имя файла, соответствующее тому, что является `icon` или `splash` и помещаться под ним в корневой каталог вашего проекта. Минимальные размеры изображения значка должны быть 192x192 px и не должны иметь округлых углов. и экран заставки намного сложнее, поэтому нажмите здесь, чтобы прочитать больше. Тем не менее, минимальные размеры должны быть 2208x2208 px.

если у вас есть файл значка для генерации, используйте эту команду `ionic resources --icon` если у вас есть файл всплеска для генерации, используйте эту команду. `ionic resources --splash`

3. Приложение для создания здания

Перед созданием вашего производственного приложения удалите любые конфиденциальные данные журнала.

Чтобы создать версию выпуска со всеми оптимизациями по умолчанию, используйте тег --release & --prod

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

Для получения полного списка доступных оптимизаций вы можете посетить репозиторий @ ionic / app-scripts

4. Создайте закрытый ключ

Теперь нам нужно подписать unsigned APK (`android-release-unsigned.apk`) и запустить на нем утилиту выравнивания, чтобы оптимизировать ее и подготовить для магазина приложений. Если у вас уже есть ключ подписи, пропустите эти шаги и используйте этот вариант.

Затем найдите неподписанный файл APK `android-release-unsigned.apk` внутри проекта dir `/platforms/android/build/outputs/apk/ keytools /platforms/android/build/outputs/apk/` и используйте команду `keytools` которая будет использоваться для подписи нашего файла apk. Вы можете использовать приведенный ниже пример:

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

вы можете найти `my-release-key.keystore` в вашем текущем каталоге.

Давайте сгенерируем наш закрытый ключ, используя команду keytool, которая поставляется вместе с JDK. Если этот инструмент не найден, обратитесь к руководству по установке:

Сначала вам будет предложено создать пароль для хранилища ключей. Затем ответьте на остальные вопросы о хороших инструментах, и когда все будет готово, у вас должен быть

файл с именем my-release-key.keystore, созданный в текущем каталоге.

Примечание. Обязательно сохраните этот файл где-нибудь в безопасности, если вы его потеряете, вы не сможете отправлять обновления в свое приложение!

5. Подпишите APK

Чтобы подписать беззнаковый APK, запустите инструмент jarsigner, который также включен в JDK:

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

Это означает, что apk на месте. Наконец, нам нужно запустить инструмент выравнивания zip для оптимизации APK. Инструмент zipalign можно найти в / path / to / Android / sdk / build-tools / VERSION / zipalign.

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

Теперь у нас есть наш финальный бинарный релиз HelloWorld.apk, и мы можем опубликовать его в Google Play Store для всего мира!

Опубликуйте свое приложение в Google Play Store. Теперь, когда наша версия APK готова к использованию в Google Play Store, мы можем создать список Play Store и загрузить нашу APK. Для начала вам нужно посетить консоль разработчика Google Play Store и создать новую учетную запись разработчика. Это будет стоить 25 долларов за один раз.

После того, как у вас есть учетная запись разработчика, вы можете продолжить и нажать «Опубликовать приложение для Android в Google Play» и следовать инструкциям на экране.

Прочитайте От кода до магазина приложений - Android онлайн: https://riptutorial.com/ru/ionic2/topic/9659/от-кода-до-магазина-приложений---android

глава 16: Социальный Вход с Angularfire2 / Firebase

Examples

Родной Facebook Войти с 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));
      });
  }
}
```

Прочитайте Социальный Вход с Angularfire2 / Firebase онлайн:
https://riptutorial.com/ru/ionic2/topic/5518/социальный-вход-с-angularfire2---firebase

глава 17: Тестирование устройства

Вступление

Единичное тестирование в целом обеспечивает дополнительную безопасность продукта для предотвращения проблем при изменении / добавлении функций. Защитная сетка, в которой говорится: «ВСЕ ЕЩЕ ЕЩЕ». Тесты модулей никоим образом не заменяют фактические пользовательские тесты, которые может выполнять надлежащий QA.

В этом документе мы опишем примеры в этом репозитории: https://github.com/driftycoionion-unit-testing-example

Examples

Единичные тесты с кармой / жасмином

Единичное тестирование в ионном режиме такое же, как в любом угловом приложении.

Для этого мы будем использовать несколько фреймворков.

Карма - основа для проведения тестов

Жасмин - основа для написания тестов

PhantomJS - приложение, которое запускает javascript без браузера

Прежде всего, установите все, чтобы убедиться, что ваш пакет package.json содержит эти строки в зависимостях dev. Я чувствую, что важно отметить, что эти зависимости dev не влияют на ваше приложение вообще и находятся там, чтобы помочь разработчику.

```
"@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"
```

Чтобы немного погрузить пакеты

```
"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.
```

Мы также должны добавить этот скрипт в наши скрипты package.json:

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

Также обратите внимание на tsconfig, что вы исключаете файлы spec.ts из компиляции:

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

Итак, теперь давайте рассмотрим фактическую конфигурацию тестирования. Создайте папку `test-config` папке проекта. (Так же, как упоминалось в сценарии package.json) Внутри папки создайте 3 файла:

`webpack.test.js` - что скажет веб-пакет, какие файлы загружать для процесса тестирования

```
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` - который будет загружать библиотеки с угловыми `karma-test-shim.js` , такие как библиотеки зон и тестов, а также настроить модуль для тестирования.

```
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` - определяет конфигурацию тестирования с кармой. Здесь вы можете переключиться с Chrome на PhantomJS, чтобы сделать этот процесс невидимым и быстрым среди других вещей.

```
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);
};
```

Теперь, когда мы настроили все, вы можете написать какой-нибудь фактический тест. В этом примере мы напишем файл спецификации app.component. Если вы хотите увидеть тесты для страницы, а не основной компонент, вы можете посмотреть здесь: https://github.com/driftycoion/inic-unit-testing-example/blob/master/src/pages/page1/page1.spec.ts

Сначала нам нужно проверить наш конструктор. Это создаст и запустит конструктор нашего app.component

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

Декларация будет включать наше основное ионное приложение. Импорт будет импорт,

необходимый для этого теста. Не все.

Поставщики будут включать в себя вещи, которые вводятся в конструктор, но не являются частью импорта. Например, app.component внедряет службу платформы, но, поскольку она является частью IonicModule, нет необходимости упоминать ее в провайдерах.

Для следующих тестов нам нужно будет получить экземпляр нашего компонента:

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

Затем несколько тестов, чтобы увидеть, что все в порядке:

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

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

Поэтому в итоге у нас будет что-то вроде этого:

```
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);
  });

});
```

Запустите тесты

```
npm run test
```

И это все для базового тестирования. Существует несколько способов сократить написание тестов, например, написание собственного TestBed и наличие наследования в тестах, которые могут помочь вам в долгосрочной перспективе.

Прочитайте Тестирование устройства онлайн: https://riptutorial.com/ru/ionic2/topic/9561/тестирование-устройства

кредиты

S. No	Главы	Contributors
1	Начало работы с ionic2	Akilan Arasu, Cameron637, carstenbaumhoegger, Community, FreeBird72, Guillaume Le Mière, Ian Pinto, Ketan Akbari, misha130, Raymond Ativie, sebaferreras, tymspy, Will.Harris
2	Angularfire2 с Ionic2	Fernando Del Olmo
3	InAppBrowser	niks
4	Push-уведомление отправлено и получено	misha130
5	геолокации	Matyas, misha130
6	Добавить ионное приложение для ионного просмотра	Saravanan Sachi
7	Использование вкладок	misha130, sebaferreras
8	Использование сервисов	sebaferreras
9	Компоненты Ionic2 CSS	Ketan Akbari
10	Конструктор и OnInit	niks
11	модальности	Raymond Ativie
12	Настройка и отладка Ionic 2 в коде Visual Studio	misha130, PRIYA PARASHAR
13	Обходной путь для 'show-delete' в неодобрение	Amr ElAdawy, Roman Lee
14	От кода до	Luis Estevez, misha130

	магазина приложений - Android	
15	Социальный Вход с Angularfire2 / Firebase	Cameron637, Gianfranco P.
16	Тестирование устройства	misha130