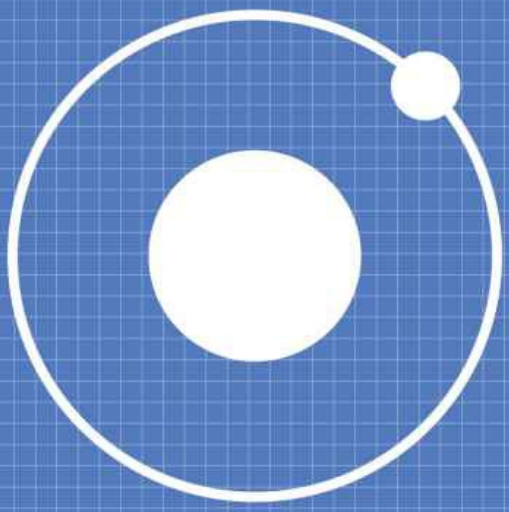
Hoc Phan

FULL-STACK MOBILE APP

with

<ionic-framework>



Full-stack Mobile App With Ionic Framework

Hoc Phan

Table of Contents

About The Author

Hoc Phan has a strong focus on frontend development and cloud computing. He started programming since the age of 12 with Pascal and Assembly on 486 computer. He has been working in various startups and large companies. In addition, he has been speaking and presenting at several local meetups, industry events and conferences. Hoc holds an MBA from the University of Washington, Michael G. Foster School of Business.

Full-stack Mobile App with Ionic Framework is his first book.

Hoc can be reached on Twitter via @innovieco.

Acknowledgments

Obviously I did not complete this book all by myself. Here is the incomplete list of people to whom I owe a huge debt of gratitude for helping make this book as good as I think it now is.

Thanks to the entire Ionic's team, particularly Katie Ginder-Vogel, for applying the proper verbiage while editing. That's why this book sounds way better than just a collection of blog posts.

Thanks to the reviewers. I apologize if I missed some of your suggestions and they were not incorporated in this final version. Particular thanks to Perry Govier for some really good catches. Well you work for Drifty so of course I have to listen to you (wink)!

Thanks to the Ionic Framework's founders: Ben Sperry (CEO) and Max Lynch (CTO). Without them, this book woundn't even make sense. Also Ben has been a huge supporter to gather the troops to help me out. Several of you retweeted about my book and it made my day.

Finally, thanks to my wife, Xena, and son, Brian, for putting up with me several nights when I got nose buried in front of the computer. None of this would have been possible without your understanding and support.

1. Introduction

Mobile app development is a growing market space. It's getting more complex than ever, due to the variation of each platform (iOS, Android, Windows Phone…). Also, in the consulting world, clients are asking for more…for less. Many might say, "We need our app to be available on all platforms, and here's our budget." That's only going to work if you use a hybrid app development framework that saves you time and money, right?

Welcome to **Full-stack Mobile App with Ionic Framework**!

This book is ideally for people who already know AngularJS and have been trying jQuery Mobile for mobile app development. You don't need to have deep knowledge of AngularJS, as it's a large topic by itself. If you have used jQuery Mobile before, Ionic Framework will give you a different perspective.

1.1 For Whom Is This Book Written?

My recommendation is to read this book actively with *a purpose*. You probably are an *indie* or running your own consulting business. You should be thinking about what kind of app you plan to build. Here are a few tips:

1. Start with your idea
2. Pick a topic
3. Spend at least a day on one example
4. Incorporate your own code

For the sake of completeness, I'll explain most concepts as if you don't have previous knowledge, but I will also leave you links to resources so you can explore further.

In each example, you should try to understand every line of code and copy the example to your own code. Make sure to play around with it by changing the configuration or how the app interacts with the data.

1.2 What If You Don't Know AngularJS?

No problem! If you are not familiar with AngularJS, here is how you can learn:

1. Read through this section and go through some AngularJS-specific online tutorials
2. Start to build the app
3. When you get stuck on AngularJS, go back to this section again to re-read
4. Isolate your questions or issues on AngularJS vs. others (Ionic or Cordova)
5. If you still cannot figure something out, post your question on Stackoverflow

1.3 What Will Not Be Covered

We will not cover **test**, as it's a big topic by itself. Plus, I just want to show you quickly how to ramp up on building things and seeing your work right away. I know it's not a good practice to write code without testing, but skipping the test topic will make things simpler to digest, as it's already

complicated.

I won’t provide instructions on the logistics between you and Apple in term of submitting apps. You can visit developer.apple.com to find out more.

1.4 Where To Find Source Code in This Book

You can access the Innovie.com Github page for all examples in this book. You can also create issues on Github against each example if you find problems or bugs.

2. Mobile App Development Today

Before we dive in to discuss different options and frameworks, let ask ourselves an obvious question: Why do we need mobile app? Because in some cases, you don't! So what should not be a mobile app at all? Here are some examples:

1. Static site that should have been responsive design website instead (i.e., restaurant website).
2. App that has fixed content that requires little updating (i.e., school curriculum).
3. App with content that be must optimized for SEO (i.e., how-to site).

There are many other criteria, but I think you get the point.

2.1 Native

I hope at this point, you're not fighting much between developing a native vs. HTML-based mobile app. In general, we use native development for:

1. High performance (i.e. heavy image processing)
2. Complex UI with lots of animation
3. Game (i.e. 3D)

Otherwise, for apps with just tabs, screens and some ways to get data via REST, welcome to the HTML5-based world!

2.2 Hybrid

There are two options for leveraging web development for a mobile app. One is to have everything (including navigation) within a Web View and plugins to access to device APIs. This is how Cordova does it, and we will talk more about that later. The second is to mix some native code with HTML. For example, the navigation menu could be native.

David Heinemeier Hansson, who created Ruby on Rails, wrote an interesting article on this topic:

> … Where we before had 100% of the meat of the app in HTML, we are now going with 90% HTML, 10% native, and really getting the best bang for our buck by picking the most worthy 10% to go native with. The integration feels really nice and seamless, and you may well not even notice at first which parts are HTML and which part is native…

A few years ago, when people tested out Titanium, the concept was similar. However, Titanium has many bad replutations on cross-platform supports and various bugs. Phonegap at the same time took off really fast due to a larger ecosystem creating free plugins.

Time also drives things to change and evolve, as Hansson mentions regarding the hardware platform itself:

> … this hybrid architecture might not work for everyone. It hardly worked for anyone in 2010 because phones were too slow, so the HTML/JS underpinnings meant worse

> performance, and users did not like that. But times have changed. Modern phones are incredibly fast, and you can even run a fair amount of JavaScript on your mobile views without hurting perceivable performance…

I think hybrid app development definitely has its place. However, there isn't a large community behind the Titanium parent company (Appcelerator), compared to Phonegap. Also, many companies decided to write their own hybrid app frameworks from scratch because they want to understand every bit of it, especially to avoid overhead performance. So, in a sense, you do need to spend hours on native development.

2.3 HTML Mobile Apps

There is a bit of history if you don't know about Phonegap vs. Cordova. The folks at Ionic wrote a very good article on this.

In a nutshell, it's the *same*. Phonegap is more of a branding on top of Cordova. Phonegap does have additional *layers of stuff* to make Cordova apps well-suited for the *build in the cloud* business model of Adobe. In this book, as does the entire Ionic Framework, we will be focusing only on Cordova.

Cordova allows native access to device APIs. What are the options of HTML/JS frameworks on top of it? There are several, actually. Most have died off over the years. One strong *legacy* framework that is still somehow lingering around is jQuery Mobile.

jQuery Mobile

Here is the problem with jQuery Mobile as of today: You barely see new tutorials on jQuery Mobile any more. They lost all their fans!

If you have a history with jQuery Mobile already, you can skip this section. But if not, here is another history lesson. jQuery Mobile has many weaknesses, but its one single strength is to allow the app developer to create apps super fast. Some common problems with jQuery Mobile are:

- It creates additional mark-up.
- Dynamic markup is a pain.
- The CSS must target the generated markup.
- Too fat & bad performance. Before it applies CSS to style your various widgets, it runs a load of Javascript, adding large numbers of classes all over the place, so that the styling comes out right. This can make for a hell of a performance hit.
- It provides many controls/functions, but only a very small part of them are used in apps, which causes much waste of performance.

I am going to exclude any discussion on Kendo Mobile or Sencha Touch because they are not really targeting indie developers but rather enterprise developers. In addition, those frameworks are suffering the same problems as jQuery Mobile.

Famo.us

Famo.us has a very interesting approach to HTML5 mobile app.

They completely rebuild the browser rendering engine, so instead of coding in typical html/css, you have to create `surface` (or `div`) and `modifier` (or `class`) in Javascript objects. The reason behind this approach is because the DOM management in the browser is very *clunky* in term of event inheritance. This causes a lot of redundancy in event processing, which results in *lagging* UI, especially when there is a long list of rendered objects.

I have coded with Famo.us for some time, and my opinion is that it's difficult and not fully baked yet (at the time I am writing this book). There are still many bugs on rendering issues, especially on scrolling (yes, they completely redid the native browser scrolling feature). In addition, it's tough to get started on Famo.us and get something done in a few hours. The reason is that you have to write the UI in Javascript completely. Famo.us is mainly targeting the Backbone.js audiences. It has a port on AngularJS, but it's still in working progress. My opinion is that it's very heavy for the AngularJS fans.

2.4 Where Does This Leave Us?

About a year ago, if you were trying to use AngularJS' out-of-the-box touch features, you should be switching to Ionic Framework now. That's why the rest of this book will be on Ionic. Why? Simply put, here are a few awesome things about Ionic:

1. It's written with AngularJS in mind.
2. UI performance is pretty good and, of course way, better than jQuery Mobile.

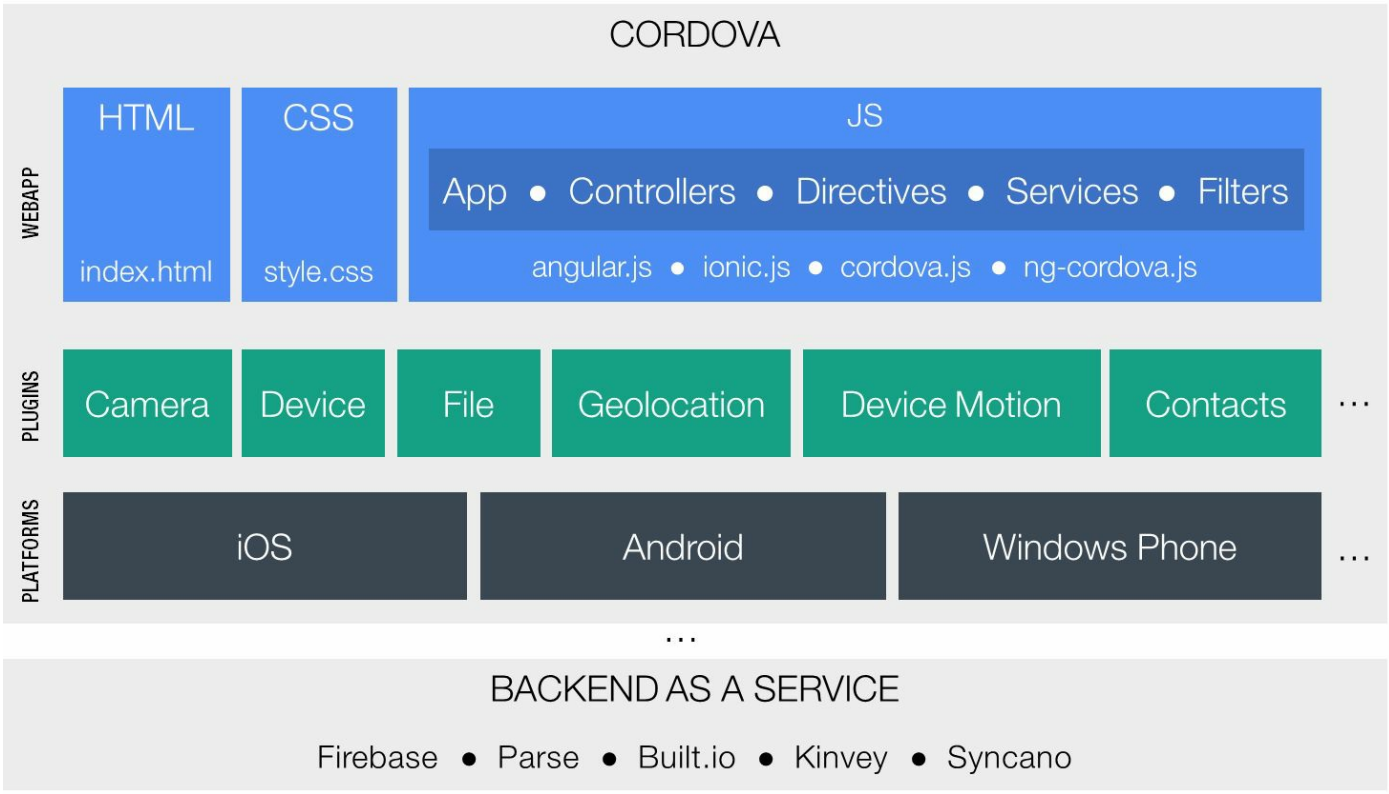
3. It offers a beautiful and comprehensive set of default styles, like a mobile focused Twitter Bootstrap
4. Sass is available.
5. Documentation is getting better.
6. There is a huge community supporting this project, and there are many apps already built using Ionic, so it's safe to make a bet on it.
7. You can literally build something within 5 minutes.

Enough said! Let's jump right into the frameworks within the Ionic stack.

3. Framework

This book will not make you an expert in any of the frameworks listed below, but it will give you a sense of how to get started and how the components integrate with each other.

Here is the high-level architecture of all the pieces:



That's a lot of stuff to learn, but you will find yourself spending most of your time in the AngularJS world of logics. The rest are just re-usable components that feed into the AngularJS environment.

3.1 Angular.js

I would say the foundation of everything we build on is AngularJS. Ionic Framework actually depends on AngularJS, unless you just want to use the CSS component of it (then you could just swap out with Bootstrap or Zurb Foundation).

If you have a history and habit of using jQuery Mobile or other frameworks, I would encourage you avoid using them at first by focusing on AngularJS-only approach. It could be challenging to get those legacy frameworks or libraries to work "nicely" with AngularJS. The following features are available in

AngularJS out-of-the-box to save you a lot of headaches:

1. **2-way Data Binding** - This makes your life a lot easier when you need to dynamically update the DOM. In the old jQuery days, you may have to keep doing `$('.someDOM').html()` each time the variable changes. In AngularJS' world, it's just a change in `$scope.someVar = "new string"` and the DOM will update itself. It works the other way around too.
2. **Scope** - You don't need to worry about trashing the global scope with variables. Once you assign a controller to a DOM like `<div ng-controller='myCtrl'>`, everything within that `div` (i.e. its children) will belong to the `myCtrl` $scope. You can update variables and communicate within the scope.
3. **Template** - A key feature of AngularJS that wins over my heart vs. other frameworks is the template engine. It came with AngularJS by default. It's completely trivial to update the DOM with a list element by using `<li ng-repeat="item in someArray">` to parse a list of `item` in `someArray` array. Some people argue that AngularJS leaves too much *power* to the HTML. I would say those things are tedious to leave within .js code itself. We just need to focus on the data model to make sure it reflects the right thing in the DOM.
4. **Ajax Call with $http** - In jQuery, you would use `$.ajax()`. While that works, it doesn't allow caching or testing to happen in a natural way. $http is a service component that deal with all ajax-related calls.
5. **Events** - Most of the times you don't need to worry about

updating views. The 2-way binding is sort of taking care for that. However, if you need to pass custom event, you can use either `$broadcast` to dispatch the event downwards to all child scopes or `$emit` to dispatch the event upwards through the scope hierarchy. You could also use `$watch()` to watch changes in a specific model. I mainly just use `$broadcast` to keep things simple. There is more to explain here.

6. **Routing** - AngularJS has its default router but we will ignore it and use the more advanced version which is angular-ui-router. What's cool about it is that you can have parent and child routes. You can assign controllers per route and template to render. Each route can attach to a *state* which has its own state object. You can also resolve a function before the controller is initialized. There is tons of flexibility.

It's good for you to grab a few concepts of AngularJS now. If you haven't used it before, that's fine. You should start Googling around by now or more when we move into each app examples / tutorials. Learning by doing is the best.

3.2 Sass

If you have been doing frontend development for sometime, you can skip this part because it's not that complicated.

The reason we mention Sass here is because Ionic Framework comes with a library of `.scss` files for you to customize.

You can take a look at the `_variables.scss` to see what's out

there for you to define your own *theme*. Their advice is not to modify any default files directly but created a new `app.scss` file and import everything in there with your overrides instead:

```
$light:                      #fff;
$stable:                     #f8f8f8;
$positive:                   #4ea4be;

@import "ionic/ionic";
```

3.3 Ionic Framework

Adam Bradley wrote a blog post entitled Where does the Ionic Framework fit in?, so I won't repeat the "stack" conversation. We are sort of using our own stack here, with the addition of Firebase for backend.

However, there are few key things you should be aware of. In a nutshell, Ionic Framework is jQuery Mobile reborn, but with a collection of CSS styles and AngularJS Directives and Services that allows you to rapidly build cross-platform mobile applications. It has built-in structure and scalability in the thought process. If you wrote a few HTML5 apps in the past with jQuery Mobile, you probably know that it doesn't design with MVC framework in mind. In Ionic, we have both the design (CSS) and the MVC mindset, leveraging AngularJS. That also means no more "spaghetti" code.

I recommend first getting familiar with the Ionic CLI.

It doesn't take much to learn Ionic because you can always go back to look up either the CSS components or Javascript AngularJS. Their docs are probably the best in class.

3.4 Cordova

Apache Cordova is the bridge between native application vs. HTML/JS. We need it to build the app for App Store / Google Play submission. Plus, it has a huge plugin library, with API access to the device's native functionalities such as GPS, camera and accelerometer.

If you don't keep track of Cordova's history, its old name was Phonegap. After being acquired by Adobe, the original platform's name changed to Cordova. Phonegap is merely a mobile cloud services company today to help you build app in different platforms without the need to setup the environment. In my opinion, you could safely ignore "Phonegap" in general for the purpose of learning.

You may want to poke around different plugins to understand what is out there. Luckily, you don't need to deal with those plugins directly. You can use the ngCordova service on top of Cordova and AngularJS. Keep in mind that even if you use ngCordova, you still need the cordova plugin because ngCordova just â€œAngular-izesâ€ the way you interact with it.

Let's say you need camera capability. You could just include `$cordovaCamera` service and call `$cordovaCamera.getPicture()` in the AngularJS' way. `ngCordova` not only made the syntax simpler, it helps to reuse AngularJS default resources such as `$q` for promise handling (things that you probably get confused).

3.5 Firebase

We need some backend capability, so that's why I picked Firebase. What it does is to provide some simple authentication feature and persistent object storage. Firebase is a newer company compared to Parse, but they are coming up to speed.

The reason I didn't build around Parse is that they seem to be too Facebook biased. After being acquired by Facebook, there is no more Twitter authentication, which you can do with Firebase, plus a bunch of others such as G+ or Github. One advantage of Parse is its vertical full-stack integration for mobile. That means you have access to ????

You will rarely call Firebase functions directly but it's good to know its capabilities.

We will use the abstraction layer on top of Firebase and AngularJS called AngularFire.

You can then use `$firebase` as a service within the controller to associate with the backend database. This is also the main advantage over Parse. Firebase is very well integrated with AngularJS. Everything happens in real-time with the 3-way data binding (view, frontend model, backend).

The example below gives you an idea of how easy it is to manipulate data in `$scope.people`, which has real backend connection to Firebase:

```
1 function MyController($scope, $firebase) {
2   var peopleRef = new Firebase("https://<my-firebase>.firebaseio.com\
```

```
3 /people");
4    $scope.people = $firebase(peopleRef);
5    $scope.addPerson = function() {
6       // AngularFire $add method
7       $scope.people.$add($scope.newPerson);
8       //or add a new person manually
9       peopleRef.update({name: 'Alex', age: 35});
10   }
11 }
```

4. Setup

I am going to assume your machine does not have anything installed yet by default.

Let's start with the basics, by installing node.js.

With node.js, now we can use `npm` to install others. We will need `cordova`, `ios-sim` (iOS Simulator) and `ionic`:

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

The `-g` parameter is to install the package globally (not just in current directory).

According to the current Ionic guide, you have three template options out-of-the-box: blank, tab, and menu.

Setup the app with a blank template from Ionic Framework:

```
$ ionic start myApp blank
```

```
$ cd myApp/
```

Add either Android and/or iOS platform:

```
$ ionic platform add ios
```

```
$ ionic platform add android
```

Note that you need to do the `platform add` before building the app or installing more plugins other than the default plugins.

This is to avoid errors later on because Cordova is confused about which platform it is building for and which plugin is for what.

Once you have your code in `www` folder (which is what we will focus on in the rest of this book) and plugins installed, do a build and run:

```
$ ionic build ios

$ ionic emulate ios
```

You could run the app using Xcode (in Mac), too. Just go to `platforms/ios/` or `platforms/android` folder and look for file `HelloCordova.*` to start.

4.1 Development Environment

I use Sublime Text for everything these days. You could code directly in Xcode or Eclipse, but those are somewhat "heavy duty" for web apps. You can download a non-commercial version on Sublime Text's website. It may pop up a dialog to ask you to buy a commercial license once in a while.

Depending on which version you downloaded, you will need Package Control, which is similar to a "Plugin Manager".

There are tons of packages that you may want to use such as Sass, Haml, Tag, ColorPicker, etc… You can browse around the Popular page for more ideas.

In order to "run" the web app, you need to turn your folder into

an http server. Sublime Text has a package to do live watch, too, but it doesn't work well. It's slow to detect changes and sometime freezes your IDE. My recommendation is to use the `ionic serve` command line from ionic-cli. It basically creates an HTTP server so you can open your app in a desktop browser. You can also use a static page generator such as Jekyll or Middleman App. Jekyll is more mature with a larger ecosystem. It's very fast and intuitive. However, I found out it doesn't work with Haml very well. Middleman App is an easier option to get both Sass and Haml support out-of-the-box. If you don't need that otherwise, Jekyll is a "safe" option.

With Jekyll, once you are in the right folder in the terminal, just do this to start watching changes in the entire folder:

```
$ jekyll serve --watch
```

You can leave it running and go back to Sublime Text to continue coding. Whenever you're curious about your code (because you want to see the results of your changes), just go to the browser and point to this url:

```
http://localhost:4000/index.html
```

For debugging, please go to the last chapter of this book.

5. Project Folder Structure

This project is based on Cordova, so most of what you see will be either iOS or Android related.

```
- platforms/
- plugins/
- scss/
  - ionic.app.scss  (your app's custom Sass file)
- www/
  - css/
    - style.css  (processed CSS file that will automatically be gene\
rated)
  - img/
  - js/
  - lib/
    - ionic/  (CSS, fonts, JS and SCSS from Ionic)
  - templates/
  - index.html
- config.xml
```

If you start a project using either of the commands below, you will get another `/www/templates` folder:

```
$ ionic start myApp tabs

$ ionic start myApp sidemenu
```

We will discuss templating in detail, but it's basically AngularJS templates using out-of-the-box functionalities.

As an app creator, you will spend most of your time in the `www` folder. As mentioned in the previous chapter, you can use `Jekyll` to watch this folder and run `http://localhost:4000/index.html` in the browser to see

your work being updated.

6. Must-know AngularJS Topics for Ionic

AngularJS is a large subject by itself. I assume you are an active learner and already looking at various AngularJS docs or blogs. While it's good to learn everything at once, it's easy to forget if we don't use or practice it. That's why I am writing this section so that you can learn those items for what you need to know to build the apps. Then you can circle back and learn the ins and outs of the entire framework later.

6.1 Basic Structure

Our app will have a single highest level module. By default from the Ionic template, it's called `starter`. You will see something like this in app.js:

```
1 angular.module('starter', ['ionic', 'ngCordova', 'starter.controller\
2 s', 'starter.services', 'starter.directives', 'starter.filters'])
```

This basically declares `starter` to be included in `ng-app="starter"` of `index.html`. We would always have `ionic` and `ngCordova` (as in other examples from this book, although `ngCordova` is not essential). The other modules are required and listed in `[...]` as well. They can be defined in separate files, such as **controllers.js:**

```
1 angular.module('starter.controllers', [])
2 .controller('MyCtrl', function($scope));
```

Here are some one-line explanations of each module: -

`controller` - Manage variables / models in the scope and trigger others, such as services or states - `directive` - Where you manipulate the DOM since directive is binded to a DOM object - `service` - Abstraction to manage models or collections of complex logic beside get/set required - `filter` - Mainly used to process an expression in the template and return some data (i.e. rounding number, add currency). Example:

```
1 {{ expression | filter:argument1:argument2:... }}
```

You can find out more information about modules here: Developer Guide - Modules

6.2 Scope

`$scope` is a context object for controller. A controller bounds to a DOM element. Controllers can be nested in the DOM, so scope will also have parents.

`$rootScope` is the highest level scope. Child scopes can inherit from all parents, including `$rootScope`. However, two unrelated or parallel controllers cannot communicate or access the other `$scope`. If you want to pass information between controllers, you need to either use events (via `$watch` or `$broadcast`) or reference to a `service`.

Directives do have their own scope and also have access to the same controller's `$scope` if it shares the same DOM or child DOM of that controller.

6.3 Top Directives to Know

If you want to *do well* in AngularJS, beside the `ng-app`, `ng-controller` or `ng-click`, you need to know more to master AngularJS:

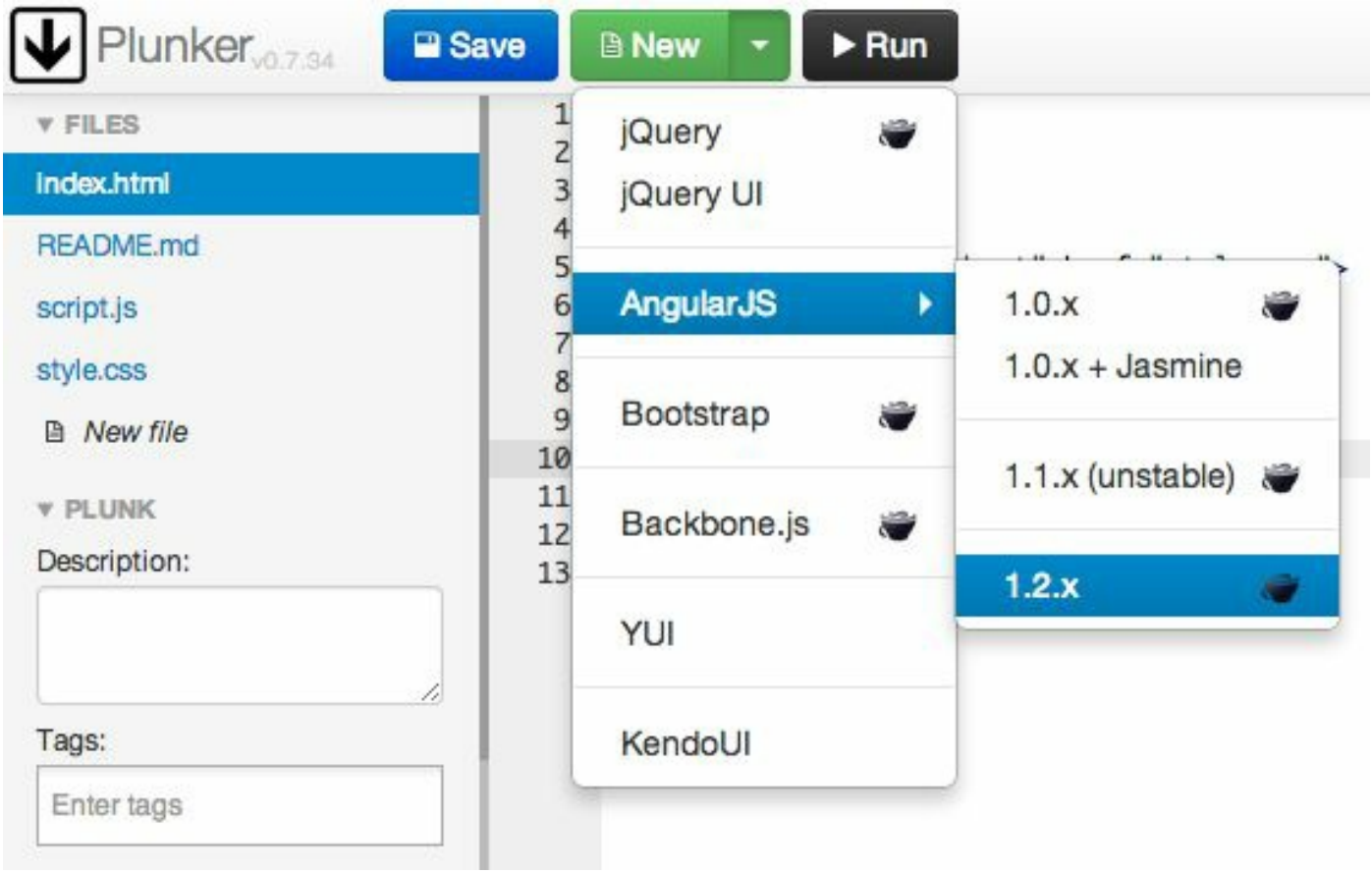
- `ng-show` and `ng-hide` - Will show or hide a DOM based on the expression being true or false. It basically just adds `display:none` to hide the element.
- `ng-if` - Similar to the use case above (hide/show), but it actually removes or adds the DOM object based on the expression value. You use this when you don't want to trash the DOM with a lot of objects that rarely appear.
- `ng-class` - You should avoid parsung a string of variables like this `class="{{ listOfClasses }}"`. It could be buggy during initialization. What you can do is to create a JSON-like expression such as `ng-class="{class1: myExpression1, class2: myExpression2}"` where `class1` and `class2` are class names. It will be *enabled* when either `myExpression1` or `myExpression2` is true.
- `ng-style` - With similar logic as `ng-class`, you should pass `{...}` to evaluate. For example: `ng-style="{ 'width' : width, 'background' : bgColor }`
- `ng-src` - You may not want to parse variables inside `src` so you have to use `ng-src`.
- `ng-href` - Like `ng-src`, it is a replacement of `href` so that Angular can control all parsing during initialization.
- `ng-model` - Assign a variable as a model for an element such as `input` field.

You don't need to use all the directives for mouse event (mouseenter, mouseleave…) because we're building mobile

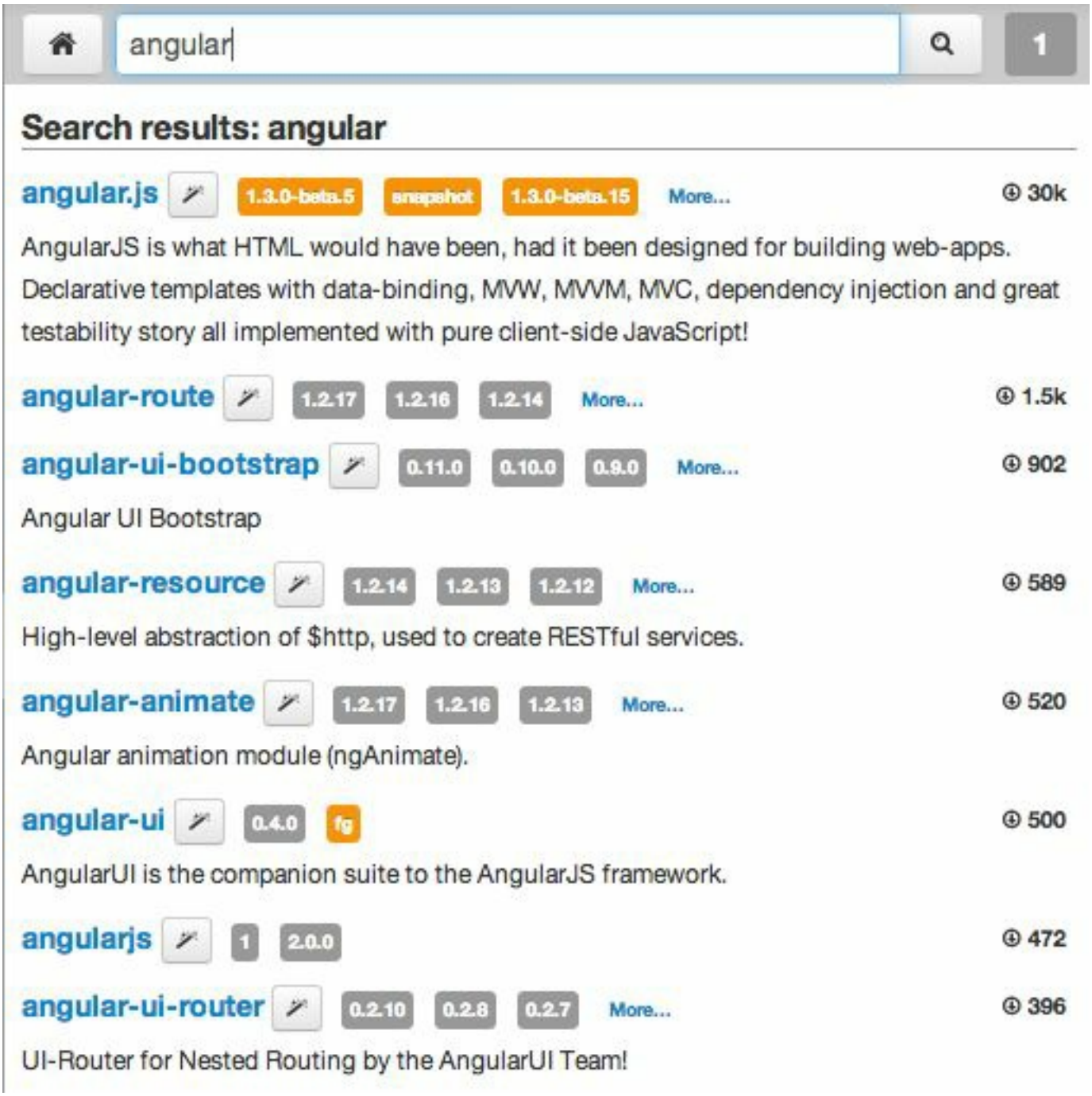
apps.

6.4 Sandbox

Sometime you just want to experiment with a few lines of code on some feature. I would recommend using plnkr.co as the sandbox. It includes an Angular 1.2.x template:



Also, you can add in a list of modules available from the site:



In terms of convenience, this is the best option, compared to jsfiddle.net or codepen.io. If you're a big fan of jsfiddle.net, you will see that it has a lot of minor problems with AngularJS.

6.5 Resources

Angular API - This is a must-view page.

"******ebook converter DEMO - www.ebook-converter.com*******"

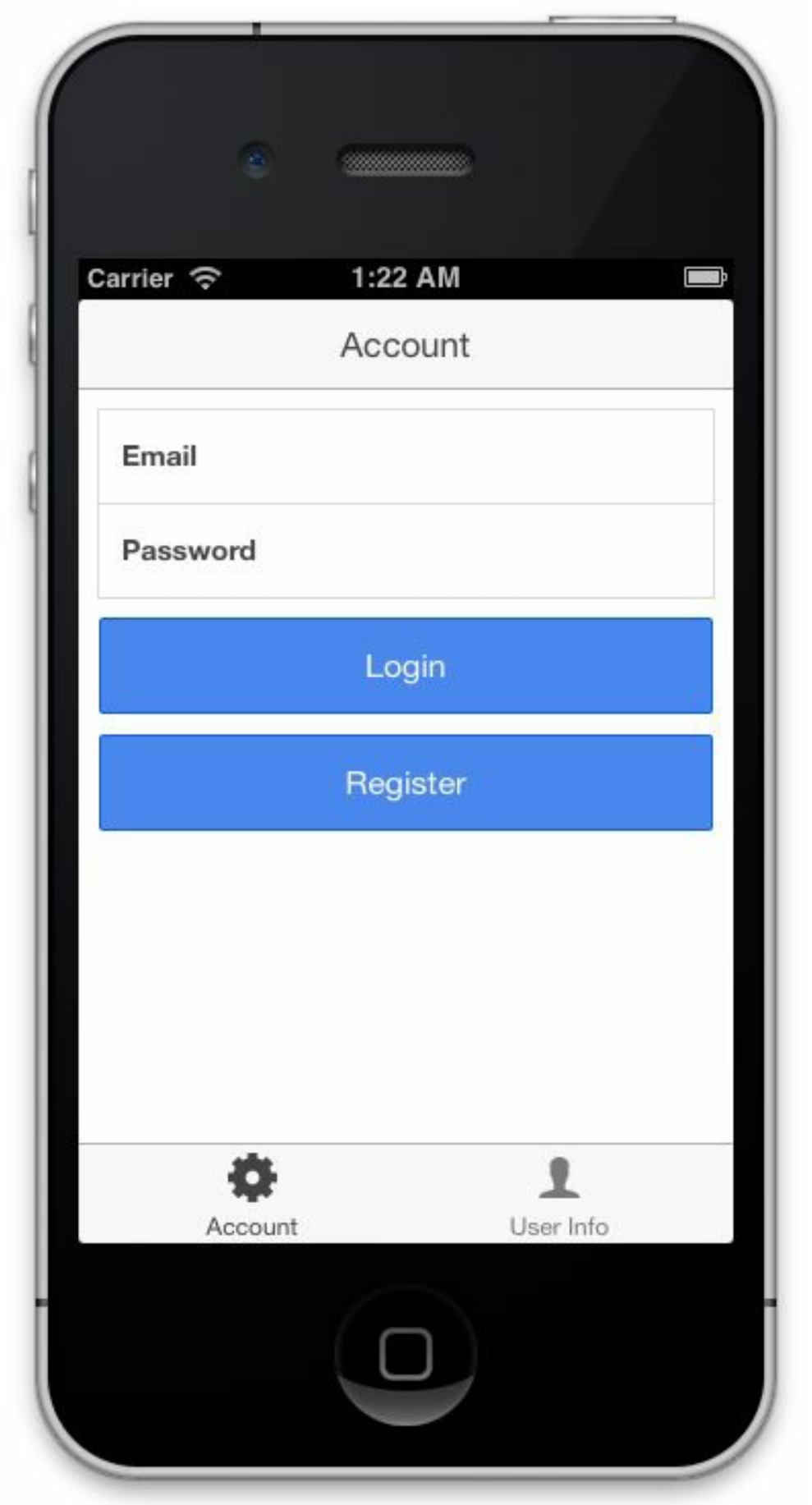
PhoneCat Tutorial App - This is a little heavy for a beginner tutorial, but that's what is available from https://angularjs.org/

An introduction to designing CSS transitions using AngularJS - Very basic tutorial on AngularJS Directives

Beginner to Expert on $scope

7. Authentication App

We will build a small app to demonstrate the ability to login, logout, register and get user information. This is how it will look at the end:



Basically, the first tab (“Account”) will allow authentication activity. The second tab (“User Info”) will show the details in user objects.

If this is your first time learning Angular.js, this example tutorial will be the hardest, so my recommendation is to play around with Angular first to understand the concept before moving further.

Because we will use some kind of backend to support authentication, Firebase is the main focus here.

7.1 What You Will Learn

- Modify a bare bones / blank Ionic project to fit your needs
- Get authentication with Firebase to work via angularFire
- Utilize all the Angular.js internal logics (app, controllers, services) to connect everything
- How Ionic makes things look beautiful

7.2 Getting Started

Let’s create a new project with a default template from Ionic Framework:

```
$ ionic start authentication-app

$ cd authentication-app/
```

For this tutorial, we don’t need some of the files, so just go ahead and delete those from the `/templates` folder:

- friend-detail.html
- tab-dash.html
- tab-friends.html

We will modify these 4 files later:

- index.html
- tabs.html
- app.js
- controllers.js
- services.js

7.3 Firebase Setup

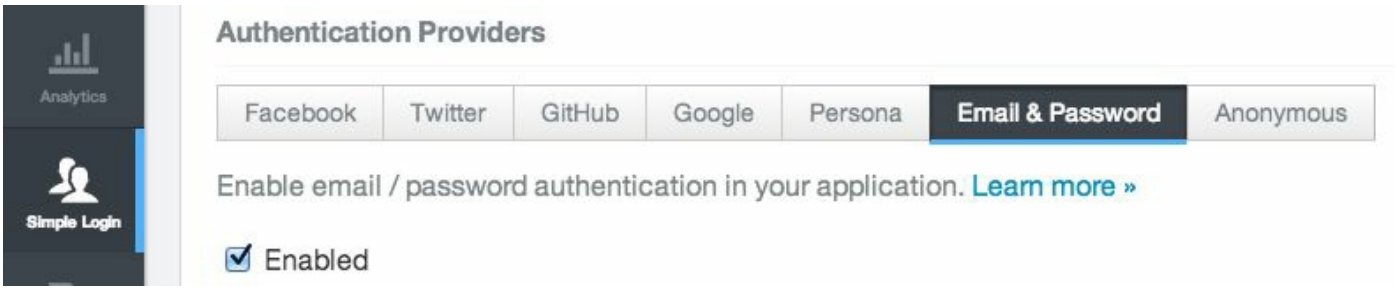
If you don't have an account yet, you should head over to https://www.firebase.com and sign up for one.

After that, create a new app by providing:

1. Name
2. Url

Once you are in the dashboard, take notice of the url, which looks something like this https://your-app-name.firebaseio.com/. This is basically your url string to use later for the `services.js` setup.

Now, click on `Simple Login` and then `Email & Password` to check on `Enable`. We will need this for simple authentication using email and password.



7.4 Define Skeleton Templates

`index.html` - This is first entry place for the app:

```
1 <body ng-app="starter" animation="slide-left-right-ios7">
2   <ion-nav-bar class="bar-stable nav-title-slide-ios7">
3     <ion-nav-back-button class="button-icon icon ion-ios7-arrow-back\
4 ">
5       Back
6     </ion-nav-back-button>
7   </ion-nav-bar>
8   <ion-nav-view></ion-nav-view>
9 </body>
```

`ion-nav-bar` is the top navigation bar.

`ion-nav-view` is the *placeholder* for other templates. Basically, Ionic will replace this with another .html file.

`tabs.html` - This template defines two bottom tabs:

```
 1 <ion-tabs class="tabs-icon-top">
 2   <ion-tab title="Account" icon="icon ion-gear-b" href="#/tab/accoun\
 3 t">
 4     <ion-nav-view name="tab-account"></ion-nav-view>
 5   </ion-tab>
 6   <ion-tab title="User Info" icon="icon ion-ios7-person" href="#/tab\
 7 /user">
 8     <ion-nav-view name="tab-user"></ion-nav-view>
 9   </ion-tab>
10 </ion-tabs>
```

At this point, you probably want to ask what that `ion-` tag thing is. Think of `ion-tabs` as `ul` and `ion-tab` as `li`. Those are Angular Directives defined by Ionic Framework. The `#/tab/account` and `#/tab/user` are different routes that we will map out in `app.js` later.

7.5 Define Router in App.js

After having the main templates, we need to change `app.js` to fix the routing:

```
$stateProvider
   .state('tab', {
     url: "/tab",
     abstract: true,
     templateUrl: "templates/tabs.html"
   })
   .state('tab.account', {
     url: '/account',
     views: {
       'tab-account': {
         templateUrl: 'templates/tab-account.html',
         controller: 'AccountCtrl'
       }
     }
   })
   .state('tab.user', {
     url: '/user',
     views: {
       'tab-user': {
         templateUrl: 'templates/tab-user.html',
         controller: 'UserCtrl'
       }
     }
   });
$urlRouterProvider.otherwise('/tab/account');
```

`$stateProvider` is an `angular-ui-router` module that makes it easier to define the mapping between route, controller,

template file,…

The `/tab` is just an abstract (the mother of other child routes). So: - `/tab/account` will load `templates/tab-account.html` to replace `<ion-nav-view></ion-nav-view>` in `index.html`. - Similarily, `/tab/user` will load `templates/tab-user.html` instead.

Note that `AccountCtrl` controller is *assigned* to `/tab/account`. So whenever that route is loaded, Angular will initialize `AccountCtrl` in `controllers.js`.

7.6 Create Login and Register Form

`tab-account.html` template will do three things:

- Show login form and button
- Show register button
- Show logout button if the user is authenticated

It has two *states*:

1. Authenticated already
2. Not authenticated yet

```
1  <ion-view title="Account">
2    <ion-content class="has-header padding" ng-show="user.id == undefi\
3  ned">
4      <div class="list">
5        <label class="item item-input">
6          <span class="input-label">Email</span>
7          <input type="email" ng-model="user.email">
8        </label>
9        <label class="item item-input">
10         <span class="input-label">Password</span>
```

```
11            <input type="password" ng-model="user.password">
12          </label>
13          <button class="button button-block button-positive" ng-click="\
14 login()">
15            Login
16          </button>
17          <button class="button button-block button-positive" ng-click="\
18 register()">
19            Register
20          </button>
21        </div>
22    </ion-content>
23    <ion-content class="has-header padding" ng-show="user.id != undefi\
24 ned">
25        <div class="list">
26          <label class="item item-input">
27            <span class="input-label">Login as</span>
28            <span class="">{{ user.email }}</span>
29          </label>
30          <button class="button button-block button-positive" ng-click="\
31 logout()">
32            Logout
33          </button>
34        </div>
35    </ion-content>
36 </ion-view>
```

The `ng-show="user.id == undefined"` means there must be a variable `user.id` exists in order to show that whole directive (or `tag` in `html` world). Let hold the thought about `user` object for a second because we will deal with it in `controllers.js`.

Any tag with `ng-click` means if the user clicks on it, it will call whatever function inside. This is similar to the old `onclick` event. So `ng-click="login()"` will call `login` function of the `AccountCtrl` (remember the controller being assigned to this template in `app.js`?).

So in this template, it will show the login form and Login /

Register buttons by default (because `user.id` doesn't exist in beginning). Once the user clicks on the Login button, it should call `login()` function and create `user.id` variable. Then the Logout button will appear. The user can then click it to `logout()`. Very simple!

Now, let's move on to the login in the controller `AccountCtrl`.

`controllers.js` should have `AccountCtrl` structured like this:

```
.controller('AccountCtrl', ["$scope", "$ionicPopup", "User", functio\
n($scope, $ionicPopup, User) {
   $scope.user = User.getUser();

   $scope.login = function () {
      User.login($scope.user.email, $scope.user.password, function(res\
) {
         if (res.id) {
            $scope.user = res;
         } else {
            $ionicPopup.alert({
               title: 'Login error!',
               template: res.message
            });
         }
      });
   };

   $scope.register = function () {
      User.register($scope.user.email, $scope.user.password, function(\
res) {
         if (res.id) {
            $scope.user = res;
         } else {
            $ionicPopup.alert({
               title: 'Register error!',
               template: res.message
            });
         }
      });
   };
```

```
32
33    $scope.logout = function () {
34      User.logout();
35      $scope.user = {};
36    };
37 }]);
```

It may look more complicated than it should because you don't know what `User` is yet. We will define this in `services.js`. But in general, we call:

1. `User.getUser()` to retrieve the user object for `$scope.user`. That means we can use it in the `ng-show` evaluation. This happens in real-time and automatically updates within Angular and its template. So when this `$scope.user` changes, the UI will change, too.
2. `User.login` will call the login function in services to access Firebase account.
3. `User.register` will register a new account in Firebase.

One more thing you need to know is that after each call to Firebase, there will be error handling using `$ionicPopup.alert()`. This is Ionic's definition of `alert()`, like in Javascript.

If you have used Angular for some time, you may ask why we need to use dot notation like `user.email` and `user.password`, rather than `email` or `password`. The answer is Ionic Framework does not do 2-way binding with pure variables. It must be objects. Check out Ionic forum thread for more information.

7.7 Create Services

Now, let's deal with the core of handling Firebase authentication.

`services.js` has `User` factory:

```
1  angular.module('starter.services', [])
2
3  .factory('User', ["$timeout", "$firebaseSimpleLogin", function($time\
4  out, $firebaseSimpleLogin) {
5    var ref = new Firebase('https://amber-fire-7765.firebaseio.com/');
6    var auth = $firebaseSimpleLogin(ref);
7    var user = {};
8
9    return {
10     login: function(email, password, callback) {
11       auth.$login('password', {
12         email: email,
13         password: password,
14         rememberMe: false
15       }).then(function(res) {
16         user = res;
17         if (callback) {
18           $timeout(function() {
19             callback(res);
20           });
21         }
22       }, function(err) {
23         callback(err);
24       });
25     },
26     register: function(email, password, callback) {
27       auth.$createUser(email, password).then(function(res) {
28         user = res;
29         if (callback) {
30           callback(res);
31         }
32       }, function(err) {
33         callback(err);
34       });
35     },
36     getUser: function() {
37       return user;
38     },
39     logout: function() {
```

```
40          auth.$logout();
41          user = {};
42        }
43    }
44
45 }]);
```

There are four functions: `login()`, `register()`, `getUser()` and `logout()`. The names are self-explanatory. Since we use `angularFire` which is the layer on top of Firebase and Angular, it already provides us with all the classes neccessary to do authentication. We just need to define this:

```
1 var ref = new Firebase('https://your-app-name.firebaseio.com/');
2 var auth = $firebaseSimpleLogin(ref);
```

The `url` is what you created earlier when setting up Firebase. It tells the `auth` object to go ahead and contact that app to do authentication against.

Another thing to take notice is the `user` object in this `User`. It's basically a private variable which stores user data. We just have to save the data there for later access (like in `getUser()` function).

7.8 Get Data for "User Info" Tab

There is one last thing we need to do in the "User Info" tab.

`tab-user.html` will define the content in this tab:

```
1 <ion-view title="User">
2   <ion-content has-header="true" padding="true" ng-show="user.id != \
3 undefined">
4       <div class="list card">
```

```
 5         <div class="item item-divider">
 6           Email
 7         </div>
 8         <div class="item item-text-wrap">
 9           {{ user.email }}
10         </div>
11       </div>
12       <div class="list card">
13         <div class="item item-divider">
14           firebaseAuthToken
15         </div>
16         <div class="item item-text-wrap">
17           {{ user.firebaseAuthToken }}
18         </div>
19       </div>
20       <div class="list card">
21         <div class="item item-divider">
22           id
23         </div>
24         <div class="item item-text-wrap">
25           {{ user.id }}
26         </div>
27       </div>
28       <div class="list card">
29         <div class="item item-divider">
30           md5_hash
31         </div>
32         <div class="item item-text-wrap">
33           {{ user.md5_hash }}
34         </div>
35       </div>
36     </ion-content>
37     <ion-content has-header="true" padding="true" class="vcenter" ng-s\
38  how="user.id == undefined">
39       <div class="row row-center">
40         <div class="col col-center">
41           Look like there is no user.id variable so you need to
login
42         </div>
43       </div>
44       <div class="row">
45         <div class="col">
46           <a class="button button-block button-positive"
ng-href="#/ta\
47  b/account">
48             Go to Login
```

```
49            </a>
50          </div>
51        </div>
52      </ion-content>
53  </ion-view>
```

This long template only has a few things in it. Similar to the "Account" tab, we also use `ng-show` on the two `ion-content` directives to toggle the login/logout state.

The first `ion-content` will show a bunch of data such as `email`, `firebaseAuthToken`, … All of these are available in the `user` object after being successfully authenticated.

The second `ion-content` only has a simple login button to redirect to `#/tab/account`. Since the user has not logged in, there is nothing to show.

One last cosmetic piece is the use of `class="vcenter"` on `ion-content`. Let's go back to `index.html` to define this "vertical center". The idea is for content inside it to be in the middle of the phone's screen.

In `index.html`, add this:

```
1  <style>
2    .scroll-content.vcenter {
3      display: -webkit-box;
4      display: -moz-box;
5      display: -ms-flexbox;
6      display: -webkit-flex;
7      display: flex;
8      -webkit-box-direction: normal;
9      -moz-box-direction: normal;
10     -webkit-box-orient: horizontal;
11     -moz-box-orient: horizontal;
```

```
12      -webkit-flex-direction: row;
13      -ms-flex-direction: row;
14      flex-direction: row;
15      -webkit-flex-wrap: nowrap;
16      -ms-flex-wrap: nowrap;
17      flex-wrap: nowrap;
18      -webkit-box-pack: center;
19      -moz-box-pack: center;
20      -webkit-justify-content: center;
21      -ms-flex-pack: center;
22      justify-content: center;
23      -webkit-align-content: stretch;
24      -ms-flex-line-pack: stretch;
25      align-content: stretch;
26      -webkit-box-align: center;
27      -moz-box-align: center;
28      -webkit-align-items: center;
29      -ms-flex-align: center;
30      align-items: center;
31    }
32  </style>
```

Since Ionic uses *flexbox* for positioning, this is a bit more than simply using `absolute` or `relative` positioning in typical css cases.

7.9 Other Options

This tutorial helps to explain simple concept of dealing with *showing* and *hiding* information based on authentication states. There is another way to do this by using route security.

The angularFire-seed project already created some nice modules to handle authentication and route mapping, instead of using the “cheap and easy” method of `ng-show` to check for `user.id`. This method is more scalable than having a bunch of `ng-show` all over the place.

8. Instagram Clone

For this app tutorial, we will make an app to take a picture and show a list of filtered images similar to what Instagram does. There will be no backend to process the image, so everything will be in Javascript, including the filtering effect process.

Here is what we plan to build:

8.1 What You Will Learn

- Access the Cordova Camera plugin to trigger camera capture and get image back in `base64` format
- Parse the `base64` data on some `canvas`
- Process the `canvas` to produce filtered image effects
- Put the images in Ionic slide box `ion-slide-box`, so you can swipe left and right to see the filtered images

8.2 Getting Started

We will start with the default template from Ionic Framework:

```
$ ionic start instaclone

$ cd instaclone/
```

Let's delete everything from the `/templates` folder, as we will use only Ionic Slide Box to manage different views from swipe left and right.

We will also create a new file called `directives.js` to contain our custom directives. More information on that later.

Since this app will need image filter features, we will add those two files into the `/lib` folder.

These files are modified from the `Filterous`:

```
1 /lib/filterous.js
2 /lib/filterousEffects.js
```

https://github.com/girliemac/Filterous

Note that if you use the one from `girliemac`, it will not work, as I have modified it to remove some of the “hard code” features (such as dealing with DOMs).

For dealing with the camera feature, we will need to talk to iOS APIs through Cordova because Cordova already has the `Camera` plugin. However, the folks at Ionic made it even better by abstracting the Cordova framework to fit in AngularJS services. Therefore, we will mostly deal with just ng-cordova. There is one file we will need to include:

```
1 /lib/ng-cordova.js
```

The `cordova.js` is already included in the template `index.html` by default. Note that you have to put `ng-cordova.js` before `cordova.js` as mentioned on ngcordova.com. Otherwise, you will see strange things happening.

The last step is to make sure the `Camera` plugin is added. The default template only has three plugins in `/plugins` folder: Console, Device and Ionic.keyboard. We want to add `Camera` as well:

```
$ cordova plugin add org.apache.cordova.camera
```

You should be able to see a new folder `org.apache.cordova.camera` being added under `/plugins` folder.

That's pretty much all the prepping we need. Basically we removed all templates, added image filter modules and included `ng-cordova` to make life easier.

8.3 Slide Box HTML

Since the UI on this app is very simple, let take a look at how we would structure the template.

We plan to use only `ion-slide-box` directly to handle all the navigation:

http://ionicframework.com/docs/api/directive/ionSlideBox/

This is basically the "skeleton" of the app:

```
<ion-nav-view>
  <ion-view>
    <ion-slide-box>
      <ion-slide>
        // Slide 1
        // Handle camera button
      </ion-slide>
      <ion-slide>
        // Slide 2
        // Render the default image after taking the photo
      </ion-slide>
      <ion-slide>
        // Slide 3
        // Render "Bright" effect
      </ion-slide>
      <ion-slide>
        // Slide 4
        // Render "Dark" effect
      </ion-slide>
      <ion-slide>
        // Slide 5
        // Render "Black & White" effect
      </ion-slide>
```

```
24       </ion-slide-box>
25     </ion-view>
26 </ion-nav-view>
```

There are 5 “screens”. Each screen is contained within the `ion-slide` directive. The first screen will give you the button to take a photo. The other screens will show the original photo, as well as one effect per screen. Here is the final code of the template:

index.html:

```
 1 <ion-nav-view>
 2   <ion-view ng-controller="CameraCtrl">
 3     <ion-slide-box>
 4       <ion-slide>
 5         <div>
 6           <h1>TAKE A PICTURE</h1>
 7           <i ng-click="getPhoto()" class="icon ion-camera button-
cam\
 8 era"></i>
 9         </div>
10       </ion-slide>
11       <ion-slide>
12         <div class="row row-center" ng-show="status.filtering > 0">
13           <div class="col col-center">
14             <button class="button button-light button-round">
15               <i class="icon ion-looping"></i>
16               APPLYING...
17             </button>
18           </div>
19         </div>
20         <div class="row row-center" ng-show="status.filtering == 0">
21           <div class="col col-center">
22             <button ng-click="convertPhoto()" class="button button-
p\
23 ositive button-round">APPLY FILTERS</button>
24           </div>
25         </div>
26         <div class="list card">
27           <div class="item item-image">
28             <source-img model="lastPhoto"></source-img>
```

```
29              </div>
30            </div>
31          </ion-slide>
32          <ion-slide>
33            <h1>BRIGHT</h1>
34            <div class="list card">
35              <div class="item item-text-wrap" ng-show="!lastPhoto ||
!s\
36 tatus.isFiltered">
37                <h2>
38                  This screen will look better after you take a
photo an\
39 d run filters!
40                </h2>
41              </div>
42              <div class="item item-image">
43                <filter-img effect="fluorescent" model="readyPhoto">
</fi\
44 lter-img>
45              </div>
46            </div>
47          </ion-slide>
48          <ion-slide>
49            <h1>DARK</h1>
50            <div class="list card">
51              <div class="item item-text-wrap" ng-show="!lastPhoto ||
!s\
52 tatus.isFiltered">
53                <h2>
54                  This screen will look better after you take a
photo an\
55 d run filters!
56                </h2>
57              </div>
58              <div class="item item-image">
59                <filter-img effect="phykos" model="readyPhoto">
</filter-\
60 img>
61              </div>
62            </div>
63          </ion-slide>
64          <ion-slide>
65            <h1>BLACK & WHITE</h1>
66            <div class="list card">
67              <div class="item item-text-wrap" ng-show="!lastPhoto ||
!s\
```

```
68 tatus.isFiltered">
69                 <h2>
70                   This screen will look better after you take a
photo an\
71 d run filters!
72                 </h2>
73               </div>
74               <div class="item item-image">
75                 <filter-img effect="sumie" model="readyPhoto"></filter-
i\
76 mg>
77               </div>
78             </div>
79           </ion-slide>
80       </ion-slide-box>
81     </ion-view>
82 </ion-nav-view>
```

While this may look like a lot of stuff, it's actually very simple. Let's start from the top down.

We have the controller called `CameraCtrl`, so everything within its child elements will be handled by this controller. That means if the user clicks a button, it will trigger `getPhoto()` function to take the photo. After that, the user can click another button to trigger `convertPhoto()` function to call a bunch of image processing functions.

You will also see various `ng-show` following with condition variable(s). We will discuss these later, as it's just a way for us to show or hide certain elements based on the states (e.g. whether the user takes a photo or not).

8.4 Camera

AngularJS has a small feature to protect image `src` from being hijacked with some URI from a local filesystem, rather than "******ebook converter DEMO - www.ebook-converter.com*******"

http in the same domain. By default, after taking a photo, iOS will give you a local filesystem URI such as `file://`, instead of relatively where the app is, such as `/img/`. Without the code below, you might not see the image in the screen after taking a photo:

app.js:

```
1 .config(function($compileProvider){
2   $compileProvider.imgSrcSanitizationWhitelist(/^\s*(https?|ftp|mail\
3 to|file|tel):/);
4 })
```

The white list is to avoid AngularJS from blocking anything starting with `file://` in `src` attribute. We have to tell $compileProvider that "Hey, it's OK to show images if they are in the local filesystem."

8.5 Controller

We will define `angular.module('instaclone.controllers', [])` in the controller with three things:

- `status` object to tell the template whether the photo has been filtered or not (`isFiltered`) and whether the app is "busy" running the filtering process (`filtering` as number of processes). This is just so we can "show" the proper element with either a text message or spinning circle.
- `$scope.getPhoto` function will be triggered when the user hits the camera button.

- `$scope.convertPhoto` function will be triggered when the user hits the “APPLY FILTERS” button.

Here is the full content of the controller:

controllers.js:

```
angular.module('instaclone.controllers', [])

.controller('CameraCtrl', ["$scope" , "$cordovaCamera", "$ionicSlide\
BoxDelegate", function($scope, $cordovaCamera, $ionicSlideBoxDelegat\
e) {
   $scope.status = {
      isFiltered: false,
      filtering: 0
   };

   $scope.getPhoto = function() {
      $ionicSlideBoxDelegate.next();

      $cordovaCamera.getPicture({
         quality: 75,
         targetWidth: 480,
         targetHeight: 640,
         correctOrientation: true,
         saveToPhotoAlbum: false
      }).then(function(imageURI) {
         $scope.lastPhoto = imageURI;
      }, function(err) {
         alert('Unable to take picture');
      });
   };

   $scope.convertPhoto = function() {
      $scope.readyPhoto = $scope.lastPhoto;
   };

}]);
```

One thing you may notice is the use of `$ionicSlideBoxDelegate` in the `getPhoto()` function. We

need to include it so that after taking the photo, we tell the Ionic Slide Box to move to **slide 2** because it's where we render the original photo with the "APPLY FILTERS" button on top.

We just trigger it by using `$ionicSlideBoxDelegate.next()`. More information about this delegate:

http://ionicframework.com/docs/api/service/$ionicSlideBoxDele

`$cordovaCamera` is something already defined in `ng-cordova.js` for us to use. We just need to call `getPicture()` with parameters and callbacks when success or error. Note that we have to reduce the image size and quality; otherwise, the app will crash (due to large memory requirement) or the image filters will take forever (like 15 minutes):

```
1 quality: 75,
2 targetWidth: 480,
3 targetHeight: 480,
```

`quality` is the quality of the saved image, expressed as a range of 0-100, where 100 is typically full resolution with no loss from file compression.

If you want to photo to be automatically saved into the device's album, you can do `saveToPhotoAlbum: true` instead.

After the photo is successfully taken, it will be returned as an URI in `imageURI` variable. First, we will assign it to `lastPhoto` in the `scope`, so that it will be rendered as an original image. Make a note of this line in **index.html**:

```
<source-img model="lastPhoto"></source-img>
```

We will discuss how `source-img` directive is being structured to handle `lastPhoto` variable as a model.

Then, whenver the user hits the `convertPhoto()` button, it will assign the URI to `readyPhoto`, so that the filter processing can be triggered to run filters. Again, those are handled in directives.

8.6 Image Effect Directives

Make sure you have this in **directives.js**:

```
angular.module('instaclone.directives', ['instaclone.services'])
```

Remember in the **index.html**, we have two new directives: `source-img` and `filter-img`. They will be defined in this file.

The `source-img` directive will watch the model and render the image by assigning the URI into `src` attribute. Then it will call `imageData` service to give the image dimension.

```
.directive("sourceImg", ['imageData', function (imageData) {
  return {
    restrict: 'AEC',
    link: function(scope, element, attrs) {
      scope.$watch(attrs.model, function(value) {
        if ((value != undefined) && (value != '')) {
          var img = new Image();

          img.src = value;
          img.onload = function() {
            imageData.set({
              width: this.width,
              height: this.height
```

```
                });
            }

            element.empty();
            element.append(img);
        }
      });
    }
  }
}])
```

You may ask: why do we need `imageData` service at all? There are two reasons: 1. If the app tries to get the image dimension somewhere in the code by directly accessing `img` object, it may get `0` because `img.src` might be still in the process of rendering. You have to get the value of `img` only through `onload` because it's the safest route. 2. Any data you tend to use across services, directives or controllers (i.e. global), you should put as a service. In this case, it's just a get/set object.

Here is the code within this service:

services.js:

```
angular.module('instaclone.services', [])

.factory('imageData', function() {
  var width, height;

  return {
    set: function(args) {
      width = args.width || 0;
      height = args.height || 0;
    },
    get: function() {
      return {
        width: width,
        height: height
      }
```

```
16       }
17     }
18 });
```

Back to the `source-img` directive again, whenever you want to watch a variable to detect changes, you can use this:

```
1 scope.$watch(model, function(value) {});
```

The `model` must be a string or reference from within the attribute `attrs` of that directive. So we basically can do this in the html file:

```
1 <source-img whatever="something"></source-img>
```

Reference to that `whatever` attribute by using `attrs.whatever`.

Next, let's talk about the core of this app, which is the `filter-img` directive:

```
 1 .directive("filterImg", ['$timeout', 'imageData', function ($timeout\
 2 , imageData) {
 3   return {
 4     restrict: 'AEC',
 5     link: function(scope, element, attrs) {
 6       scope.$watch(attrs.model, function(value) {
 7         if ((value != undefined) && (value != '')) {
 8           var img = new Image();
 9           var originImgData = imageData.get();
10
11           img.src = value;
12           img.width = originImgData.width;
13           img.height = originImgData.height;
14
15           scope.status.filtering++;
16           $timeout(function() {
17             ApplyEffects[attrs.effect](img, 'jpeg');
18             scope.status.isFiltered = true;
19             scope.status.filtering--;
```

```
20                }, 1);
21
22                element.empty();
23                element.append(img);
24            }
25          });
26       }
27    }
28 }]);
```

This directive will watch for changes from the `readyPhoto` variable, as it was assigned like this in the html:

```
1 <filter-img effect="fluorescent" model="readyPhoto"></filter-img>
```

The only place that can change `readyPhoto` variable is within the controller's `convertPhoto()` function, if you recall. So after the user hits the convert button, all three directives (with effects called "Bright", "Dark" and "Black & White") will be called to process whatever in `readyPhoto`. First, it will assign the URI to `img.src` for rendering. Then it assigns the dimension from thr `imageData` service. Keep in mind that the `Image()` we created will not have dimension, even if we assign `src` a value.

We will call `ApplyEffects` to run through the filter effects within the `$timeout()` function. This is a trick to make the directives to pass through `ApplyEffects` and make it a separate parallel / non-blocking process. Otherwise, the UI will not see any update from changes of `scope.status.filtering++`. This `filtering` is needed to show the spinning circle in our **index.html**:

```
1 <div class="row row-center" ng-show="status.filtering > 0">
```

```
2     <div class="col col-center">
3       <button class="button button-light button-round">
4         <i class="icon ion-looping"></i>
5         APPLYING...
6       </button>
7     </div>
8 </div>
```

After every cycle of filtering, we want to make sure the element is empty before adding the new image object:

```
1 element.empty();
2 element.append(img);
```

8.7 Finalize

In order for everything to work, we need to make sure that the `app.js` file includes everything in `instaclone` module:

app.js:

```
1 angular.module('instaclone', ['ionic', 'instaclone.services', 'insta\
2 clone.directives', 'instaclone.controllers', 'ngCordova'])
```

This is a small step that people usually miss, especially including `ngCordova` in the list.

After that you should try to build the app.

```
$ ionic build ios
```

Then emulate it on iOS.

```
$ ionic emulate ios
```

8.8 More Ideas

There is an Instagram plugin for Cordova on Github. You could write some extra code to pass the image to Instagram. The user must have Instagram installed in the phone first, though. I think this idea is nice when you plan to do some “cool” image processing (i.e adding funny text) before letting Instagram filter.

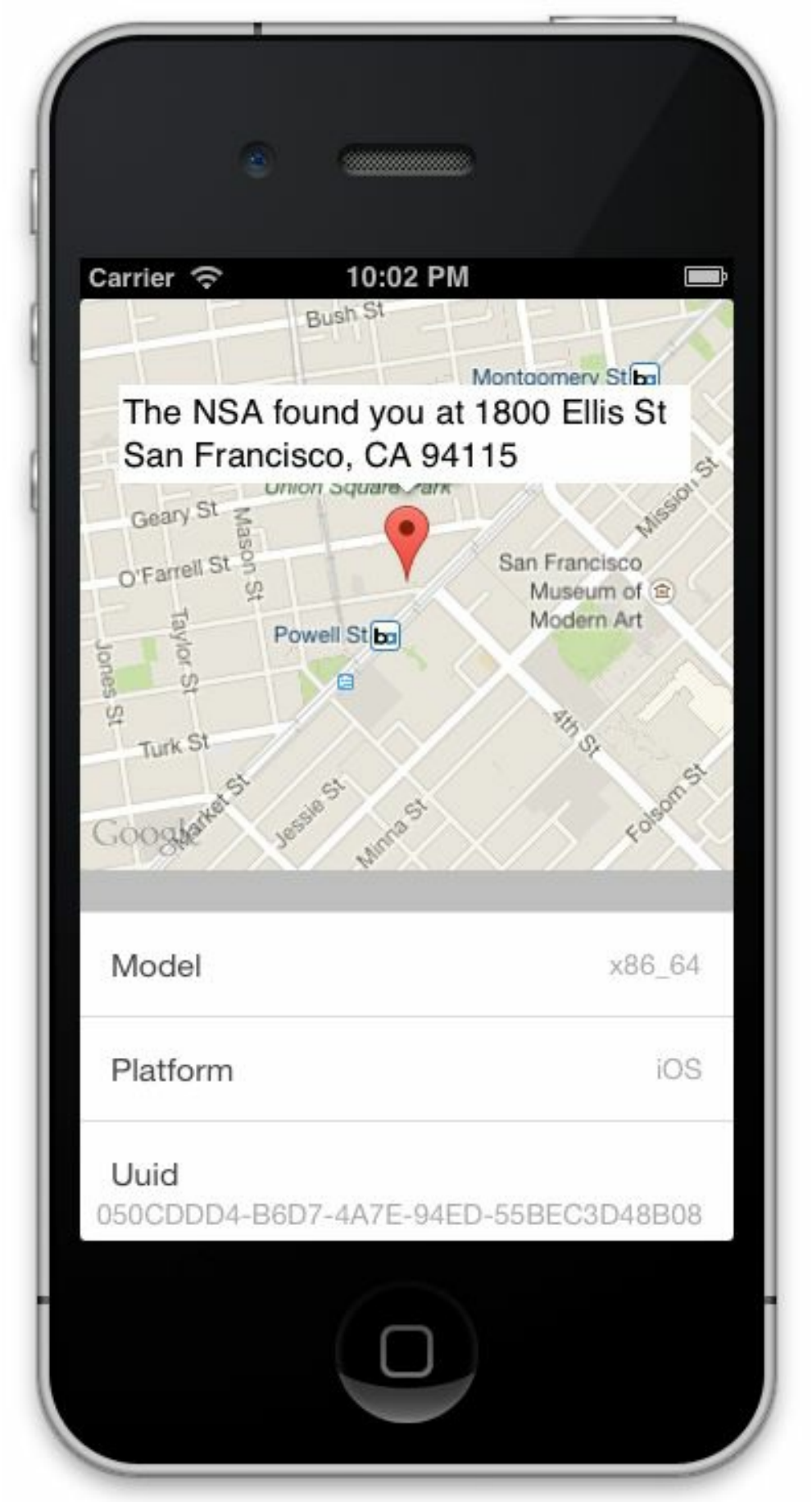
This tutorial does not go over the File plugin so that you can access existing photos or save the new photos in the iOS filesystem.

I leave you the option to test around the Social Network plugin and post the resulted images to Twitter or Facebook. I will cover a separate app tutorial on Social Networks.

9. Geolocation and Google Maps

At the time of this writing, there's a lot of concern about how much the NSA knows about our personal lives. This topic provides an opportunity to show how to use the Google Maps and Geolocation plugins. This app will show your current address and your device's details such as model, version, etc.

Here is the screenshot (some sensitive data removed):



9.1 What You Will Learn

- Use Geolocation plugin to get your longitude and latitude
- Pass the coordinate information to the Google Maps plugin to render the location in the map
- Get the current address using the coordinate and add a marker with that address text
- Pull device information and parse it into an Ionic item list

You may ask why we're not just using the HTML5 and Javascript version of geolocation and maps, instead of Cordova plugins. The answer is simple: performance. It's very obvious that if you use the SDK, map rendering and optimization tends to be faster. In addition, HTML5 geolocation sometimes has some strange bugs that require the user to accept permission twice: once for the app and once for the inside browser object.

9.2 Getting Started

Create a blank Ionic project and add the platform you are working on (`ios` in this example):

```
$ ionic start nsaeye blank

$ cd nsaeye/

$ ionic platform add ios
```

We will use the Device plugin. It already came with the default project, so we don't need to install it separately.

There are two additional plugins required for this app:
"******ebook converter DEMO - www.ebook-converter.com*******"

Geolocation and Google Maps.

Installing the Geolocation plugin is straightforward:

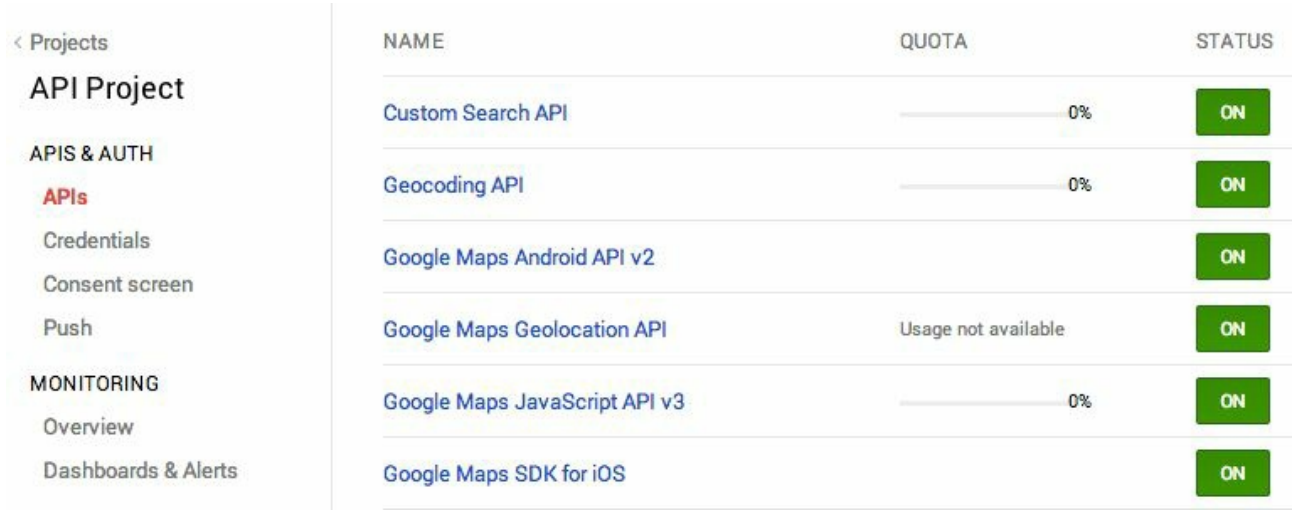
```
$ cordova plugin add org.apache.cordova.geolocation
```

Google Maps plugin is a bit more complicated because you will need the Google Maps API key for your project. The Github page provides instructions about how to set this up, but for the sake of completeness, I will go over the iOS installation:

https://github.com/wf9a5m75/phonegap-googlemaps-plugin/wiki/Tutorial-for-Mac

You don't need to go through the steps of creating the project here, because we are using Ionic, instead of just bare Cordova. Here are the steps:

1. Head over to Google APIs Console
2. You will need to log in with your Gmail account and register a new project. Please just follow the instructions from Google, because this page changes all the time.
3. The Google Maps plugin instructions say to turn on only **Google Maps SDK for iOS**, but I would say that it doesn't hurt to ensure other APIs related to maps are enabled as well. This is in my setup (probably from other old projects too):



4. Go to Credentials to create your own key
5. Click **Create new key** and select **iOS**
6. The end result should look like this:

7. Copy that key (highlighted black)
8. Go back to your terminal window and install Google Maps plugin with your key replacing `YOUR_IOS_API_KEY_IS_HERE`:

```
$ cordova plugin add plugin.google.maps --variable
API_KEY_FOR_IOS="YOUR_IOS_API_KEY_IS_HERE"
```

9.3 Create Template

Our template will be in only `index.html` file as this is a simple

project with two components: map and item list.

We will create a directive called `ionMap` which will be discussed later. But the idea is to call the Google Maps plugin to render the map inside this DOM object:

```
1 <ion-map width="100%" height="300px"></ion-map>
```

The device information (i.e. platform, version…) will be in standard Ionic list:

```
 1 <ul class="list" ng-controller="DeviceCtrl">
 2   <li class="item">
 3     Model
 4     <span class="item-note">
 5       {{ deviceInfo.model }}
 6     </span>
 7   </li>
 8   <li class="item">
 9     Platform
10     <span class="item-note">
11       {{ deviceInfo.platform }}
12     </span>
13   </li>
14   <li class="item">
15     Uuid
16     <span class="item-note">
17       {{ deviceInfo.uuid }}
18     </span>
19   </li>
20   <li class="item">
21     Version
22     <span class="item-note">
23       {{ deviceInfo.version }}
24     </span>
25   </li>
26   <li class="item">
27     Cordova
28     <span class="item-note">
29       {{ deviceInfo.cordova }}
30     </span>
```

```
31    </li>
32 </ul>
```

This list is managed by `DeviceCtrl` which will call the Device plugin to get data for the following:

- Model
- Platform
- Uuid (iOS)
- Version
- Cordova version

It will then parse the data into `deviceInfo` object.

Also make sure you have `ngCordova` included:

```
1 <script src="lib/ng-cordova.js"></script>
2 <script src="cordova.js"></script>
```

9.4 Get Device Information

We have only one controller in this app to handle the device information:

controllers.js:

```
1 angular.module('NSAEye.controllers', [])
2
3 .controller('DeviceCtrl', ['$scope', '$cordovaDevice', '$ionicPlatfo\
4 rm', function($scope, $cordovaDevice, $ionicPlatform) {
5   $ionicPlatform.ready(function() {
6     $scope.deviceInfo = $cordovaDevice.getDevice();
7     $scope.$digest();
8   });
9 }]);
```

Basically we initiate `$cordovaDevice` to talk to Device plugin and get all the information from `$cordovaDevice.getDevice()`.

Note that `getDevice()` depends on the success initialization of Cordova. That's why we need to wrap the code within `$ionicPlatform.ready` to make sure both Ionic and Cordova are finished with initialization.

You also need to call `$scope.$digest();` in order for AngularJS to pass its updated scope variables and trigger UI changes. This is a large topic by itself on AngularJS. By default, Angular triggers the digest cycle in each controller and directive. This "cycle" will do two main things:

1. Detect changes in scope variables
2. Update the changes in the UI or other bindings

In general, if you change Angular models in the scope object **within a callback**, you need to tell Angular framework to *rescan* the entire scope to detect changes. Since we do some changes of `$scope.deviceInfo` within `$ionicPlatform.ready` callback, we have to trigger this detection manually. Otherwise, you will see an annoying bug where the model is actually changed, but somehow the UI still shows the old values.

9.5 Get Coordinate via Geolocation Plugin

To get current coordinate, use `$cordovaGeolocation`

```
$cordovaGeolocation.getCurrentPosition().then(function(location) {
   // Do something here with location object
   // Then run digest cycle again
   $scope.$digest();
});
```

We get the coordinate data from within the `ionMap` directive. The callback from `getCurrentPosition()` will trigger map initialization and adding marker. Note that we still need to use `$scope.$digest()` within `getCurrentPosition()` callback as discussed previously.

9.6 Create ionMap Directive

The `ionMap` is as simple as getting coordinate and then calling `MyHouse` service to create the map and add marker:

```
angular.module('NSAEye.directives', [])

.directive("ionMap", ['MyHouse', '$cordovaGeolocation', '$ionicPlatf\
orm', function (MyHouse, $cordovaGeolocation, $ionicPlatform) {
   return {
      restrict: 'AEC',
      link: function(scope, element, attrs) {
         $ionicPlatform.ready(function() {
            var div = document.createElement('div');
            div.style.width = attrs.width;
            div.style.height = attrs.height;

            element.append(div);

            $cordovaGeolocation.getCurrentPosition().then(function(locat\
ion) {
               MyHouse.init(div, location);
               $scope.$digest();
            });
         });
      }
   }
}]);
```

The flow is basically this: - Whenever Ionic and Cordova is ready, trigger `$ionicPlatform.ready` callback - Create a div and append into the `ion-map` DOM object - Call `$cordovaGeolocation.getCurrentPosition()` to get location data - Once done, call `MyHouse.init` and pass the `div` object with `location` data to continue

9.7 Initialize Map and Add Marker

The hardest part of this app is creating the `MyHouse` service because this works directly with Google Maps plugin.

The initialization is as follows:

```
return {
  init: function(div, location) {
    map = plugin.google.maps.Map.getMap(div);
    myLatLng = new plugin.google.maps.LatLng(location.coords.latitud\
e, location.coords.longitude);
    map.addEventListener(plugin.google.maps.event.MAP_READY, setMyMa\
p);
    return;
  }
}
```

First, we have to create the `map` object and *link* the `div` object, so the Google Maps plugin knows where to render the map. Then we assign the `myLatLng` object with the current coordinate: `location.coords.latitude` and `location.coords.longitude`.

It's important to know that `plugin.google.maps.Map.getMap` does take sometime to process, and it will trigger a *ready* event once it has successfully created the map. That's why we need to

add an event listener for `plugin.google.maps.event.MAP_READY`.

Function `init` is the only thing we *expose* to the outside world for this service. Everything else is *private*. Now, let's talk about what's in `setMyMap`.

There are two private functions:

```
var addMyMarker = function(results) {
  if (results.length) {
    var result = results[0];

    map.addMarker({
      'position': myLatLng,
      'draggable': false,
      'title': 'The NSA found you at ' + result.extra.lines[0] + '
'\
 + result.extra.lines[1]
    }, function(marker) {
      marker.showInfoWindow();
    });
  } else {
    alert("Not found");
  }
}

var setMyMap = function(map) {
  map.moveCamera({
    'target': myLatLng,
    'zoom': 15
  });

  map.geocode({
    'position': myLatLng
  }, addMyMarker);
}
```

Basically, `setMyMap` does two things:

1. Moves the camera to current coordinate (`myLatLng`) by calling `map.moveCamera`
2. From the latitude and longitude data, calls `map.geocode` to get the address of that location. Once done, trigger the callback `addMyMarker`

Because Google Maps reverse geocoding returns several addresses, `addMyMarker` will take only the first result (which is most likely the correct address). We will call `map.addMarker` and fill out the JSON with parameters we want, including the address string itself, in two lines:

- `result.extra.lines[0]`
- `result.extra.lines[1]`

The data in `geocode()` has a lot more information. You could do a `console.log(results)` to see the full list of fields.

9.8 Edit app.js Bootstrap

Our final step is to make sure the bootstrap file `app.js` includes all the controllers, directives and services. Modify the file a below:

app.js:

```
1 angular.module('NSAEye', ['ionic', 'NSAEye.services', 'NSAEye.direct\
2 ives', 'NSAEye.controllers', 'ngCordova'])
3
4 .run(function($ionicPlatform) {
5   $ionicPlatform.ready(function() {
6     console.log('test ionicPlatform');
7     if(window.StatusBar) {
8       // org.apache.cordova.statusbar required
```

```
        StatusBar.styleDefault();
      }
    });
});
```

You now can build and run the app in `nsaeye` folder:

```
$ ionic build ios

$ ionic emulate ios
```

9.9 Replace Google Maps API Key

Note that Google has quota on free API usage. For example, you cannot exceed one request per second per use, and you can only have a couple thousands requests per day. This quota changes all the time, but you need to know about this.

In any case, if you have problem with your key, you have to go back to the Credentials page and **Regenerate key**. In order to change the key manually in your app, you have to edit `/plugins/ios.json`. Look for the two places below:

```
"*-Info.plist": {
   "parents": {
      "Google Maps API Key": [
         {
            "xml": "<string>YOUR_IOS_API_KEY_IS_HERE</string>",
            "count": 1
         }
      ]
   }
}
```

```
"plugin.google.maps": {
   "API_KEY_FOR_IOS": "YOUR_IOS_API_KEY_IS_HERE",
   "PACKAGE_NAME": "com.ionicframework.starter"
}
```

Just edit the `YOUR_IOS_API_KEY_IS_HERE` line and replace with your new key.

9.10 Other Cool Features of Google Maps Plugin

Marker has a long list of features, such as:

- Add a Marker
- Show InfoWindow
- Add a marker with multiple line
- Modify Icon
- Text Styling
- Base64 Encoded Icon
- Click a marker
- Click an infoWindow
- Create a marker draggable
- Drag Events
- Create a flat marker

Since you cannot pop up a div on top of native Google Maps, the `Marker` features are very handy. My suggestion is to experiment with a few scenarios, such as:

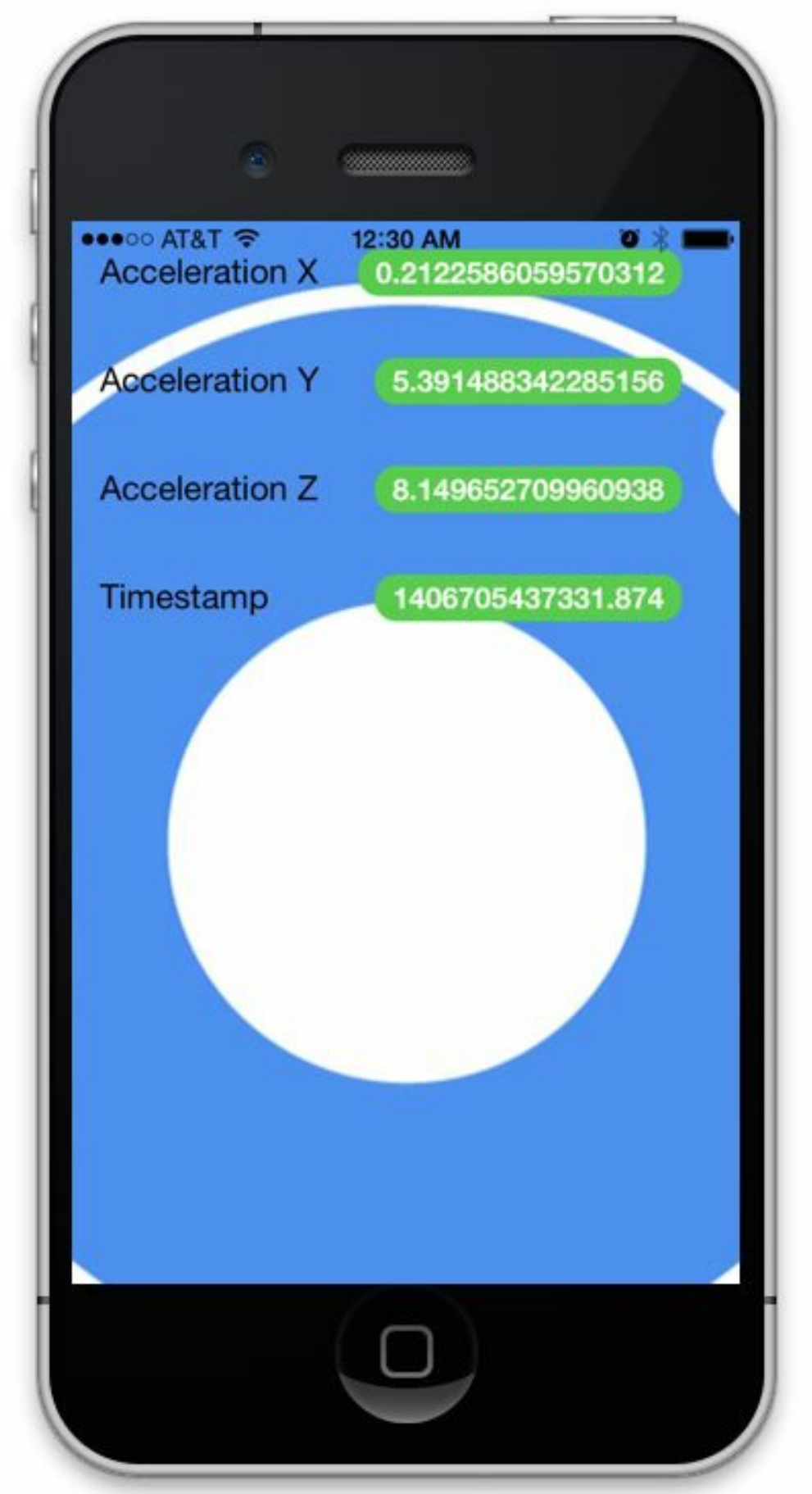
- **Touch a marker and go to a page:** You just need to listen to `plugin.google.maps.event.MARKER_CLICK` event and do whatever is needed in the callback function.
- **Show an avatar / profile image as a marker:** The `addMarker` does take base64 image string. So you can pass something like this in the argument `'title':`

```
canvas.toDataURL()
```

There are lots of way to work with Google Maps. You can visit the Github page of Google Maps plugin to find out more.

10. Accelerometer

In this chapter, we will leverage the device's accelerometer to move an image around based on the x and y axis. The app will show a big image in the background and what we got from x, y, z and current timestamp:



If you tilt the phone to the left, the image should move to the left as well. This is just a fun way to show the tilt effect by calculating image position from x and y.

10.1 What You Will Learn

- Get accelerometer data from the device
- Calculate and transform the data into image position
- Reposition background image with 2-way data binding

10.2 Getting Started

Create `imageMotion` from a blank Ionic project:

```
$ ionic start imageMotion blank

$ cd imageMotion/

$ ionic platform add ios
```

The only plugin we need is `device-motion` to get accelerometer information:

```
$ cordova plugin add org.apache.cordova.device-
motion
```

We will modify `index.html` and `app.js` (of course with some `css`). So let's dive into it.

10.3 Create Template

First make sure `ng-cordova.js` is included:

```
<script src="lib/ng-cordova.js"></script>
<script src="cordova.js"></script>
```

The structure of this template will need to have the background image and listing table with x, y, z and timestamp data. We will modify the `css` to have the background image:

```
.img-content {
  background-image: url(../img/test.png);
  background-repeat: no-repeat;
}
```

Then `ion-content` must adjust its position based on the x and y movement in real time. We will bind that data in the `$scope.pos` variable like this:

```
<ion-content ng-style="{'background-position': pos}" ng-class="{'img\
-content': pos}">
```

The reason we use `ng-class="{'img-content': pos}"` is to avoid the image showing up first before the device provides x and y position. Otherwise you may see the image in its default position and quickly *jerk* into a new position (based on the device's current angles). So by checking if `$scope.pos` exists first, we can avoid assigning `img-content` too early.

Lastly, we will use list elements from Ionic Framework. Our code for the template will be:

index.html:

```
<body ng-app="imageMotion">
  <ion-pane ng-controller="DeviceMotionCtrl">
    <ion-content ng-style="{'background-position': pos}" ng-class="{\
'img-content': pos}">
```

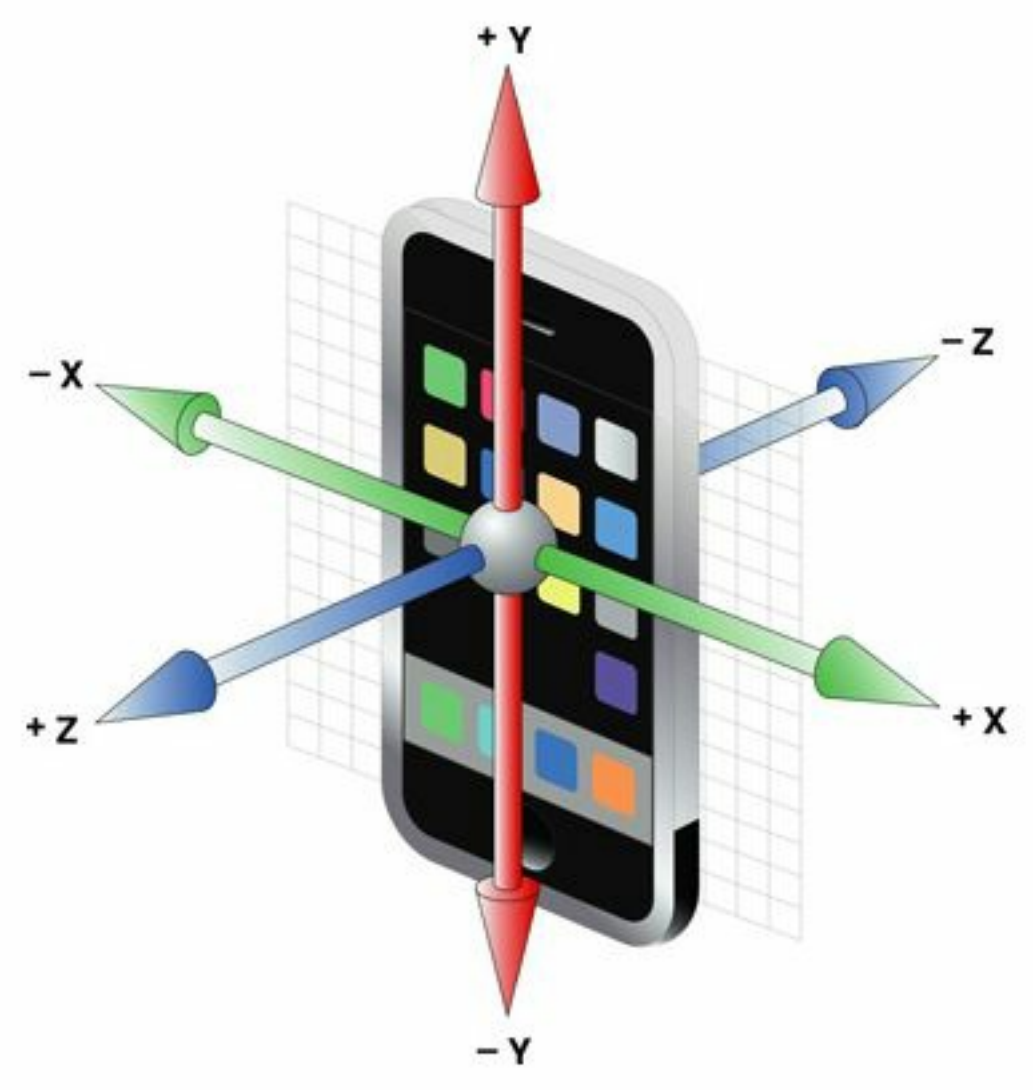
```
 5         <div class="list">
 6           <a class="item" href="#">
 7             Acceleration X
 8             <span class="badge badge-balanced">{{ acceleration.x
}}</s\
 9 pan>
10           </a>
11           <a class="item" href="#">
12             Acceleration Y
13             <span class="badge badge-balanced">{{ acceleration.y
}}</s\
14 pan>
15           </a>
16           <a class="item" href="#">
17             Acceleration Z
18             <span class="badge badge-balanced">{{ acceleration.z
}}</s\
19 pan>
20           </a>
21           <a class="item" href="#">
22             Timestamp
23             <span class="badge badge-balanced">{{
acceleration.timesta\
24 mp }}</span>
25           </a>
26         </div>
27       </ion-content>
28     </ion-pane>
29 </body>
```

We want the list background color to be transparent (so we can see through the image):

```
1 .item {
2   background-color: transparent;
3   border-color: transparent;
4 }
```

10.4 How Accelerometer Works

The picture below offers the best explanation about what kind of value will be returned from the device accelerometer:



The values of x and y will be zero when the device sits flat on the surface. If you “roll” it from one axis, the value will change. It could go from `-9` to `9`. One thing you may notice by testing this is that the numbers tend to move in a small range of `0.1`, even though you’re not touching the device. That’s why you either have to round up the number or reduce polling frequency.

10.5 Device Motion Controller

Using Device Motion is actually pretty simple:

```
1 $cordovaDeviceMotion.watchAcceleration({ frequency: 100 }).then(
2 function() {},
```

```
3 function(err) {},
4 function(acceleration) {
5   // Do something with acceleration here
6 });
```

Basically, for every **100 ms**, you get the new `acceleration` information. This is just a JSON object:

```
1 {
2   x: number,
3   y: number,
4   z: number,
5   timestamp: number
6 }
```

The more complicated part here is what to do with acceleration calculation. We only need to use x and y coordinates. This number ranges from -9 to 9, according to what I saw on the iPhone 5. You may want adjust this number later. But for now, I assign the range into two arrays, `xAccRange` and `yAccRange`, for calculation.

Also, I want the background image to move within 5% of its center. The center positioning in css is `50% 50%`, so I will make the array `[0.45, 0.55]` for x and y. This gives you the flexibility to change later.

The rest of the formula is to make sure if it is `-9`, then change position to `45%`, and so on.

Here is our controller in **app.js:**

```
1 .controller('DeviceMotionCtrl', function($scope, $ionicPlatform, $co\
2 rdovaDeviceMotion) {
3   var   = [-9, 9],
```

```
 4         yAccRange = [-9, 9],
 5         xPerRange = [0.45, 0.55],
 6         yPerRange = [0.45, 0.55],
 7         xPerInc = (xPerRange[1] - xPerRange[0]) / 100,
 8         yPerInc = (yPerRange[1] - yPerRange[0]) / 100,
 9         xAccTotal = xAccRange[1] - xAccRange[0],
10         yAccTotal = yAccRange[1] - yAccRange[0];
11
12    $scope.acceleration = {};
13
14    $ionicPlatform.ready(function(){
15       $cordovaDeviceMotion.watchAcceleration({ frequency: 100 }).then(
16       function() {},
17       function(err) {},
18       function(acceleration) {
19          $scope.acceleration = acceleration;
20          var xAcc = (-acceleration.x.toFixed(1) +
Math.abs(xAccRange[0]\
21 )),
22              yAcc = (acceleration.y.toFixed(1) +
Math.abs(yAccRange[0])\
23 );
24          var x = ((xPerInc * (xAcc / xAccTotal) * 100 + xPerRange[0])
*\
25   100).toFixed(1),
26              y = ((yPerInc * (yAcc / yAccTotal) * 100 +
yPerRange[0]) *\
27   100).toFixed(1);
28          $scope.pos = x + '% ' + y + '%';
29       });
30    });
31 });
```

10.6 Expand Usage of Accelerometer

There is a Shake Gesture Detection for the Cordova plugin. You can find out more information on the [Github account of leecrossley](#).

To install, use:

```
$ cordova plugin add
```

"******ebook converter DEMO - www.ebook-converter.com*******"

```
https://github.com/leecrossley/cordova-plugin-shake-detection.git
```

This plugin is not a part of ngCordova yet, so you have to use it directly in the directive:

```
1 shake.startWatch(onShake);
```

`onShake` is a function you want to call when there is a shake event.

11. Splashscreen

Splashscreens are very simple to create, but splashscreens with ending animation aare a little bit more tricky. We will build an Ionic splashscreen that will zoom in and then fade out just like the Twitter app:

You may think to use the splashscreen plugin from the tutorial from the Ionic folks. However, this plugin will not let you to do any kind of animation. As a matter of fact, we will not be using the splashscreen plugin at all. In my opinion, this plugin is not that useful because it only allows you to `show` or `hide` the splashscreen, which is somewhat being taken care of by Cordova already.

11.1 What You Will Learn

- The method we will use is to replace the default splashscreen with our own image
- Then we show the same image on the first page
- At the end of timeout (or a promise), we add an animation class

11.2 Getting Started

Create `ionsplash` from a `tabs` Ionic project:

```
$ ionic start ionsplash tabs

$ cd ionsplash/

$ ionic platform add ios
```

I am just using `tabs` so that we have something to play around with after the splashscreen disappears. Otherwise, we may see just a blank page, as in `blank` template.

11.3 Replacing Splash Images

First, we need to find out where Cordova stores its default splash images. If you look in the folder `platforms/ios/HelloCordova/Resources/splash/`, you can see all the default images there. For now, I am just going to replace two files from `img/` folder:

- `Default-568h@2x~iphone.png`
- `Default@2x~iphone.png`

The other files are for very old iPhone versions or landscape/iPads, on which we don't need to focus in this tutorial.

You could have more fun with customization by changing any splashscreen related variables in `platforms/ios/cordova/defaults.xml`, but I am going to skip that.

11.4 Create Splash Image in Template

We will use a controller to handle the animation and timing. So let's add `LoadingCtr` to the `body` of `index.html`:

```
1 <body ng-app="starter" animation="slide-left-right-ios7" ng-controll\
2 er="LoadingCtrl">
```

The splashscreen itself will appear after the Cordova's splashscreen. It will be the same image, so people won't notice the *switch*.

index.html:

```
<ion-content scroll="false" scrollbar-x="false" scrollbar-y="false" \
ng-if="!splashComplete">
  <div class="thumbnail">
    <div class="image" ng-class="{zoom: splashAnimate}">
      <img src="img/Default-568h@2x~iphone.png"/>
    </div>
  </div>
</ion-content>
```

Let's take some time to understand what's going on here. First, we have `$scope.splashComplete` to show or hide the element. If we hide it, the splashscreen will disappear. There is also `ng-class="{zoom: splashAnimate}"` to add `zoom` class when `$scope.splashAnimate` is true. At this point, we are basically *simulating* splashscreen appearance, zoom in animation, and then disappearing.

To complete the look and feel, we can add some classes:

```
img {
  max-width: 100%;
  display: block;
}

.thumbnail {
  position: absolute;
}

.image {
  width: 100%;
  height: 100%;
}

.image img {
  -webkit-transition: all 0.5s ease; /* Safari and Chrome */
  -moz-transition: all 0.5s ease; /* Firefox */
  -o-transition: all 0.5s ease; /* IE 9 */
  -ms-transition: all 0.5s ease; /* Opera */
  transition: all 0.5s ease;
}

```

```
.image.zoom img {
   -webkit-transform:scale(2.25); /* Safari and Chrome */
   -moz-transform:scale(2.25); /* Firefox */
   -ms-transform:scale(2.25); /* IE 9 */
   -o-transform:scale(2.25); /* Opera */
   transform:scale(2.25);
   opacity: 0;
}
```

Note that `zoom` class will make the element transform, by scaling up `2.5x` and opacity to `0`.

11.5 Manage Animation in Controller

Add two `$timeout` in **controllers.js**

```
.controller('LoadingCtrl', function($scope, $timeout) {
   $timeout(function() {
      $scope.splashAnimate = true;
   }, 4500);

   $timeout(function() {
      $scope.splashComplete = true;
   }, 5000);
})
```

In the real world, this is probably within some promise. Otherwise, it wouldn't make sense to have users to wait for 5 seconds to see the splashscreen. In our `LoadingCtr`, we basically make the animation happen in 4.5 seconds by turning on `$scope.splashAnimate = true`. Then after 5 seconds, we say the splash is completed, and the UI can go ahead and hide it by detecting `$scope.splashComplete = true`. Users then see our normal tab navigation from the Ionic template.

11.6 Known Issue

"******ebook converter DEMO - www.ebook-converter.com*******"

If your device has an on-going call or Hotspot connection, the top status bar will be extended by about `20px`. That means your splash image will be pushed down a little after the default Cordova splash image. This happens very quickly, and you may not notice if you don’t pay attention. However, this is a problem across the board in iOS. I am not sure about Android at this point. But if you take a look at other apps, such as Twitter and Yelp, they have the same issue.

12. Debug

I mostly just debug an app by using `console.log()`. However, for it to work via the Xcode console, you need to install the Cordova Console plugin:

```
$ cordova plugin add org.apache.cordova.console
```

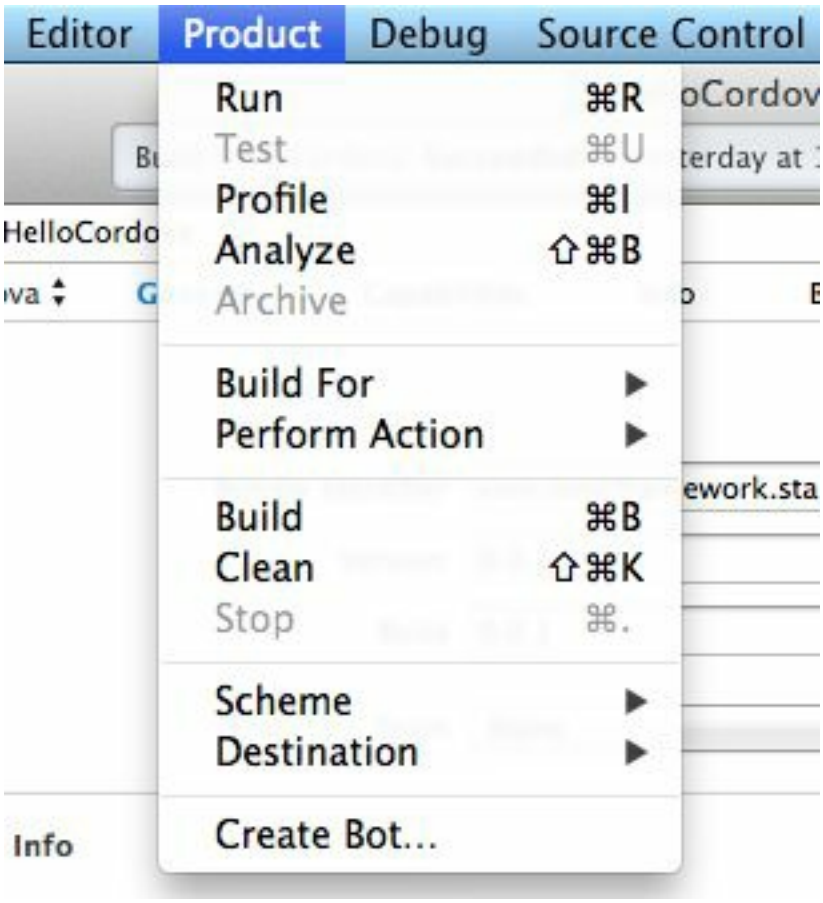
I usually *run* the app via `Jekyll` and debug it in Chrome Developer first. Once its `hmtl/js/css` is working, I emulate it in iOS or Android and debug in the console from there. I like the console method because it's simple and allow me to get values at multiple points in one run.

However, there are more *elegant* ways to debug an Ionic app. The folks at Ionic Framework wrote a very good blog post on Debugging AngularJS Apps from the Console. One trick is to use AngularJS Batarang, which is a Chrome plugin, to drill down to scopes and models.

If you're developing for iOS alone, Safari allows you to point the web inspector to your iPhone. Note that this method only works for Safari on a Mac. It won't work for Windows.

12.1 Tips and Tricks

In XCode, you may need to clean your build folder once in awhile. This is applicable to other IDE, such as Eclipse or Visual Studio, as well. Go to **Product > Clean**



If you see random errors and cannot run the app in the iOS simulator, go to the iOS Simulator menu and click **Reset Content and Settings…**

Hopefully, this will clear all the content causing issues.

Note that in iOS, you cannot do `/somepath` to reference to typical `http://localhost/somepath` because iOS is using a relative path. For example, `/img/test.png` won't work, but something like `../img/test.png` will work just fine.

13. Ionic Creator

Ionic Creator is a great interface builder similar to the Visual Basic's drag-and-drop style to accelerate your app development. The Ionic's founders started Codiqa and Jetstrap several years ago to make things easier for jQuery Mobile developers. But now you're in the new world with AngularJS, Cordova, Bootstrap… Let's welcome Ionic Creator! First you need is to sign up for a free account to get started.

Ionic Creator is a better way to bootstrap your project and visualize how it should look via a web-based interface. That means you can drag-and-drop components (i.e. buttons, images, checkboxes…) into the UI and move them around the way you like. Once finished, you can export everything as a project with all .html, .css and .js files.

13.1 When to Use Ionic Creator

Ionic Creator should be used when you want to quickly create a "skeleton" of your project. It's more user friendly than using this command line as we introduced in an earlier chapter:

```
$ ionic start myApp blank
```

Your development workflow would look like this:

1. Create a template in Ionic Creator
2. Export the working app as a project
3. Test on your phone to validate the UX

4. Start coding (UI tweaks, backend…)

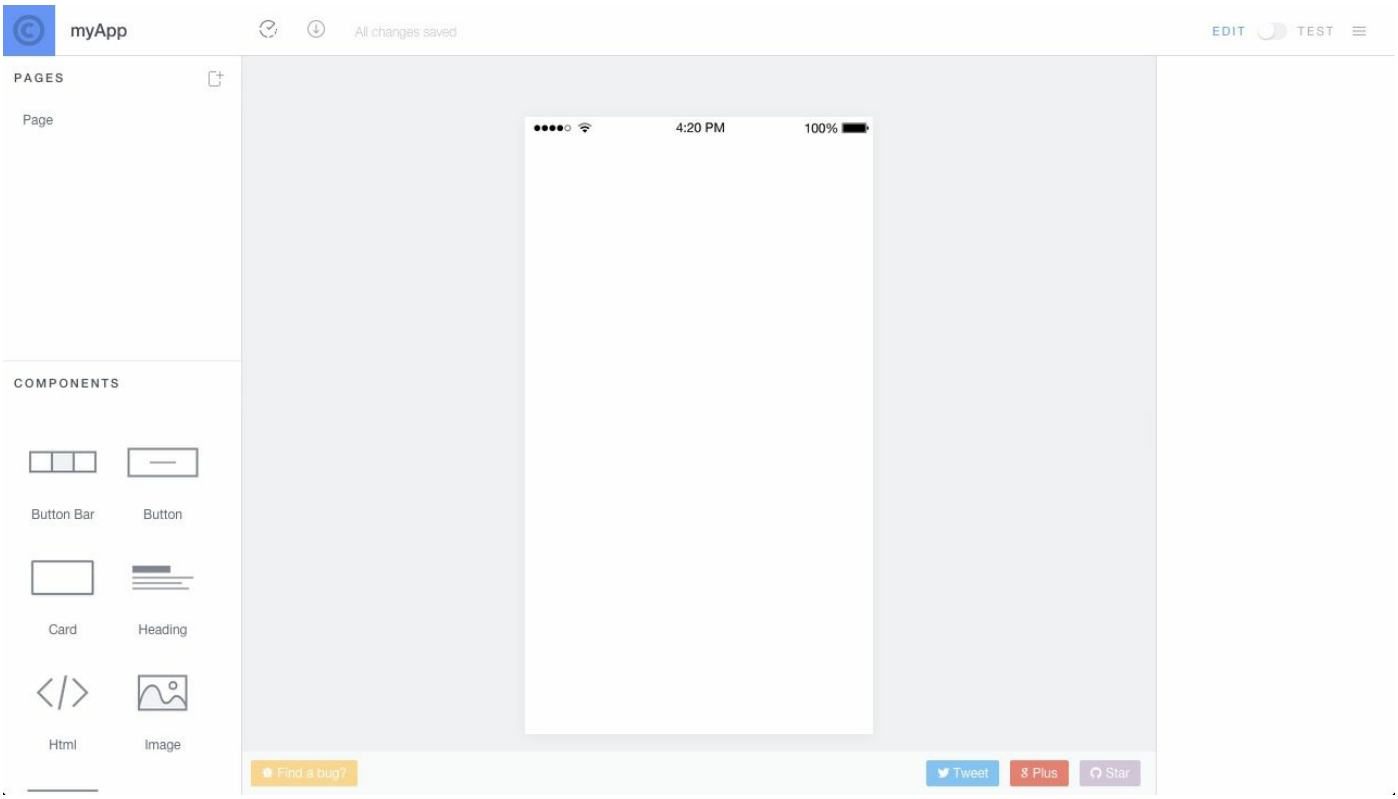
Ionic Creator doesn't do everything for you. For example, if you want to access specific Cordova plugin features, you have to write that code separately. Also if you want to tweak the interface outside of what is allowed within Creator, that will also need specific modification in the .html and .css files.

13.2 How to Get Started

This book will not go over specific app examples using Ionic Creator. However we can take a look at some high level features to understand the workflow.

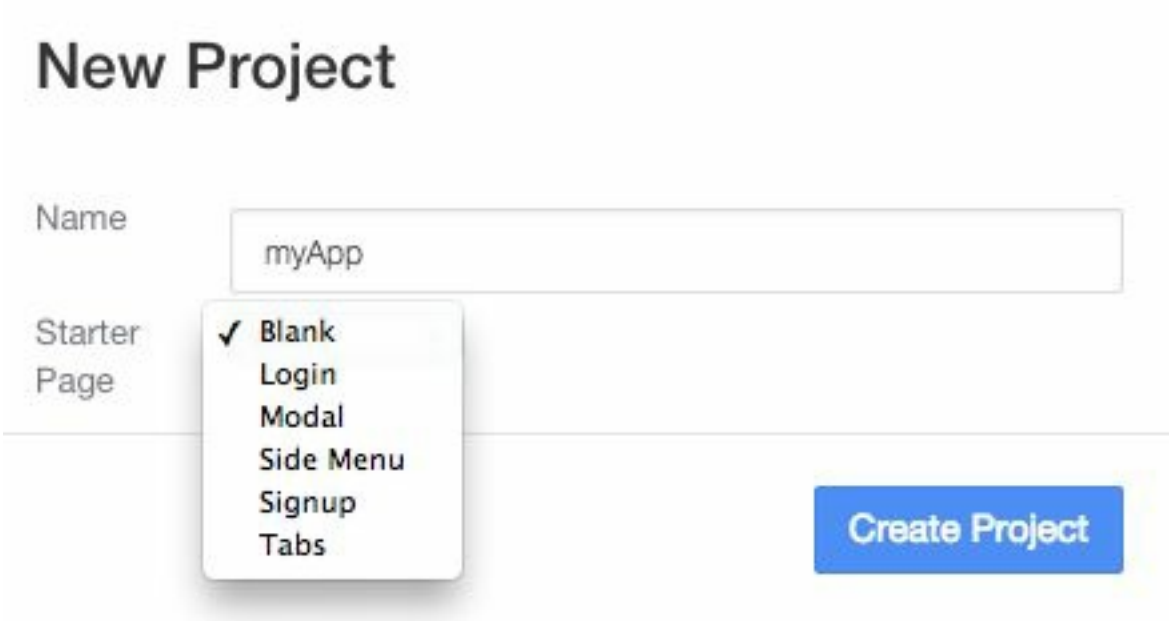
First, you will need to login https://creator.ionic.io/

Immediately you will see this simple screen:



The center area is your app interface. The left side gives you a list of “pages”. So one page is a single route. You also have access to a number of UI components which you normally have to code by hand in the html file. The right panel shows properties of any selected component.

Click on the hamburger icon to slide out the right menu and click the plus icon to add a new project:



You're free to do whatever you need to do here by dropping components to the center screen. If you need to create a new page, you have to click the plus sign in the Pages panel. Each page is represented as a "link" which is basically route in `Angular UI Router`'s definition. To navigate to another page (i.e. after clicking a button), you can just change the Link property and point to that page.

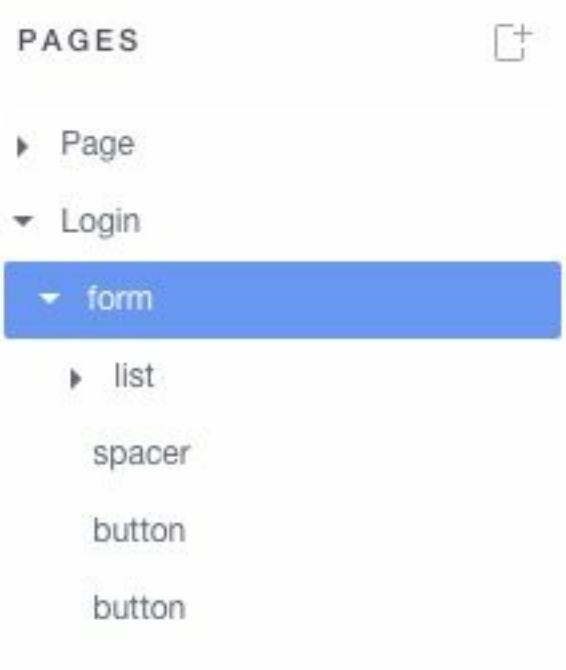
There is an Edit button on top where you can toggle back and forth between Edit Mode and Preview Mode. It's very useful to see how your app would look like and behave.

Once completed, click the Export button on the top navigation. You have three options:

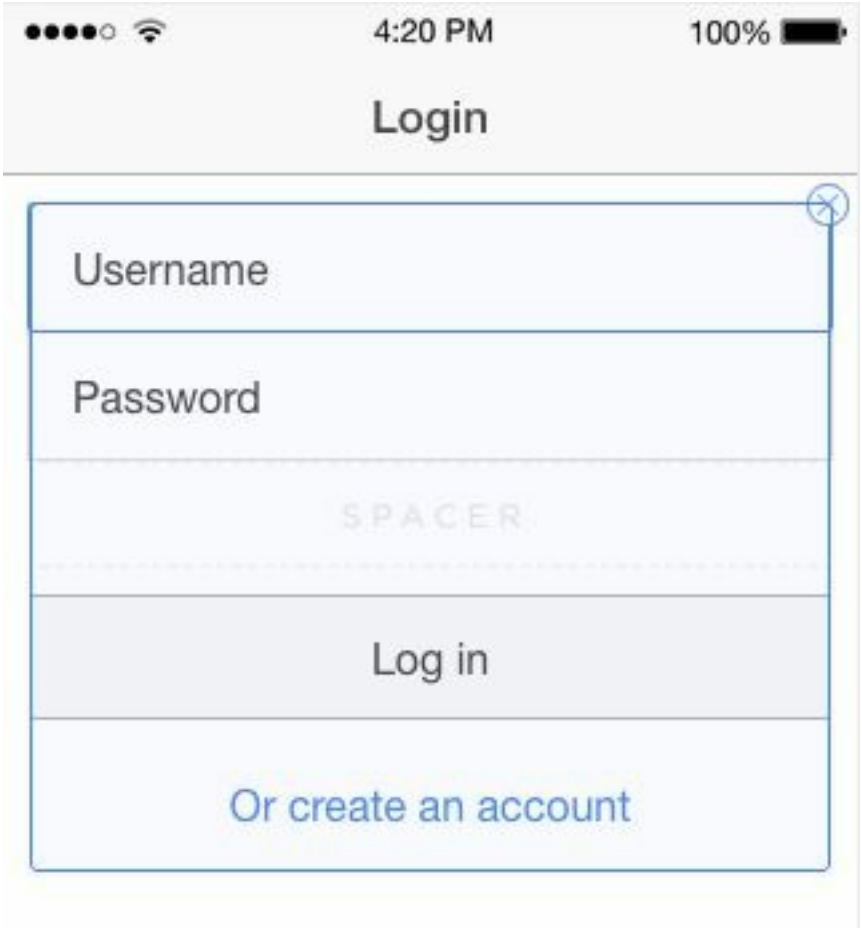
1. Use the Ionic CLI tool to get the code
2. Download the project as a zip file
3. Review the raw HTML

I like to see the raw code because sometime I just use Ionic Creator to copy a portion of the generated code into my project. Your project folder will tend to be structured differently so it is nice to just copy and paste.

The best way to learn Ionic Creator is to play with it. You can add a new page and pick out any existing template. This example shows a Login page template:



Here is how it should look out-of-the-box:



If you break something, it’s very simple to start a new project.

"******ebook converter DEMO - www.ebook-converter.com*******"

At the time of writing this book, Ionic Creator is still in the early phase so you will see more features and bug fixes in the near future. It’s a great tool to use for “prototyping” and get the initial template or project scaffolding. You should continue to code in your regular IDE for the rest of the app.

14. Multi-Platforms Compilation and Signing with PhoneGap Build

Since the acquisition, PhoneGap.com has been launching several product offerings for the PhoneGap community. One of them is PhoneGap Build.

The main reason you need to use PhoneGap Build is when you don't want to install and configure the developer environment for all mobile platforms on your computer. It's a very time consuming process. You have to find out the latest version of all the components and SDKs for Android, Windows Phone or Blackberry. On top of that they must not be conflicting each other or have versioning issues. Alternatively you could create a separate virtual machine for each but you will face performance problems. Even if you use Ionic Box via Vagrant, it's still not that straight forward (i.e. installing Vagrant, VirtualBox, command lines…). I have some success using Ionic Box with Genymotion which is a full-blown Android Emulator. It's no different from Android OS running on a virtual machine and it could be a time-saver. You can read the Genymotion Documentation online to find out more. However, if you just need to build the app for multiple platforms quickly, PhoneGap Build is the right answer.

One disadvantage of using PhoneGap Build is that you cannot really test or debug the app in real-time. It gives you the built package and that's it. You have to figure out how to install it on your local physical device.

14.1 How to Get Started

Once your Ionic app is ready, visit the Getting Started with Build page and go through the instructions. We don't get into the details here because the specific steps could change by the time you're reading this book. There are a few things to keep in mind though:

1. Make sure to delete `phonegap.js` or `cordova.js` from your project folder. PhoneGap Build will automatically include this.
2. In the `index.html` file, include the below script:

   ```
   <script src="phonegap.js"></script>
   ```

   OR

   ```
   <script src="cordova.js"></script>
   ```

The other potential issue is plugin declaration. You may need to modify the `config.xml` file to include the right plugins during the build process. Sometime PhoneGap Build can detect them automatically. Most of the time, you have to add them manually as below:

```
<gap:plugin name="com.phonegap.plugins.example"
version="2.2.1" />
```

Here is an example of how an `config.xml` file may look like:

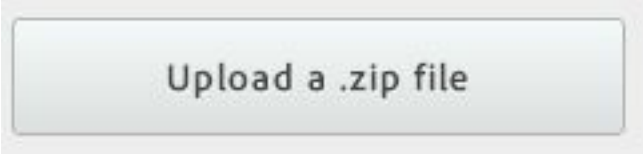
```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <widget xmlns     = "http://www.w3.org/ns/widgets"
3 xmlns:gap    = "http://phonegap.com/ns/1.0"
4 id             = "com.phonegap.example"
```

```
 5 versionCode = "10"
 6 version      = "1.0.0" >
 7
 8    <!-- versionCode is optional and Android only -->
 9
10    <name>PhoneGap Example</name>
11
12    <description>
13       An example for phonegap build docs.
14    </description>
15
16    <author href="https://build.phonegap.com" email="support@phonegap.\
17 com">
18       Hardeep Shoker
19    </author>
20
21    <!-- We'll include the Barcode plugin as an example -->
22    <gap:plugin name="com.phonegap.plugins.barcodescanner" />
23 </widget>
```

Basically you need to declare all PhoneGap plugins at the bottom using `<gap:plugin>`.

The rest of the steps should be easy:

1. Register for an account and login the Build Dashboard. For the purpose of learning, you can get the Free Plan which limits to one private app.
2. You can connect with your Github or upload a .zip file by clicking the button below



3. PhoneGap Build will automatically build for all platforms.

For Android, you should read more about app signing at PhoneGap Docs and Android Docs to understand how it works.

After the build process, the .apk file will be ready for download. This is what you need to upload to Google Play.

Again the scope of this chapter is not to cover all details of each platform but just to provide some high level concepts and some potential “gotchas”.

15. What's Next

This book has talked a lot about basic frameworks and how Ionic works in the stack. We also covered basic knowledge about AngularJS and Cordova.

In my next version of this book, I will focus more on the UI layer, specifically areas in which it could be tricky to use HTML5 and Javascript to achieve similar performance to native.

Here are some ideas:

1. **Pinterest Button Effect** - Tap the button, and it spins while popping up a layer of menu on top with bouncing animation.
2. **Twitter's Pull Down and Blur** - If you go to a profile page in a Twitter app, try to pull down, and you will see the background profile zoom in, while blurring a little bit. This is a very cool effect because you have to tighten the scrolling position to the exact frame you want to animate.
3. **Fill a Form and Send Email as a PDF Attachment** - Imagine your app needs user inputs in a form. Instead of submitting the form data to your server, you can capture all the data on the client side to generate a PDF file from it. Then the user can decide on what to do with that PDF, such as sending it via email. There is **no server** required in this whole process.
4. **Lazyload Image on Infinite Scroll** - This is an old class problem. If you look at the Pinterest app, it doesn't load all

images up front. When you start to scroll, it will *lazyload* the images on the fly.

If you have other cool ideas and want to see a tutorial, just let me know!