FREE eBook

LEARNING ionic-framework

Free unaffiliated eBook created from
Stack Overflow contributors.

#ionic-framework

Table of Contents

About

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

It is an unofficial and free ionic-framework 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 ionic-framework.

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

Chapter 1: Getting started with ionic-framework

Remarks

Ionic is a framework for developing mobile applications with HTML, CSS, and JavaScript. Ionic applications run as native applications and have a native "look and feel".

Ionic is built on the AngularJS framework and provides a complete solution to design, build, and package mobile applications. Design is accomplished with a collection of template tools and a custom icon library. Ionic provides custom CSS/SASS components as well as Javascript UI Extensions. Ionic apps can be built, emulated, and packaged with their Command Line Interface (CLI).

Ionic templates are dynamic and responsive and adapt to their environment to provide a native "look and feel". This adaptation includes layout, style, and icons. Ionic makes independent platform customization available as well. Because Ionic apps use front end web technology, they can also be viewed in a browser for faster development.

Ionic apps are built on top of Apache Cordova by default. They have access to all Cordova Plugins which let you use native functionality, such as push notifications, camera, accelerometer, etc. Cordova apps work on multiple platforms and devices (phones, tablets, etc.) with very little extra effort. Cordova may be switched out with other cross-platform technologies such as trigger.io.

Versions

Version	Release Date
1.3.1 "el salvador"	2016-05-12
1.3.0 "delhi"	2016-04-21
1.2.0 "zirconium-zeren"	2015-12-09
1.1.0 "xenon-xerus"	2015-08-13
1.0.0 "uranium-unicorn"	2015-05-12

Examples

Installation or Setup

1. Install the Ionic Framework and Cordova (since Ionic apps

are based on Cordova) using npm (the Node Package Manager):

Make sure you have an up-to-date version of Node.js installed on your system. If you don't have Node.js installed, you can install it from here.

Also, for Mac users, having the latest Xcode version installed on your system brings to you command-line-tools and iOs Simulator, download here.

Open a terminal window (Mac) or a command window (Windows), and install Cordova and Ionic:

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

On a Mac, you may have to use sudo depending on your system configuration:

```
$ sudo npm install -g cordova ionic
```

If you already have Cordova and Ionic installed on your computer, make sure you update to the latest version:

```
$ npm update -g cordova ionic
```

or

```
$ sudo npm update -g cordova ionic
```

Follow the Android and iOS platform guides to install required platform dependencies.

Note: iOS development requires Mac OS X. iOS simulator through the Ionic CLI requires the ios-sim npm package, which can be installed with the command:

```
$ sudo npm -g install ios-sim
```

---

2. Start a new Ionic project:

Create an Ionic project using one of the ready-made app templates, or a blank one to start fresh.

```
$ ionic start myApp blank
```

or

```
$ ionic start myApp tabs
```

or

```
$ ionic start myApp sidemenu
```

---

3. Test the Ionic app:

To test your Ionic app in a desktop browser on both iOS and Android platforms:

```
$ ionic serve --lab
```

Whereas `ionic serve --lab` is great to test the app UI on multiple platforms it could lead to some problems for Javascript Console or Element Inspection, in that case what you may prefer:

```
$ ionic serve
```

To test your Ionic app in an emulator:

```
$ cd myApp
$ ionic platform add ios android
$ ionic build ios
$ ionic emulate ios
```

Substitute ios with android for Android emulator testing:

```
$ ionic build android
$ ionic emulate android
```

To test your Ionic app in an Android device connected via USB:

```
$ ionic run android
```

To test your Ionic app in an iOS device connected via USB:

```
$ ionic run ios --device
```

Ionic Framework Introduction and Installation and Setup

Ionic Framework

A Cross-platform mobile application development framework using Angular JS and Front End web technologies.

Official website: http://ionicframework.com/

Documentation: http://ionicframework.com/docs/

Installation and Setup

Installation Ionic required NPM(Node Package Manager) and Cordova.

You can download and install Npde JS from here which comes with NPM out of the Box.

To download Apache Cordova you can use NPM from command line

```
npm install -g cordova
```

If you already have NPM and Cordova then you can install ionic framework from Command line using following command.

```
npm install -g ionic
```

This will install and setup ionic framework for you to use it from command line.

Note* Based on your system environment you might need to execute with Admin privileges.

Starting A New Project

To Start a new Ionic Framework Project you can use following command

```
ionic start myproject
```

or

```
ionic start myproject [template_name]
```

Templates:

Ionic allows You to create project using some built in templates

`tabs`(default): which will create a simple app with tab view.

`sidemenu`: which will create ionic app with side menu.

`blank`: which will create a blank ionic app.

which will create a new folder named `myproject` with all the ionic project files.

to test project in your browser you can use following command

```
ionic serve --lab
```

or

```
ionic serve
```

Run an Emulate To execute or test app on emulator or phone first you will have to add platform for that you can use following command

```
ionic platform [Platform Name]
```

```
ionic build [Platform Name]
ionic emulate [platform name]
```

Platform Names you can directly mention android and ios for respective Platforms you can mention multiple platform names also separated by space.

To run your app you can use

```
ionic run [platform name]
```

For help you can use

```
ionic --help
```

or

```
ionic help
```

Refer this link for detailed explanation of ionic cli.

Refer this link for CSS components available in ionic.

Refer this link for Javascript API reference for ionic.

For quicker development with ionic you can try ionic Playground also.

Best of luck with ionic framework...

Ionic Framework Hello World App

After All the setup, To make Hello World App

- To create Simple Blank App, run below command on terminal :

```
ionic start HelloWorld blank  // create new project

cd HelloWorld                 // get into HelloWorld directory
```

- open the HelloWorld Project in subline/webstrome IDE :
    - Edit index.html, in www/ ditectory

```
<body ng-app="starter">
  <ion-pane>
    <ion-header-bar class="bar-stable">
      <h1 class="title">Ionic Hello World App</h1>
    </ion-header-bar>
    <ion-content>
       <div class="center">Hello World..!</div>
    </ion-content>
   </ion-pane>
</body>
```

- To run in browser from terminal :

```
ionic serve                 // run the app in browser
```

- To add platform :

```
ionic platform add android   // add android platform
ionic platform add ios       // add ios platform
```

- To run on device :

```
adb devices                  // to check devices is connected
ionic run android            // to run on android devices
ionic run ios                // to run on ios devices
```

- To run in livereload :

```
ionic run android -c -s -l    // to check app in live reload with console.
```

Ionic Platform (Ionic Cloud) for Yo (Yeoman) Ionic Projects

Ionic Platform:

Build, push, deploy, and scale your Ionic apps, the easy way.

Title Description:

The Ionic Platform is a cloud platform for managing and scaling cross-platform mobile apps. Integrated services enable you and your team to build, deploy, and grow your apps efficiently.

Document Objective:
Ionic Platform works well with the standard Ionic projects. But projects following any Non-standard Directory Structure may face a few hurdles. This documents provides the steps to use Ionic Platform in the Ionic projects created using Yeoman.

Document Scope:
This document covers the basic steps for creating an Ionic project using Yeoman and integrating it with Ionic Platform using the Ionic Platform Web Client. This document covers the basic steps to utilize Ionic Deploy, Ionic Analytics and Ionic Push.

Intended Audience:
The intended audience for this document is Web/Mobile App developers, with both beginner and expert level expertise, who are familiar with the below Prerequisites.

Prerequisites:
You should be familiar with the following frameworks/tools before trying this document.

- AngularJs: https://docs.angularjs.org/guide
- IonicFramework: http://ionicframework.com/docs/guide
- Yeoman: http://yeoman.io/codelab/index.html
- Ionic Generator: https://github.com/diegonetto/generator-ionic
- Ionic Platform: https://ionic.io/platform

Ionic Framework generator

A generator for the Ionic Framework from Yeoman, the Web's Scaffolding tool for modern webapps

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. npm is the package manager for JavaScript. Download and install Node (and npm) from http://nodejs.org

```
$ npm install npm -g
$ npm install -g yo
```

Yeoman helps you to kick-start new projects, prescribing best practices and tools to help you stay productive.

```
$ yo ionic [app-name]
```

In *package.json* include the following in devDependencies

```
"grunt-string-replace": "^1.2.1"
```

In *bower.json* include the following in dependencies

```
"ionic-platform-web-client": "^0.7.1"
```

In *Gruntfile.js* change the *scripts* folder to *'js'*. Change in *index.html* too if required.

```
grunt.initConfig({    yeoman: {……
    scripts: 'js',
    …… } })
```

Then run

```
$ bower install && npm install
```

```
$ grunt
$ grunt serve

$ cordova platform add android
$ grunt build:android --debug
```

---

ionic-platform-web-client

A web client that provides interactions with the Ionic platform.

We need some code to let your app talk to the Ionic Platform. We need to add the Ionic platform web client for the Ionic app to interface with the plugins and the Ionic.io platform.

```
$ ionic io init
```

In your *app.js* add the *'ionic.service.core'* module dependency. In *Gruntfile.js* add the grunt task *'ionicSettings'* as given below.

```
grunt.initConfig({
ionicSettings: JSON.stringify(grunt.file.readJSON('./.io-config.json')),

ionicIoBundlePath: 'www/bower_components/ionic-platform-web-
client/dist/ionic.io.bundle.min.js',

'string-replace': {
  ionicSettings: {
    files: {
      '<%= ionicIoBundlePath %>': '<%= ionicIoBundlePath %>',
    },
    options: {
      replacements: [
        {
          pattern:
        '"IONIC_SETTINGS_STRING_START";"IONIC_SETTINGS_STRING_END"',
          replacement:
        '"IONIC_SETTINGS_STRING_START";var settings =<%= ionicSettings %>; return { get:
function(setting) { if (settings[setting]) { return settings[setting]; } return null; }
};"IONIC_SETTINGS_STRING_END";'
        }
      ]
    }
  }
},
       copy: {
    ionicPlatform:{
                expand: true,
                cwd: 'app/bower_components/ionic-platform-web-client/dist/',
                src: ['**'],
```

```
                dest: 'www/bower_components/ionic-platform-web-client/dist'
              }
    }
});

grunt.registerTask('ionicSettings', ['copy:ionicPlatform','string-replace:ionicSettings']);
```

Add the *'ionicSettings'* in *init* and *compress* tasks after *copy*. In *index.html* move the below tag after all the tag declarations.

```
<script src="bower_components/ionic-platform-web-client/dist/ionic.io.bundle.min.js"></script>
```

Then run

```
$ Grunt serve
```

Ionic Deploy

Push real-time updates to your production apps, and manage version history.

Ionic Deploy lets you update your app on demand, for any changes that do not require binary modifications, saving you days, or even weeks, of wait time. Follow the below procedure to configure Ionic Deploy for your App.

In *Gruntfile.js* add the grunt task *'deploy'* as given below.

```
grunt.registerTask('deploy', function () {
  return grunt.task.run(['init', 'ionic:upload' + this.args.join()]);
});
```

then run

```
$ ionic plugin add ionic-plugin-deploy
```

Ionic Deploy Code:

```
var deploy = new Ionic.Deploy();

// Check Ionic Deploy for new code
deploy.check().then(function(hasUpdate) {
}, function(err) {
});
```

```
// Update app code with new release from Ionic Deploy
deploy.update().then(function(result) {
}, function(error) {
}, function(progress) {
});
```

Deploying Updates:

Send out new code for your app.

Create an apk and install your app. Make few changes in your code and deploy the changes using '*grunt deploy*'. Then update it from your app.

You can also deploy it from the *apps.ionic.io* dashboard. You can deploy the app without the deploy parameter. Then, in the dash board you can add the metadata and versioning details and deploy the app from there.

```
$ grunt build:android --debug

$ grunt deploy --note "release notes"
$ grunt deploy --note "release notes" --deploy=production
```

Ionic Analytics

View the live feed of events or the raw / unique number of events / users over time.

How many users are on your app right now? How many of those will use your app tomorrow, or next week? Without information, you have no way of telling if your app is being used in the ways that you expect. Follow the below procedure to configure Ionic Analytics for your App.

In your *app.js* add the '*ionic.service.analytics*' module dependency after the *ionic.service.core* Run the analytics register method in our module's run function.

```
$ionicAnalytics.register();
```

In Ionic Analytics, each tracked action a user makes in your app is represented by an event object. An event is a single action done at a specific point in time. To track your own events, call `$ionicAnalytics.track(eventType, eventData)` whenever an action occurs.

```
$ionicAnalytics.track('User Login', {
  user: $scope.user
});
```

The *ion-track-tap* directive sends an event when its host element is tapped. The associated *ion-track-data* directive attaches event data.

```
<button ion-track-tap="eventType" ion-track-data="expression"></button>
```

In the *apps.ionic.io* dashboard you can view the following analytics data,

Events: View the raw number of events over time, or the number of unique users who completed an event. An event can be anything from a user loading the app, to confirming a purchase.

Funnels: A funnel is a sequence of actions that you expect users to take in your app, leading up to a defined goal. Thoughtful use of funnels will let help you improve conversion rates.

Segments: View events over time, grouped by a specified property. Or, calculate the percentage of events that match a given property. Segments helps you understand your user base and see how properties change over time.

Retention: Track how long users are active on your app before they stop using it. Or, identify how long it takes for users to reach a defined goal, like a completed sale.

Pulse: A live feed of events coming in from your users.

Ionic Push

Send targeted and automated push notifications to your users.

Ionic Push lets you create targeted push notifications through a simple dashboard that will be sent automatically when users match specific criteria, and offers a simple API to send push notifications from your own servers.

Android Push Profiles:

Android push notifications use the *Google Cloud Messaging* (GCM) service. Open the *Google Developers Console* and create a project. Copy down your *project number*. This will be the *GCM sender ID* or ***GCM Project Number***.

In the *API Manager* section, enable the *Google Cloud Messaging API*. Then navigate to *Credentials* section and select Create credentials, then choose API Key, then Server Key. Name your API key and leave the *Accept requests from*... field blank and click *Create*. Save your ***API key***!

Authentication:

Go to your app's dashboard on the Ionic Platform and navigate to *Settings -> Certificates*. If you haven't already, create a new security profile, then hit *edit*. Note down the ***Profile Tag***.

Now, click the *Android* tab and find the section marked *Google Cloud Messaging*, enter the *API Key* you generated on the Google Developer Console, then click *Save*. Go to *Settings -> API Keys* . Under *API Tokens*, create a new token and copy it. This will your ***API Token***.

```
$ ionic plugin add phonegap-plugin-push --variable SENDER_ID="GCM_PROJECT_NUMBER"
$ ionic config set gcm_key <your-gcm-project-number>
$ ionic config set dev_push false
$ ionic io init
```

Note: phonegap-plugin-push requires Android Support Repository version 32+

In your *app.js* add the '*ionic.service.push*' module dependency after the *ionic.service.core*

Ionic Push Code:

Initialize the service and register your device in your module's run function. You'll need the device token that is registered by the user for sending notification to the user.

```
$ionicPush.init({
  debug: true,
  onNotification: function (notification) {
    console.log'token:', notification.payload);
  },
  onRegister: function (token) {
    console.log('Device Token:', token);
    $ionicPush.saveToken(token); // persist the token in the Ionic Platform
  }
});

$ionicPush.register();
```

then run

```
$ grunt build:android --debug
```

Ionic Push lets you create targeted push notifications through the dashboard. You can also send notifications from the server in the below format.

```
curl -X POST -H "Authorization: Bearer API_TOKEN" -H "Content-Type: application/json" -d '{
    "tokens": ["DEVICE_TOKEN"],
        "profile": "PROFILE_TAG",
        "notification": {
                "message": "Hello World!"
        "android": {
                  "title": "Hi User",
                  "message": "An update is available for your App",
                  "payload": {
                        "update": true
                  }
            }
    } }' "https://api.ionic.io/push/notifications"
```

Note: The steps to configure Ionic Push for iOS is the same except for creating the Push Profiles. To create iOS push profiles refer http://docs.ionic.io/v2.0.0-beta/docs/ios-push-profiles

---

Sample App

Download the sample app here.

A Sample app is attached here for reference.

```
IonicApp:
|
|   bower.json
|   Gruntfile.js
|   package.json
|
└──app
    |   index.html
    |
    ├──js
    |       app.js
    |       controllers.js
    |
    └──templates
            home.html
            menu.html
```

Note: This is not a standalone project. The code given is only for comparison against a project created and implemented using the procedures given above in this document, in case of any issues or errors.

Read Getting started with ionic-framework online: https://riptutorial.com/ionic-framework/topic/893/getting-started-with-ionic-framework

Chapter 2: Connecting Ionic with any database

Examples

You can't do it directly from Ionic framework

The thing is; you can't connect Ionic to any database (MySQL, Postgres, MSSQL, ...) **directly**. The keyword here is *directly*.

No, there's no workaround, no magic involved, it's just not the way this is supposed to work. Ionic works on top of Angular and Angular is a frontend framework.

However, the way you should do it is that you basically create a (RESTful) API on your server side.

Most likely this will be made with some serverside language (PHP, Go, Python, ...) which will talk directly to your database and query it.

After you write your (RESTful) API you can consume it through your services in Angular by using Angular's `$resource` or `$http` service.

An example of consuming Giphy API with Angular's `$http` service:

```
var search = 'cats';
var link = 'http://api.giphy.com/v1/gifs/search?api_key=dc6zaTOxFJmzC&amp;q=' + search;

$http.get(link).then(function(result){
    console.log(result);
});
```

Read Connecting Ionic with any database online: https://riptutorial.com/ionic-framework/topic/4003/connecting-ionic-with-any-database

Chapter 3: Create Dialog in Ionic

Parameters

Parameters	Detail
`title: '',`	// String. The title of the popup.
`cssClass: '',`	// String, The custom CSS class name
`subTitle: '',`	// String (optional). The sub-title of the popup.
`template: '',`	// String (optional). The html template to place in the popup body.
`templateUrl: '',`	// String (optional). The URL of an html template to place in the popup body.
`scope: null,`	// Scope (optional). A scope to link to the popup content.

Remarks

The Ionic Popup service allows programmatically creating and showing popup windows that require the user to respond in order to continue.

Examples

Create Dialog in Ionic

```
// An alert dialog
$scope.showAlert = function() {
  var alertPopup = $ionicPopup.alert({
    title: 'Don\'t eat that!',
    template: 'It might taste good'
  });

  alertPopup.then(function(res) {
    console.log('Hello your first example.');
  });
 };
});
```

Read Create Dialog in Ionic online: https://riptutorial.com/ionic-framework/topic/6461/create-dialog-in-ionic

Chapter 4: Deploy Ionic as a website

Examples

Simply copy the www folder to your web server

Ionic 1.2 officially supports deployment as a website

If you're not using any Cordova plugins then there is no problem (if you really wish to) to upload the contents of the `www` folder to your server, and woila - you'll have the same app.

However, it is important to note that Ionic 1 never intended for such a use, and the users of your "website" will have to have the newest browser in order to see the "website" correctly (not broken down due to some feature that Ionic is using in either CSS or HTML that some older browsers do not support).

Read Deploy Ionic as a website online: https://riptutorial.com/ionic-framework/topic/4001/deploy-ionic-as-a-website

Chapter 5: Device camera and photo library access from Ionic application

Remarks

also refer this link

Examples

Open camara and photo gallery

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="Content-Security-Policy" content="default-src *; script-src 'self'
'unsafe-inline' 'unsafe-eval' *; style-src  'self' 'unsafe-inline' *"/>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no,
width=device-width">
    <title></title>

    <link href="lib/ionic/css/ionic.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">

    <!-- IF using Sass (run gulp sass first), then uncomment below and remove the CSS includes
above
    <link href="css/ionic.app.css" rel="stylesheet">
    -->

    <!-- ionic/angularjs js -->
    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- cordova script (this will be a 404 during development) -->
    <script src="js/ng-cordova.min.js"></script>
    <script src="cordova.js"></script>

    <!-- your app's js -->
    <script src="js/app.js"></script>
    <script src="js/controllers.js"></script>
    <script src="js/services.js"></script>
  </head>
  <body ng-app="starter">
  <ion-content class="has-header padding" ng-controller="ImageController">
    <button class="button button-full button-energized" ng-click="addMedia()">
      Add image
    </button>
    <button class="button button-full button-positive" ng-click="sendEmail()">
      Send mail
    </button>
    <br><br>
    <ion-scroll direction="y" style="height:200px; min-height: 200px; overflow: scroll; white-
```

```
space: nowrap;">
      <img ng-repeat="image in images track by $index" ng-
src="{{image}}" style="height:200px; padding: 5px 5px 5px 5px;"/>
    </ion-scroll>

  </ion-content>
</body>
</html>
```

controller.js

```
angular.module('starter')

.controller('ImageController', function($scope, $cordovaDevice, $cordovaFile, $ionicPlatform,
$cordovaEmailComposer, $ionicActionSheet, ImageService, FileService) {

  $ionicPlatform.ready(function() {
    $scope.images = FileService.images();
    $scope.$apply();
  });

  $scope.addMedia = function() {
    $scope.hideSheet = $ionicActionSheet.show({
      buttons: [
        { text: 'Take photo' },
        { text: 'Photo from library' },
          {  text: '<b ng-disabled="user.length<1">Delete</b>',
              type: 'input type="file"'}
      ],
      titleText: 'Add images',
      cancelText: 'Cancel',
      buttonClicked: function(index) {
        $scope.addImage(index);
      }
    });
  }

  $scope.addImage = function(type) {
    $scope.hideSheet();
    ImageService.handleMediaDialog(type).then(function() {
      $scope.$apply();
    });
  }

  $scope.sendEmail = function() {
    if ($scope.images != null && $scope.images.length > 0) {
      var mailImages = [];
      var savedImages = $scope.images;
      for (var i = 0; i < savedImages.length; i++) {
          mailImages.push('base64:attachment'+i+'.jpg//' + savedImages[i]);
        }
      $scope.openMailComposer(mailImages);
    }
  }

  $scope.openMailComposer = function(attachments) {
    var bodyText = '<html><h2>My Images</h2></html>';
    var email = {
        to: '',
        attachments: attachments,
```

```
        subject: 'Devdactic Images',
        body: bodyText,
        isHtml: true
      };

    $cordovaEmailComposer.open(email, function(){
        console.log('email view dismissed');

    }, this);
  }
});
```

service.js

```
angular.module('starter')

.factory('FileService', function() {
  var images;
  var IMAGE_STORAGE_KEY = 'dav-images';

  function getImages() {
    var img = window.localStorage.getItem(IMAGE_STORAGE_KEY);
    if (img) {
      images = JSON.parse(img);
    } else {
      images = [];
    }
    return images;
  };

  function addImage(img) {
    images.push(img);
    window.localStorage.setItem(IMAGE_STORAGE_KEY, JSON.stringify(images));
  };

  return {
    storeImage: addImage,
    images: getImages
  }
})

.factory('ImageService', function($cordovaCamera, FileService, $q, $cordovaFile) {

  function optionsForType(type) {
    var source;
    switch (type) {
      case 0:
        source = Camera.PictureSourceType.CAMERA;
        break;
      case 1:
        source = Camera.PictureSourceType.PHOTOLIBRARY;
        break;
    }
    return {
      quality: 90,
      destinationType: Camera.DestinationType.DATA_URL,
      sourceType: source,
      allowEdit: false,
      encodingType: Camera.EncodingType.JPEG,
      popoverOptions: CameraPopoverOptions,
```

```
      saveToPhotoAlbum: false,
      correctOrientation:true
    };
  }

  function saveMedia(type) {
    return $q(function(resolve, reject) {
      var options = optionsForType(type);

      $cordovaCamera.getPicture(options).then(function(imageBase64) {
            FileService.storeImage(imageBase64);
      });
    })
  }
  return {
    handleMediaDialog: saveMedia
  }
});
```

For Ionic 3 Example

Install the Cordova and Ionic Native plugins:

```
$ ionic cordova plugin add cordova-plugin-camera
$ npm install --save @ionic-native/camera
```

Your `app.module.ts` will need to inject camera:

```
import { Camera } from '@ionic-native/camera';
..........
@NgModule({
  declarations: [
    MyApp
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    ...........
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp
  ],
  providers: [
    StatusBar,
    SplashScreen,
    Camera,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    ..........
  ]
})
export class AppModule {}
```

Camera can be used with Action sheet easily with Ionic 3, Your page.ts will be like below:

```
import { Camera, CameraOptions } from '@ionic-native/camera';
```

```
.........

export class YourPage {

    private base64:any;

constructor(private camera: Camera,private actionsheetCtrl: ActionSheetController) { }


cameragalleryfun(){

    let actionSheet = this.actionsheetCtrl.create({
      title: 'Camera-Gallery',
      cssClass: 'action-sheets-basic-page',
      buttons: [
        {
          text: 'Camera',
          icon: 'camera',
          handler: () => {
            //console.log('Camera');
            const options: CameraOptions = {
                    quality: 60,
                    destinationType: this.camera.DestinationType.DATA_URL,
                    encodingType: this.camera.EncodingType.JPEG,
                    mediaType: this.camera.MediaType.PICTURE,
                    sourceType : this.camera.PictureSourceType.CAMERA,
                    targetWidth: 500,
                    saveToPhotoAlbum: false,
                    correctOrientation:true
                  }

                     this.camera.getPicture(options).then((imageData) => {
                     this.base64 = '' + imageData;
                    }, (err) => {
                     // Handle error
                    });
          }
        },
        {
          text: 'Gallery',
          icon: 'images',
          handler: () => {
            //console.log('Gallery');
                const options: CameraOptions = {
                    quality: 60,
                    destinationType: this.camera.DestinationType.DATA_URL,
                    encodingType: this.camera.EncodingType.JPEG,
                    mediaType: this.camera.MediaType.PICTURE,
                    sourceType : this.camera.PictureSourceType.PHOTOLIBRARY,
                    targetWidth: 500,
                    saveToPhotoAlbum: false,
                    correctOrientation:true
                  }

                      this.camera.getPicture(options).then((imageData) => {
                      this.base64 = '' + imageData;
                    }, (err) => {
                     // Handle error
                    });
          }
        },
```

```
        {
          text: 'Cancel',
          role: 'cancel', // will always sort to be on the bottom
          icon: 'close',
          handler: () => {
            //console.log('Cancel clicked');
          }
        }
      ]
    });
    actionSheet.present();

  }
}
```

Call `cameragalleryfun` function from any event like click on button, this will return base64 string for image. More option can be applied. for extra option see:https://github.com/apache/cordova-plugin-camera

Read Device camera and photo library access from Ionic application online: https://riptutorial.com/ionic-framework/topic/7086/device-camera-and-photo-library-access-from-ionic-application

Chapter 6: How to use EcmaScript 6 features in Ionic?

Examples

By using gulp-babel and gulp-plumber

Ionic uses Gulp, so install gulp-babel and gulp-plumber.

```
npm install --save-dev gulp-babel gulp-plumber
```

Add babel to `gulpfile.js` like so:

```
//...
var babel = require("gulp-babel");
var plumber = require("gulp-plumber");

var paths = {
  es6: ['./src/es6/*.js'],
  sass: ['./scss/**/*.scss']
};

gulp.task('default', ['babel', 'sass']);

gulp.task("babel", function () {
  return gulp.src(paths.es6)
    .pipe(plumber())
    .pipe(babel())
    .pipe(gulp.dest("www/js"));
});

//...

gulp.task('watch', function() {
  gulp.watch(paths.es6, ['babel']);
  gulp.watch(paths.sass, ['sass']);
});
//...
```

Edit `ionic.project`:

```
"gulpStartupTasks": [
    "babel",
    "sass",
    "watch"
 ],
```

Now when you run `ionic serve`, code will be transpiled for you.

Read How to use EcmaScript 6 features in Ionic? online: https://riptutorial.com/ionic-framework/topic/4847/how-to-use-ecmascript-6-features-in-ionic-

Chapter 7: Ionic - Analyze your app with jshint and gulp-jshint as part of your build process

Remarks

Linting your ionic app before running has huge advantages. It will analyse code for potential errors and save you tremendous amount of time.

What is linting and how to install the required packages?

"Linting is the process of running a program that will analyse code for potential errors." - see What is "Linting"?

Your ionic app comes with a package.json file. Go to the root of you app in a Command Line/Terminal and install the following packages:

```
npm install jshint --save-dev
npm install jshint-stylish --save-dev
npm install gulp-jshint --save-dev
```

Examples

Add a gulp task

In the root of your ionic app, there is a gulpfile.js file. Open it in an editor and paste the following gulp task:

```
gulp.task('lint', function() {
    return gulp.src(['./www/js/**/*.js'])
        .pipe(jshint('.jshintrc'))
        .pipe(jshint.reporter('jshint-stylish'))
        .pipe(jshint.reporter('fail'))
});
```

This looks for a folder called 'js' inside the 'www' folder. If you have other folders containing JavaScript files, add those too. For example, lets also add a folder called 'views':

```
gulp.task('lint', function() {
    return gulp.src(['./www/js/**/*.js','./www/views/**/*.js'])
        .pipe(jshint('.jshintrc'))
        .pipe(jshint.reporter('jshint-stylish'))
        .pipe(jshint.reporter('fail'))
});
```

Explanations:

```
1) /**/*.js - This syntax means to look at all the js files in the subfolders too
2) .jshintrc - This is a configuration file that we will create in the next example.
```

Create .jshintrc file (Optional)

Create a file named '.jshintrc' in the root of your app, where package.json is.

*Note on windows: create a file named "jshintrc.txt". Then rename it to ".jshintrc." (notice the dot at the end).

This is a configuration file. It can for example tell jshint to ignore certain variables and many other things. Here is mine:

```
{
    "predef": [
        "window",
        "console",
        "cordova",
        "device",
        "alert",
        "document",
        "debug",
        "setServiceVars",
        "StatusBar",
        "config"
    ],
    "globals": {
        "angular"        : false,
        "myApp"            : false,
        "myControllers"    : false,
        "myDirectives"    : false,
        "localStorage"     : false,
        "navigator"         : false,
        "emit"             : false,
        "atob"             : false,
        "moment"        : false,
        "btoa"            : false
    },
    "node"          : true
}
```

Add Makefile

1. Create a file named: "Makefile" (with no extension) in the root of your app

2. Open it in a text editor and add this:

```
android:
    gulp lint
    gulp sass
    ionic run android --device

ios:
```

```
    gulp lint
    gulp sass
    ionic build ios
```

This will lint your app and if that passes, it will compile sass and build you app.

Usage: To run your app, instead of the regular "ionic run android --device", run these commands:

```
Android: make android
iOS    : make ios
```

Read Ionic - Analyze your app with jshint and gulp-jshint as part of your build process online: https://riptutorial.com/ionic-framework/topic/4889/ionic---analyze-your-app-with-jshint-and-gulp-jshint-as-part-of-your-build-process

Chapter 8: Ionic AngularJS extensions

Remarks

Ionic offers a variety of Javascript AngularJS extensions for you to use. These extensions can be anything from normal form inputs to modal windows and makes coding your basic app a lot faster using these ready extensions.

Examples

Form inputs

Ionic inputs are no different from normal HTML inputs but they are styled differently and used as a directive. You can also use normal HTML inputs in Ionic apps. Here are some basic examples that Ionic offers ready-to-go.

Checkbox:

```
<ion-checkbox ng-model="checkbox">Label</ion-checkbox>
```

Radio button:

```
<ion-radio ng-model="radio" ng-value="'radiovalue'">Value</ion-radio>
```

Toggle:

```
<ion-toggle ng-model="toggle" toggle-class="toggle-calm">Toggle</ion-toggle>
```

Modal windows

Ionic has it's own extension for displaying a modal window. Modals can be created by inserting the template straight to the view with a `<script>` tag or by using a separate template file. In this example we are assuming you have a html file named `modal-template.html` in a folder called `templates`. You set the template path in the modal initialisation function with `fromTemplateUrl`.

Creating the modal object in the scope

HTML

```
<ion-modal-view>
  <ion-header-bar>
    <h1>Modal title</h1>
  </ion-header-bar>
  <ion-content>
```

```
    <p>Modal content</p>
  </ion-content>
</ion-modal-view>
```

Controller

```
$ionicModal.fromTemplateUrl('/templates/modal-template.html', {
  scope: $scope,
  animation: 'slide-in-up'
}).then(function(modal) {
  $scope.modal = modal;
});
```

Control the modal

You can then control the modal with the following commands.

Open modal

```
$scope.modal.show();
```

Close modal

```
$scope.modal.hide();
```

Remove modal

```
$scope.modal.remove();
```

Modal events

You can listen to modal events with the following functions.

Modal is hidden

```
$scope.$on('modal.hidden', function() {
  // Do something when modal is hidden
});
```

Modal is removed

```
$scope.$on('modal.removed', function() {
  // Do something when modal is removed
});
```

Read Ionic AngularJS extensions online: https://riptutorial.com/ionic-framework/topic/6446/ionic-angularjs-extensions

Chapter 9: Ionic Backend Services (ionic.io)

Examples

Introduction and setup

The Ionic Platform offers a range of powerful, hybrid-focused mobile backend services and tools to make it easy to scale beautiful, performant hybrid apps, at a rapid pace.

In order to use Ionic Platform you need to have the Ionic Framework installed.

1.) **Registration** (sign-up)

You need to enter: name, username, company, phone (optional), email and password. After you receive confirmation email you may log-in.

> Welcome to Ionic. Let's create your first app! Create a new app above, then follow the Quick Start guide to upload your first app.

2.) **Setup**

First, create your firs Ionic app. Now you can add the platform web client:

> The web client gives you a way to interact with the Ionic Platform services from within your app code.

```
$ ionic add ionic-platform-web-client
```

Now, you need the Platform to assign your app a unique app id and api key. To do that use the `ionic io init` command.

```
$ ionic io init
```

> This will automatically prompt you to login to your Platform account. The app id and api key will then be stored in your project's Ionic Platform config.
>
> Your app is now hooked up to the Ionic Platform and will be listed in your Dashboard.

3.) **Now, you can move on to installing one of Ionic Platform services**:

- Users / Auth
- Deploy
- Push
- Package
- Analytics

Read Ionic Backend Services (ionic.io) online: https://riptutorial.com/ionic-

framework/topic/3622/ionic-backend-services--ionic-io-

Chapter 10: Ionic CLI hooks

Remarks

Introduction

Hooks are pieces of code that Cordova CLI executes at certain points in your Cordova/Ionic application build. Hooks can be used for example to manipulate files in our project, automatically add plugins into your application or as in the example above check for code errors in your files.

Note: It is highly recommended writing your hooks using Node.js so that they are cross-platform but you can write them also for example in Javascript.

Hook types

The following hook types are supported and execution order is quite self-explanatory according to the name.

```
after_build
after_compile
after_docs
after_emulate
after_platform_add
after_platform_rm
after_platform_ls
after_plugin_add
after_plugin_ls
after_plugin_rm
after_plugin_search
after_prepare
after_run
after_serve
before_build
before_compile
before_docs
before_emulate
before_platform_add
before_platform_rm
before_platform_ls
before_plugin_add
before_plugin_ls
before_plugin_rm
before_plugin_search
before_prepare
before_run
before_serve
pre_package/ <-- Applicable to Windows 8 and Windows Phone only. This hook is deprecated.
```

Ways to define hooks:

Hooks could be defined in project's `config.xml` using `<hook>` elements, for example:

```
<hook type="after_build" src="scripts/appAfterBuild.js" />
```

As a plugin developer you can define hook scripts using `<hook>` elements in a `plugin.xml` like this:

```
<hook type="after_build" src="scripts/afterBuild.js" />
```

`before_plugin_install`, `after_plugin_install`, `before_plugin_uninstall` plugin hooks will be fired exclusively for the plugin being installed/uninstalled.

Note: Placing hooks in the `root/hooks` directory is considered deprecated in favor of the hook elements in `config.xml` and `plugin.xml`. If you however use this approach remember to set execute rights to the files in the `root/hooks` folder.

Documentation for Cordova Hooks can be found here.

Examples

Checking for errors in your Javascript files in before_prepare using jshint

```
#!/usr/bin/env node

var fs = require('fs');
var path = require('path');
var jshint = require('jshint').JSHINT;
var async = require('async');

var foldersToProcess = [
  'js'
];

foldersToProcess.forEach(function(folder) {
  processFiles("www/" + folder);
});

function processFiles(dir, callback) {
  var errorCount = 0;
  fs.readdir(dir, function(err, list) {
    if (err) {
      console.log('processFiles err: ' + err);
      return;
    }
    async.eachSeries(list, function(file, innercallback) {
      file = dir + '/' + file;
      fs.stat(file, function(err, stat) {
        if (!stat.isDirectory()) {
          if (path.extname(file) === ".js") {
            lintFile(file, function(hasError) {
              if (hasError) {
```

```
                errorCount++;
              }
              innercallback();
            });
          } else {
            innercallback();
          }
        } else {
          processFiles(file);
        }
      });
    }, function(error) {
      if (errorCount > 0) {
        process.exit(1);
      }
    });
  });
}

function lintFile(file, callback) {
  console.log("Linting " + file);
  fs.readFile(file, function(err, data) {
    if (err) {
      console.log('Error: ' + err);
      return;
    }
    if (jshint(data.toString())) {
      console.log('File ' + file + ' has no errors.');
      console.log('------------------------------------------');
      callback(false);
    } else {
      console.error('Errors in file ' + file);
      var out = jshint.data(),
          errors = out.errors;
      for (var j = 0; j < errors.length; j++) {
        console.error(errors[j].line + ':' + errors[j].character + ' -> ' + errors[j].reason +
' -> ' + errors[j].evidence);
      }
      console.error('------------------------------------------');
      setTimeout(function() {
        callback(true);
      }, 10);
    }
  });
}
```

Read Ionic CLI hooks online: https://riptutorial.com/ionic-framework/topic/6520/ionic-cli-hooks

Chapter 11: Ionic Cloud for Yeoman Ionic Projects

Examples

Ionic Platform (Ionic Cloud) for Yo (Yeoman) Ionic Projects

Ionic Platform:

Build, push, deploy, and scale your Ionic apps, the easy way.

Title Description:

The Ionic Platform is a cloud platform for managing and scaling cross-platform mobile apps. Integrated services enable you and your team to build, deploy, and grow your apps efficiently.

Document Objective:
Ionic Platform works well with the standard Ionic projects. But projects following any Non-standard Directory Structure may face a few hurdles. This documents provides the steps to use Ionic Platform in the Ionic projects created using Yeoman.

Document Scope:
This document covers the basic steps for creating an Ionic project using Yeoman and integrating it with Ionic Platform using the Ionic Platform Web Client. This document covers the basic steps to utilize Ionic Deploy, Ionic Analytics and Ionic Push.

Intended Audience:
The intended audience for this document is Web/Mobile App developers, with both beginner and expert level expertise, who are familiar with the below Prerequisites.

Prerequisites:
You should be familiar with the following frameworks/tools before trying this document.

- AngularJs: https://docs.angularjs.org/guide
- IonicFramework: http://ionicframework.com/docs/guide
- Yeoman: http://yeoman.io/codelab/index.html
- Ionic Generator: https://github.com/diegonetto/generator-ionic
- Ionic Platform: https://ionic.io/platform

Ionic Framework generator

A generator for the Ionic Framework from Yeoman, the Web's Scaffolding tool for modern webapps

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. npm is the package manager for JavaScript. Download and install Node (and npm) from http://nodejs.org

```
$ npm install npm -g
$ npm install -g yo
```

Yeoman helps you to kick-start new projects, prescribing best practices and tools to help you stay productive.

```
$ yo ionic [app-name]
```

In *package.json* include the following in devDependencies

```
"grunt-string-replace": "^1.2.1"
```

In *bower.json* include the following in dependencies

```
"ionic-platform-web-client": "^0.7.1"
```

In *Gruntfile.js* change the *scripts* folder to *'js'*. Change in *index.html* too if required.

```
grunt.initConfig({   yeoman: {……
    scripts: 'js',
    …… } })
```

Then run

```
$ bower install && npm install
$ grunt
$ grunt serve

$ cordova platform add android
$ grunt build:android --debug
```

---

ionic-platform-web-client

A web client that provides interactions with the Ionic platform.

We need some code to let your app talk to the Ionic Platform. We need to add the Ionic platform web client for the Ionic app to interface with the plugins and the Ionic.io platform.

```
$ ionic io init
```

In your *app.js* add the *'ionic.service.core'* module dependency. In *Gruntfile.js* add the grunt task *'ionicSettings'* as given below.

```
grunt.initConfig({
ionicSettings: JSON.stringify(grunt.file.readJSON('./.io-config.json')),

ionicIoBundlePath: 'www/bower_components/ionic-platform-web-
client/dist/ionic.io.bundle.min.js',

'string-replace': {
  ionicSettings: {
    files: {
      '<%= ionicIoBundlePath %>': '<%= ionicIoBundlePath %>',
    },
    options: {
      replacements: [
        {
          pattern:
        '"IONIC_SETTINGS_STRING_START";"IONIC_SETTINGS_STRING_END"',
          replacement:
        '"IONIC_SETTINGS_STRING_START";var settings =<%= ionicSettings %>; return { get:
function(setting) { if (settings[setting]) { return settings[setting]; } return null; }
};"IONIC_SETTINGS_STRING_END";'
        }
      ]
    }
  }
},
       copy: {
    ionicPlatform:{
                expand: true,
                cwd: 'app/bower_components/ionic-platform-web-client/dist/',
                src: ['**'],
                dest: 'www/bower_components/ionic-platform-web-client/dist'
               }
    }
});

grunt.registerTask('ionicSettings', ['copy:ionicPlatform','string-replace:ionicSettings']);
```

Add the *'ionicSettings'* in *init* and *compress* tasks after *copy*. In *index.html* move the below tag after all the tag declarations.

```
<script src="bower_components/ionic-platform-web-client/dist/ionic.io.bundle.min.js"></script>
```

Then run

```
$ Grunt serve
```

Ionic Deploy

Push real-time updates to your production apps, and manage version history.

Ionic Deploy lets you update your app on demand, for any changes that do not require binary modifications, saving you days, or even weeks, of wait time. Follow the below procedure to configure Ionic Deploy for your App.

In *Gruntfile.js* add the grunt task *'deploy'* as given below.

```
grunt.registerTask('deploy', function () {
  return grunt.task.run(['init', 'ionic:upload' + this.args.join()]);
});
```

then run

```
$ ionic plugin add ionic-plugin-deploy
```

Ionic Deploy Code:

```
var deploy = new Ionic.Deploy();

// Check Ionic Deploy for new code
deploy.check().then(function(hasUpdate) {
}, function(err) {
});

// Update app code with new release from Ionic Deploy
deploy.update().then(function(result) {
}, function(error) {
}, function(progress) {
});
```

Deploying Updates:

Send out new code for your app.

Create an apk and install your app. Make few changes in your code and deploy the changes using

'*grunt deploy*'. Then update it from your app.

You can also deploy it from the *apps.ionic.io* dashboard. You can deploy the app without the deploy parameter. Then, in the dash board you can add the metadata and versioning details and deploy the app from there.

```
$ grunt build:android --debug

$ grunt deploy --note "release notes"
$ grunt deploy --note "release notes" --deploy=production
```

Ionic Analytics

View the live feed of events or the raw / unique number of events / users over time.

How many users are on your app right now? How many of those will use your app tomorrow, or next week? Without information, you have no way of telling if your app is being used in the ways that you expect. Follow the below procedure to configure Ionic Analytics for your App.

In your *app.js* add the '*ionic.service.analytics*' module dependency after the *ionic.service.core* Run the analytics register method in our module's run function.

```
$ionicAnalytics.register();
```

In Ionic Analytics, each tracked action a user makes in your app is represented by an event object. An event is a single action done at a specific point in time. To track your own events, call `$ionicAnalytics.track(eventType, eventData)` whenever an action occurs.

```
$ionicAnalytics.track('User Login', {
  user: $scope.user
});
```

The *ion-track-tap* directive sends an event when its host element is tapped. The associated *ion-track-data* directive attaches event data.

```
<button ion-track-tap="eventType" ion-track-data="expression"></button>
```

In the *apps.ionic.io* dashboard you can view the following analytics data,

Events: View the raw number of events over time, or the number of unique users who completed an event. An event can be anything from a user loading the app, to confirming a purchase.

Funnels: A funnel is a sequence of actions that you expect users to take in your app, leading up to a defined goal. Thoughtful use of funnels will let help you improve conversion rates.

Segments: View events over time, grouped by a specified property. Or, calculate the percentage of events that match a given property. Segments helps you understand your user base and see how properties change over time.

Retention: Track how long users are active on your app before they stop using it. Or, identify how long it takes for users to reach a defined goal, like a completed sale.

Pulse: A live feed of events coming in from your users.

Ionic Push

Send targeted and automated push notifications to your users.

Ionic Push lets you create targeted push notifications through a simple dashboard that will be sent automatically when users match specific criteria, and offers a simple API to send push notifications from your own servers.

Android Push Profiles:

Android push notifications use the *Google Cloud Messaging* (GCM) service. Open the *Google Developers Console* and create a project. Copy down your *project number*. This will be the *GCM sender ID* or ***GCM Project Number***.

In the *API Manager* section, enable the *Google Cloud Messaging API*. Then navigate to *Credentials* section and select Create credentials, then choose API Key, then Server Key. Name your API key and leave the *Accept requests from*... field blank and click *Create*. Save your ***API key***!

Authentication:

Go to your app's dashboard on the Ionic Platform and navigate to *Settings -> Certificates*. If you haven't already, create a new security profile, then hit *edit*. Note down the ***Profile Tag***.

Now, click the *Android* tab and find the section marked *Google Cloud Messaging*, enter the *API Key* you generated on the Google Developer Console, then click *Save*. Go to *Settings -> API Keys* . Under *API Tokens*, create a new token and copy it. This will your ***API Token***.

```
$ ionic plugin add phonegap-plugin-push --variable SENDER_ID="GCM_PROJECT_NUMBER"
$ ionic config set gcm_key <your-gcm-project-number>
$ ionic config set dev_push false
```

```
$ ionic io init
```

Note: phonegap-plugin-push requires Android Support Repository version 32+

In your *app.js* add the '*ionic.service.push*' module dependency after the *ionic.service.core*

Ionic Push Code:

Initialize the service and register your device in your module's run function. You'll need the device token that is registered by the user for sending notification to the user.

```
$ionicPush.init({
  debug: true,
  onNotification: function (notification) {
    console.log'token:', notification.payload);
  },
  onRegister: function (token) {
    console.log('Device Token:', token);
    $ionicPush.saveToken(token); // persist the token in the Ionic Platform
  }
});

$ionicPush.register();
```

then run

```
$ grunt build:android --debug
```

Ionic Push lets you create targeted push notifications through the dashboard. You can also send notifications from the server in the below format.

```
curl -X POST -H "Authorization: Bearer API_TOKEN" -H "Content-Type: application/json" -d '{
    "tokens": ["DEVICE_TOKEN"],
        "profile": "PROFILE_TAG",
        "notification": {
                "message": "Hello World!"
        "android": {
                  "title": "Hi User",
                  "message": "An update is available for your App",
                  "payload": {
                        "update": true
                  }
            }
    } }' "https://api.ionic.io/push/notifications"
```

Note: The steps to configure Ionic Push for iOS is the same except for creating the Push Profiles. To create iOS push profiles refer http://docs.ionic.io/v2.0.0-beta/docs/ios-push-profiles

---

Sample App

Download the sample app here.

A Sample app is attached here for reference.

```
IonicApp:
|
|   bower.json
|   Gruntfile.js
|   package.json
|
└──app
    |   index.html
    |
    ├──js
    |       app.js
    |       controllers.js
    |
    └──templates
            home.html
            menu.html
```

Note: This is not a standalone project. The code given is only for comparison against a project created and implemented using the procedures given above in this document, in case of any issues or errors.

Read Ionic Cloud for Yeoman Ionic Projects online: https://riptutorial.com/ionic-framework/topic/6389/ionic-cloud-for-yeoman-ionic-projects

Chapter 12: Ionic CSS components

Remarks

Ionic has a lot of great ready declared CSS components to make your life easier while coding your next hybrid mobile application. These components vary from a basic grid system to styling your forms. These components are in your use if you choose to install Ionic with the pre-set CSS stylesheets.

One of the basic functions that Ionic CSS brings to your hand is that it comes with a set of colors to start with, but as a general rule colors are meant to be overridden. Utility colors are added to help set a naming convention. You could call it a basic theme of the application. To customize the colors you can simply override those coming from the ionic.css CSS file. Additionally, since Ionic is built using Sass, for more power and flexibility you could also modify and extend the color variables within the _variables.scss file.

You can setup an Ionic project to use SASS very easily by running the `ionic setup sass` command in your terminal.

You can find the official documentation on Ionic CSS here:
http://ionicframework.com/docs/components/

Examples

Basic grid system syntax

Basic grid

In Ionic you can declare rows by setting the `row` class to a element. Rows will be elements which are horizontally aligned and anything inside this element everything will belong to the row. Inside the row you can declare different width columns. You have a choice of the following declarations.

Class	Width
.col-10	10%
.col-20	20%
.col-25	25%
.col-33	33.3333%
.col-50	50%
.col-67	66.6666%

Class	Width
.col-75	75%
.col-80	80%
.col-90	90%

The maximum value columns may have inside a row is 100. Here's a few examples of the basic grid.

```
<div class="row">
  <div class="col col-50">.col.col-50</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col col-75">.col.col-75</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col col-75">.col.col-75</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

Offset grids

You can also set `col-offset-<value>` to the columns. In the example below the one-third of a width column has one-third of the width offset, which means it will be one-third width wide and be centered in the page because of it's offset.

```
<div class="row">
  <div class="col col-33 col-offset-33">.col</div>
  <div class="col">.col</div>
</div>
```

Align columns

Aligning columns vertically is possibly by setting the `col-<align_value>` to a column separately like this.

```
<div class="row">
  <div class="col col-top">.col</div>
```

```
  <div class="col col-center">.col</div>
  <div class="col col-bottom">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>
```

The above will align each column on it's own. If you want to align all the columns inside the row you can set the align value to the row itself. In the example below all the columns inside this row will align themselves vertically in the center of the row.

```
<div class="row row-center">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>
```

Responsive grid

You might also want to have the columns responsive as of they will stack on top of each other at some viewport width. You have three choices.

Class	Breakpoint (approximately)
.responsive-sm	Smaller than landscape phone
.responsive-md	Smaller than portrait tablet
.responsive-lg	Smaller than landscape tablet

In this example these columns will stack under the width of approximately a landscape phone.

```
<div class="row responsive-sm">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

You can also make your own media queries of course if these breakpoints are not suitable for you and/or you need more specific breakpoints.

Basic list item syntax

Almost every application has some kind of a list. Ionic has it's own build-in ready-to-go list item CSS declarations to make it easy to do lists inside your application. You can either use HTML elements and declare a class for the or use the directive `ion-list` to make them. Example of a directive is at the bottom.

Basic list item CSS syntax:

```
<ul class="list">
  <li class="item"></li>
</ul>
```

List with dividers:

```
<div class="list">
  <a class="item" href="#">
    List item
  </a>
  <div class="item item-divider">
    Divider that looks a bit different from items
  </div>
  <a class="item" href="#">
    Another list item
  </a>
</div>
```

List items with icons:

```
<div class="list">
  <a class="item item-icon-left" href="#">
    <i class="icon ion-trash-b"></i>
    List item with a trashcan icon on the left
  </a>
</div>
```

You can also set icons on both sides of the items with the following syntax:

```
<div class="list">
  <a class="item item-icon-left item-icon-right" href="#">
    <i class="icon ion-trash-b"></i>
    List item with a trashcan icon on the left and a briefcase icon on the right
    <i class="icon ion-briefcase"></i>
  </a>
</div>
```

An list item with button or buttons can be created like this:

```
<div class="list">
  <div class="item item-button-right">
    Item with a button on the right that has a clock icon in it
    <button class="button button-positive">
      <i class="icon ion-clock"></i>
    </button>
  </div>
</div>
```

It's also possible to create list items with avatars, thumbnails and inset which will create padding around the list items. Ionic also handles setting icons etc in list items by setting padding accordingly to the list items.

Ionic also has it's own directives for checkboxes, radio buttons etc. Here's an example of a checkbox list with Ionic.

```
<ion-list>
  <ion-checkbox ng-model="choice1">Choice 1</ion-checkbox>
  <ion-checkbox ng-model="choice2">Choice 2</ion-checkbox>
</ion-list>
```

Basic usage of utility colors

Preset Ionic CSS will have a theme and pre-set colors for it. You can modify or override the basic values in the ionic.css or in your custom CSS file. You can also define these with SASS and to use SASS in Ionic you just need to run the `ionic setup sass` command in your terminal.

Basic usage of colors in a button. The `button-<phrase>` class will make the button background and borders the color of the set theme.

```
<button class="button button-positive">
  button-positive
</button>

<button class="button button-calm">
  button-calm
</button>

<button class="button button-balanced">
  button-balanced
</button>
```

Your CSS prefix choices are the following:

- `<element>-light`
- `<element>-stable`
- `<element>-positive`
- `<element>-calm`
- `<element>-balanced`
- `<element>-energized`
- `<element>-assertive`
- `<element>-royal`
- `<element>-dark`

These classes can be added also for example in badges etc. Here's an example of a badge:

```
<span class="badge badge-positive">Positive badge</span>
```

Read Ionic CSS components online: https://riptutorial.com/ionic-framework/topic/6689/ionic-css-components

Chapter 13: Ionic infinite scroll to show load items on demand (Already available data not by Http Request)

Examples

Load n number of available data on demand

HTML :

```
<li class="item" ng-repeat="schedule in Schedules | filter:scheduleSearch |
limitTo:numberOfItemsToDisplay">
    Display some data
</li>
<ion-infinite-scroll on-infinite="addMoreItem()" ng-if="Schedules.length >
numberOfItemsToDisplay"></ion-infinite-scroll>
```

Controller :

```
$scope.numberOfItemsToDisplay = 10; // Use it with limit to in ng-repeat
$scope.addMoreItem = function(done) {
    if ($scope.Schedules.length > $scope.numberOfItemsToDisplay)
        $scope.numberOfItemsToDisplay += 10; // load number of more items
        $scope.$broadcast('scroll.infiniteScrollComplete')
}
```

Load 10 items every time addMoreItem() call.

Read Ionic infinite scroll to show load items on demand (Already available data not by Http Request) online: https://riptutorial.com/ionic-framework/topic/9490/ionic-infinite-scroll-to-show-load-items-on-demand--already-available-data-not-by-http-request-

Chapter 14: Ionicons

Remarks

In the modern web development it's common practice to use fonts to display icons. Since fonts are vectors, they are resolution independent and can be easily colored through CSS, just to name a few advantages compared to bitmap images etc. Ionicons was created by the same team that Ionic Framework was created by and can be used in any project since they are 100% free and open source. MIT Licensed.

Ionicons can be used on their own or with Ionics CSS components when they have certain styles according to the parent elements.

The homepage and list of the icons can be found here: http://ionicons.com/

Examples

Basic usage

Font icons are usually placed inside a `<i>` tag. Ionic has default css styles for the icons for easy use. The most basic example of use:

```
<i class="icon ion-home"></i>
```

Extended usage

Ionic has some CSS components where you can use Ionicons as a default which have preset styling. The `range` class in the item `<div>` will apply correct styling to both the input and the icons inside it. Here's an example of a range slider.

```
<div class="item range">
  <i class="icon ion-volume-low"></i>
  <input type="range" name="volume">
  <i class="icon ion-volume-high"></i>
</div>
```

Another example of Ionicon usage in Ionic tabs which will create a tab like menu. The `tabs-striped tabs-color-assertive` classes define the style of the tabs themselves. Icons are used with simple `<i>` tags and they get their positional styling from the classes applied to the parent divs.

```
<div class="tabs-striped tabs-color-assertive">
  <div class="tabs">
    <a class="tab-item" href="#">
      <i class="icon ion-home"></i>
      Home
    </a>
    <a class="tab-item" href="#">
```

```
      <i class="icon ion-gear-b"></i>
      Settings
    </a>
</div>
```

Read Ionicons online: https://riptutorial.com/ionic-framework/topic/5886/ionicons

Chapter 15: Publishing your Ionic app

Examples

Building release version from macOS

Build the APK

First of all we need to build the APK.

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

Generate private Key

Then we will create a keystore to sign the APK. `keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000`

- Change **my-release-key** with your key name.
- Change **alias_name** with your key alias.

Sign the APK

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

- Change **my-release-key** with your key name.
- Change **HelloWorld-release-unsigned** with your unsigned apk. `ionic-project/platforms/android/build/outputs/apk`.
- Change alias_name with your key alias.

Zip the APK

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

- You can find zipalign in `/Users/username/Library/Android/sdk/build-tools/XXX/`
- Change **HelloWorld-release-unsigned** with your unsigned apk. `ionic-project/platforms/android/build/outputs/apk`.
- Change **HelloWorld.apk** with your prefered apk file name. This will be uploaded to Google Play.

Read Publishing your Ionic app online: https://riptutorial.com/ionic-framework/topic/7755/publishing-your-ionic-app

Chapter 16: Run Ionic App on Emulator or on your Phone

Examples

Run Ionic App on Emulator or on your Phone

1. Add a platform target

iOS:

```
$ ionic platform add ios
```

Android:

```
$ ionic platform add android
```

Windows:

```
$ ionic platform add windows
```

---

2. Build your app

iOS:

```
$ ionic build ios
```

Android:

```
$ ionic build android
```

Windows:

```
$ ionic build windows
```

Live Reload App During Development (beta)

The `run` or `emulate` command will deploy the app to the specified platform devices/emulators. You can also run **live reload** on the specified platform device by adding the `--livereload` option. The live reload functionality is similar to `ionic serve`, but instead of developing and debugging an app

using a standard browser, the compiled hybrid app itself is watching for any changes to its files and reloading the app when needed. This reduces the requirement to constantly rebuild the app for small changes. However, any changes to plugins will still require a full rebuild. For live reload to work, the dev machine and device must be on the same local network, and the device must support web sockets.

With live reload enabled, an app's console logs can also be printed to the terminal/command prompt by including the `--consolelogs` or `-c` option. Additionally, the development server's request logs can be printed out using `--serverlogs` or `-s` options.

Command-line flags/options for `run` and `emulate`

```
[--livereload|-l] .......  Live Reload app dev files from the device (beta)
[--consolelogs|-c] ......  Print app console logs to Ionic CLI (live reload req.)
[--serverlogs|-s] .......  Print dev server logs to Ionic CLI (live reload req.)
[--port|-p] .............  Dev server HTTP port (8100 default, live reload req.)
[--livereload-port|-i] ..  Live Reload port (35729 default, live reload req.)
[--debug|--release]
```

While the server is running for live reload, you can use the following commands within the CLI:

```
restart or r to restart the client app from the root
goto or g and a url to have the app navigate to the given url
consolelogs or c to enable/disable console log output
serverlogs or s to enable/disable server log output
quit or q to shutdown the server and exit
```

---

3. Emulating your app

Deploys the Ionic app on specified platform emulator. This is simply an alias for `run --emulator`.

iOS:

```
$ ionic emulate ios [options]
```

Android:

```
$ ionic emulate android [options]
```

Windows:

```
$ ionic emulate windows [options]
```

During emulating app in AVD or mobiles, you can inspect that app in chrome browser. Type following command in address bar of the chrome browser.

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

---

4. Running your app

Deploys the Ionic app on specified platform devices. If a device is not found it'll then deploy to an emulator/simulator.

iOS:

```
$ ionic run ios [options]
```

Android:

```
$ ionic run android [options]
```

Windows:

```
$ ionic run windows [options]
```

4.1. Specifying your target

```
$ ionic run [ios/android/windows] --target="[target-name]"
```

You can check the target name of your device/emulator running `$ adb devices`.

Read Run Ionic App on Emulator or on your Phone online: https://riptutorial.com/ionic-framework/topic/4175/run-ionic-app-on-emulator-or-on-your-phone

Chapter 17: Start Building Apps in Ionic

Examples

Starting an Ionic App

Starting an Ionic App

```
$ ionic start myapp [template]
```

Starter templates can either come from a named template, a Github repo, a Codepen, or a local directory. A starter template is what becomes the `www` directory within the Cordova project.

Named template starters

- tabs (Default)
- sidemenu
- blank

Github Repo starters

- Any Github repo url, ex: https://github.com/driftyco/ionic-starter-tabs
- Named templates are simply aliases to Ionic starter repos

Codepen URL starters

- Any Codepen url, ex: [http://codepen.io/ionic/pen/odqCz][1]
- Ionic Codepen Demos

Local directory starters:

- Relative or absolute path to a local directory

Command-line flags/options:

```
--appname, -a ....... Human readable name for the app (Use quotes around the name)

--id, -i ............ Package name set in the <widget id> config ex: com.mycompany.myapp

--no-cordova, -w .... Do not create an app targeted for Cordova
```

Read Start Building Apps in Ionic online: https://riptutorial.com/ionic-framework/topic/3945/start-building-apps-in-ionic

Chapter 18: Testing Ionic App in a Browser

Remarks

Testing of native device features like Camera, Vibration and others, many of which are found in the documentation of Ionic Native, cannot be done in the browser. This is an inherent limitation of the fact that Cordova, the platform on which Ionic depends to be able to access native Android, iOS, and Windows Mobile APIs of a device, cannot run on the browser.

One can work around this issue by mocking the functionality of the native plugin.

Example

Here's an example on how to mock the `Camera` plugin:

Go ahead and create an optional folder in your project root folder.

```
cd src
mkdir mocks
cd mocks
touch camera-mock.ts
```

Open camera-mock.ts and copy paste the following code:

```
export class CameraMock {
    getPicture(params) {
        return new Promise((resolve, reject) => {
            resolve("BASE_64_IMAGE_DATA");
        });
    }
}
```

Next open `src/app.module.ts` and import the mock class"

```
import { CameraMock } from "../mocks/camera-mock";
```

Then add it to module providers array:

```
@NgModule({
declarations: [
    MyApp,
    HomePage
],
imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
],
bootstrap: [IonicApp],
entryComponents: [
    MyApp,
```

```
    HomePage
],
providers: [
    StatusBar,
    SplashScreen,
    CameraMock,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
]
})
export class AppModule {}
```

Now you can use it in any component after importing it.

Examples

Testing in a Browser

Use `ionic serve` to start a local development server for app dev and testing. This is useful for both desktop browser testing, and to test within a device browser which is connected to the same network. Additionally, this command starts LiveReload which is used to monitor changes in the file system. As soon as you save a file the browser is refreshed automatically. Take a look at the Sass docs if you would also like to have `ionic serve` watch the project's Sass files.

```
$ ionic serve [options]
```

For chrome browser you can inspect devices (AVD or Mobiles), type following command in address bar of chrome browser.

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

LiveReload

By default, LiveReload will watch for changes in your `www/` directory, excluding `www/lib/`. To change this, you can specify a `watchPatterns` property in the `ionic.project` file located in your project root to watch (or not watch) for specific changes.

```
{
  "name": "myApp",
  "app_id": "",
  "watchPatterns": [
    "www/js/*",
    "!www/css/**/*"
  ]
}
```

For a reference on glob pattern syntax, check out globbing patterns on the Grunt website.

Note:

```
$ ionic setup sass
```

will add a `watchPatterns` propery with the default values to your `ionic.project` file that you can then edit, in addition to the `gulpStartupTasks` property as described in the Sass documentation.

Ionic Lab

Ionic Lab http://ionicframework.com/img/blog/lab.png

Ionic Lab is a feature on top of `ionic serve` that makes it easy to test your app in a phone frame and with iOS and Android platforms side-by-side. To use it, just run

```
$ ionic serve --lab
```

Read the full release announcement for all the details!

Specifying an IP Address to use

Say you want to specify what address your browser will connect to, say for testing or external users. Specify the address with the `--address` argument.

```
$ ionic serve --address 68.54.96.105
```

Service Proxies

The `serve` command can add some proxies to the http server. These proxies are useful if you are developing in the browser and you need to make calls to an external API. With this feature you can proxy request to the external api through the ionic http server preventing the `No 'Access-Control-Allow-Origin' header is present on the requested resource` error.

In the `ionic.project` file you can add a property with an array of proxies you want to add. The proxies are object with two properties:

- `path`: string that will be matched against the beginning of the incoming request URL.
- `proxyUrl`: a string with the url of where the proxied request should go.

```
{
  "name": "appname",
  "email": "",
  "app_id": "",
  "proxies": [
    {
      "path": "/v1",
      "proxyUrl": "https://api.instagram.com/v1"
    }
  ]
}
```

Using the above configuration, you can now make requests to your local server at `http://localhost:8100/v1` to have it proxy out requests to `https://api.instagram.com/v1`

For example:

```
angular.module('starter.controllers', [])
.constant('InstagramApiUrl', '')
// .constant('InstagramApiUrl','https://api.instagram.com')
//In production, make this the real URL

.controller('FeedCtrl', function($scope, $http, InstagramApiUrl) {

  $scope.feed = null;

  $http.get(InstagramApiUrl +
'/v1/media/search?client_id=1&lat=48&lng=2.294351').then(function(data) {
    console.log('data ' , data)
    $scope.feed = data;
  })

})
```

See also this gist for more help.

Command-line flags/options

```
[--consolelogs|-c] ......  Print app console logs to Ionic CLI
[--serverlogs|-s] .......  Print dev server logs to Ionic CLI
[--port|-p] .............  Dev server HTTP port (8100 default)
[--livereload-port|-i] ..  Live Reload port (35729 default)
[--nobrowser|-b] ........  Disable launching a browser
[--nolivereload|-r] .....  Do not start live reload
[--noproxy|-x] ..........  Do not add proxies
```

Read Testing Ionic App in a Browser online: https://riptutorial.com/ionic-framework/topic/3256/testing-ionic-app-in-a-browser

Chapter 19: What's the difference between “ionic build” and “ionic prepare”?

Examples

ionic build vs ionic prepare

From the official documentation:

> If you want to get advanced, you can also open up the project file for a specific platform by opening the required XCode or Android Eclipse project in platforms/PLATFORM inside the root of your project. Then, you can build and test from inside the platform-specific IDE. Note: if you go this route, I recommend still working inside of the root www folder, and when you've made changes to this folder, run the command: `$ cordova prepare ios` which will update the iOS specific project with the code from the `www` folder. Note: this will overwrite any changes you've made to the `platforms/ios/www` and other platform-specific folders.

So, to summarize this part - if you're using XCode to test and run your code, after you change some part of the code you just have to run `ionic prepare` to update the iOS project which then again you continue to use in XCode.

`ionic build` command actually prepares the final (for example in Android it's the .apk file) file which then could be copied to your device and test by running it manually on the device (or by using the `ionic emulate` command to test it on the emulator).

Read What's the difference between “ionic build” and “ionic prepare”? online: https://riptutorial.com/ionic-framework/topic/2911/what-s-the-difference-between--ionic-build--and--ionic-prepare--

Credits

S. No	Chapters	Contributors
1	Getting started with ionic-framework	Akshay Khale, Andrea Macchieraldo, Community, Devid Farinelli, Ian Pinto, leetheguy, Lightbeard, Mazz, Newton Joshua, the_mahasagar
2	Connecting Ionic with any database	Nikola
3	Create Dialog in Ionic	A-Droid Tech
4	Deploy Ionic as a website	Nikola
5	Device camera and photo library access from Ionic application	British, sonu
6	How to use EcmaScript 6 features in Ionic?	Nikola
7	Ionic - Analyze your app with jshint and gulp-jshint as part of your build process	Akilan Arasu, Ian Pinto, Sumama Waheed, thepio
8	Ionic AngularJS extensions	thepio
9	Ionic Backend Services (ionic.io)	Tomislav Stankovic
10	Ionic CLI hooks	thepio
11	Ionic Cloud for Yeoman Ionic Projects	Newton Joshua
12	Ionic CSS components	thepio
13	Ionic infinite scroll to	SANAT

	show load items on demand (Already available data not by Http Request)	
14	Ionicons	thepio
15	Publishing your Ionic app	Gerard Cuadras
16	Run Ionic App on Emulator or on your Phone	A-Droid Tech, Gerard Cuadras, Ian Pinto, Ketan Akbari, olivier, Raymond Ativie
17	Start Building Apps in Ionic	A-Droid Tech, Ian Pinto
18	Testing Ionic App in a Browser	A-Droid Tech, Ian Pinto, Ketan Akbari, maninak
19	What's the difference between “ionic build” and “ionic prepare”?	Nikola