Build Mobile Apps with Ionic 2 and Firebase

Hybrid Mobile App Development

Fu Cheng

Apress®

Build Mobile Apps with Ionic 2 and Firebase

Hybrid Mobile App Development

Fu Cheng

Apress®

Build Mobile Apps with Ionic 2 and Firebase: Hybrid Mobile App Development

Fu Cheng
Sandringham, Auckland
New Zealand

ISBN-13 (pbk): 978-1-4842-2736-7 ISBN-13 (electronic): 978-1-4842-2737-4
DOI 10.1007/978-1-4842-2737-4

Library of Congress Control Number: 2017941053

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Technical Reviewer: Massimo Nardone
Coordinating Editor: Jessica Vakili
Copy Editor: Karen Jameson
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover image designed by Freepik

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-2736-7. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my wife Andrea and my daughter Olivia

Contents at a Glance

Contents

About the Author

Fu Cheng is a full-stack software developer working in a healthcare start-up in Auckland, New Zealand. During his many years of experience, he worked in different companies to build large-scale enterprise systems, government projects, and SaaS products. He is an experienced JavaScript and Java developer and always wants to learn new things. He enjoys sharing knowledge by writing blog posts, technical articles, and books.

About the Technical Reviewer

Massimo Nardone has more than 22 years of experience in Security, Web/Mobile development, Cloud, and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Technical skills include the following: Security, Android, Cloud, Java, MySQL, Drupal, Cobol, Perl, Web and Mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, etc.

He currently works as Chief Information Security Officer (CISO) for Cargotec Oyj.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Massimo has reviewed more than 40 IT books for different publishing companies, and he is the coauthor of *Pro Android Games* (Apress, 2015).

This book is dedicated to Antti Jalonen and his family who are always there when I need them.

Preface

Developing mobile apps is an interesting yet challenging task. Different mobile platforms have their own ecosystems. There are new programming languages, frameworks, libraries, and tools to learn. Building complicated mobile apps or games requires a lot of experience. But not all mobile apps are complicated. There are still many mobile apps that are content-centric. This kind of apps focuses on content presentations and doesn't use many native features. For these kinds of apps, PhoneGap and its successor Apache Cordova offer a different way to build them.

Mobile platforms usually have a component to render web pages. Cordova uses this component to create a wrapper for running web pages. Cordova provides different wrappers for different platforms. The web pages become the mobile apps to develop. After using Cordova, developers can use front-end skills to create cross-platform mobile apps. This empowers front-end developers to create good enough content-centric mobile apps. Many other frameworks build on top of Cordova to provide out-of-box components to make building mobile apps much easier.

This book focuses on the latest version 2 of the popular Ionic framework. The best way to learn a new framework is using it in real product development. This book is not a manual for Ionic 2, but a field guide of how to use it. We'll build a Hacker News client app using Ionic 2 and use this as the example to discuss different aspects of Ionic 2. This book not only covers the implementation of the Hacker News client app, but also the whole development life cycle, including unit tests, end-to-end tests, continuous integration, and app publish. After reading this book, you should get a whole picture of building mobile apps using Ionic 2.

Most of the nontrivial mobile apps need back-end service to work with them. Using mobile apps back-end services is a new trend that eliminates the heavy burden to write extra code and maintain the back-end infrastructure. Google Firebase is a popular choice of mobile apps back-end services. The Hacker News client app uses Firebase to handle user authentication and user favorites data storage. After reading this book, you should be able to integrate Firebase in your own apps.

Prerequisites

Ionic 2 builds on top of Angular 2 and uses TypeScript instead of JavaScript. Basic knowledge of Angular 2 and TypeScript is required to understand the code in this book. This book provides the basic introduction to Angular 2 and TypeScript, but it's still recommended to refer to other materials for more details.

To build Ionic 2 apps running on iOS platform, macOS is required to run the emulator and Xcode. You may also need real physical iOS or Android devices to test the apps.

Acknowledgments

This book would not have been possible without the help and support of many others. Thank you to my editors, Aaron Black, Jessica Vakili, and James Markham; and the rest of the Apress team, for bringing this book into the world. Thank you to my technical reviewer Massimo Nardone for your time and insightful feedback.

Many thanks to my whole family for the support during the writing of this book.

Chapter 1

Getting Started

Mobile apps development is a hot topic for both companies and individual developers. You can use various kinds of frameworks and tools to build mobile apps for different platforms. In this book, we use Ionic 2 to build so-called hybrid mobile apps. As the first chapter, this chapter provides the basic introduction of hybrid mobile apps and helps you to set up the local environment for development, debugging, and testing.

Mobile Apps Refresher

With the prevalence of mobile devices, more and more mobile apps have been created to meet all kinds of requirements. Each mobile platform has its own ecosystem. Developers use SDKs provided by the mobile platform to create mobile apps and sell them on the app store. Revenue is shared between the developers and the platform. Table 1-1 shows the statistics of major app stores at the time of writing.

***Table 1-1.** Statistics of major app stores*

App Store	Number of available apps	Downloads to date
App Store (iOS)	2.2 million	140 billion
Google Play	2.6 million	65 billion
Windows Store	669,000+	--
BlackBerry World	240,000+	4 billion
Amazon Appstore	334,000+	--

F. Cheng, *Build Mobile Apps with Ionic 2 and Firebase*,
DOI 10.1007/978-1-4842-2737-4_1

The prevalence of mobile apps also creates a great opportunity for application developers and software companies. A lot of individuals and companies make big money on the mobile apps markets. A classic example is the phenomenal mobile game Flappy Bird. Flappy Bird was developed by Vietnam-based developer Dong Nguyen. The developer claimed that Flappy Bird was earning $50,000 a day from in-app advertisements as well as sales. Those successful stories encourage developers to create more high-quality mobile apps.

Let's now take a look at some key components of mobile app development.

Hybrid Mobile Apps

Developing mobile apps is not an easy task. If you only want to target a single mobile platform, then the effort may be relatively smaller. However, most of the times we want to distribute apps on many app stores to maximize the revenue. To build that kind of apps which can be distributed to various app stores, developers need to use different programming languages, SDKs, and tools, for example, Objective-C/Swift for iOS and Java for Android. We also need to manage different code bases with similar functionalities but implemented using different programming languages. It's hard to maximize the code reusability and reduce code duplications across different code bases, even for the biggest players in the market. That's why cross-platform mobile apps solutions, like Xamarin (https://www.xamarin.com/), React Native (https://facebook.github.io/react-native/), and RubyMotion (http://www.rubymotion.com/) also gain a lot of attention. All these solutions have a high learning curve for their programming languages and SDKs, which creates a burden for ordinary developers.

Comparing to Objective-C/Swift, Java, C# or Ruby, web development skills, for example, HTML, JavaScript, and CSS are much easier to learn. Building mobile apps with web development skills is made possible by HTML5. This new type of mobile apps is called hybrid mobile apps. In hybrid mobile apps, HTML, JavaScript, and CSS code run in an internal browser (WebView) that is wrapped in a native app. JavaScript code can access native APIs through the wrapper. Apache Cordova (https://cordova.apache.org/) is the most popular open source library to develop hybrid mobile apps.

Compared to native apps, hybrid apps have their benefits and drawbacks. The major benefit is that developers can use existing web development skills to create hybrid apps and use only one code base for different platforms. By leveraging responsive web design techniques, hybrid apps can easily adapt to different screen resolutions. The major drawback is the performance issues with hybrid apps. As the hybrid app is running inside of an internal browser, the performance of hybrid apps cannot compete with native apps. Certain types of apps, such as games or apps that rely on complicated native functionalities, cannot be built as hybrid apps. But many other apps can be built as hybrid apps.

Before making the decision of whether to go with native apps or hybrid apps, the development team needs to understand the nature of the apps to build. Hybrid apps are suitable for content-centric apps, such as news readers, online forums, or showcasing products. Another important factor to consider is the development team's skill sets. Most apps companies may need to hire both iOS and Android developers to support these two major platforms for native apps. But for hybrid apps, only front-end developers are enough. It's generally easier to hire front-end developers than Java or Swift/Objective-C developers.

Apache Cordova

Apache Cordova is a popular open source framework to develop hybrid mobile apps. It originates from PhoneGap (http://phonegap.com/) created by Nitobi. Adobe acquired Nitobi in 2011 and started to provide commercial services for it. The PhoneGap source code was contributed to the Apache Software Foundation and the new project Apache Cordova was started from its code base.

An Apache Cordova application is implemented as a web page. This web page can reference JavaScript, CSS, images, and other resources. The key component of understanding how Cordova works is the `WebView`. `WebView` is the component provided by native platforms to load and run web pages. Cordova applications run inside the `WebViews`. A powerful feature of Cordova is its plugin interface which allows JavaScript code running in a web page to communicate with native components. With the help of plugins, Cordova apps can access a device's accelerometer, camera, compass, contacts, and more. There are already many plugins available in Cordova's plugin registry (http://cordova.apache.org/plugins/).

Apache Cordova is just a runtime environment for web apps on native platforms. It can support any kind of web pages. To create mobile apps that look like native apps, we need other UI frameworks to develop hybrid mobile apps. Popular choices of hybrid mobile apps UI frameworks include Ionic framework (http://ionicframework.com/), Sencha Touch (https://www.sencha.com/products/touch/), Kendo UI (http://www.telerik.com/kendo-ui), and Framework7 (http://framework7.io/). Ionic framework is the one we are going to cover in this book.

Ionic Framework

Ionic framework is a powerful tool to build hybrid mobile apps. It's open source (https://github.com/driftyco/ionic) and has over 28,500 stars on GitHub, the popular social coding platform. Ionic framework is not the only player in hybrid mobile apps development, but it's the one that draws a lot

of attention and is recommended as the first choice by many developers. Ionic is popular for the following reasons:

- Use Angular (https://angular.io/) as the JavaScript framework. Since Angular is a popular JavaScript framework, the large number of Angular developers find it quite easy when moving to use Ionic for mobile apps development.
- Provide beautifully designed out-of-box UI components that work across different platforms. Common components include lists, cards, modals, menus, and pop-ups. These components are designed to have a similar look and feel as native apps. With these built-in components, developers can quickly create prototypes with good enough user interfaces and continue to improve them.
- Leverage Apache Cordova as the runtime to communicate with native platforms. Ionic apps can use all the Cordova plugins to interact with the native platform. Ionic Native further simplifies the use of Cordova plugins in Ionic apps.
- Performs great on mobile devices. The Ionic team devotes great effort to make it perform great on different platforms.

Ionic 2 (http://ionic.io/2) is a completely rewritten version of Ionic framework based on Angular 2. It dramatically improves performance and reduces the complexity of the code. It's recommended to use this new version of Ionic framework to build hybrid mobile apps. This book uses the `2.0.1` version of Ionic 2.

Firebase

Mobile apps usually need back-end services to work with the front-end UI. This means that there should be back-end code and servers to work with mobile apps. Firebase (https://firebase.google.com/) is a cloud service to power apps' back-end. Firebase can provide support for data storage and user authentication. After integrating mobile apps with Firebase, we don't need to write back-end code or manage the infrastructure.

Firebase works very well with Ionic to eliminate the pain of maintaining back-end code. This is especially helpful for hybrid mobile apps developers with only front-end development skills. Front-end developers can use JavaScript code to interact with Firebase.

Prepare Your Local Development Environment

Before we can build Ionic apps, we need to set up the local development environment first. We'll need various tools to develop, test, and debug Ionic apps.

Node.js

Node.js is the runtime platform for Ionic CLI. To use Ionic CLI, we need to install Node.js (https://nodejs.org/) on the local machine first. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It provides a way to run JavaScript on the desktop machines and servers. Ionic CLI itself is written in JavaScript and executed using Node.js. There are two types of release versions of Node.js – the stable LTS versions and current versions with the latest features. It's recommended to use Node.js version 6 or greater, especially the latest LTS version (6.9.4 at the time of writing).

Installing Node.js also installs the package management tool npm. npm is used to manage Node.js packages used in projects. Thousands of open source packages can be found in the npmjs registry (https://www.npmjs.com/). If you have a background with other programming languages, you may find npm is similar to Apache Maven (https://maven.apache.org/) for Java libraries or Bundler (http://bundler.io/) for Ruby gems.

Ionic CLI

After Node.js is installed, we can use npm to install Ionic command-line tools and Apache Cordova.

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

> **Note** You may need to have system administrator privileges to install these two packages. For Linux and macOS, you can use `sudo`. For Windows, you can start a command-line window as the administrator. However, it's recommended to avoid using `sudo` when possible, as it may cause permission errors when installing native packages. Treat this as the last resort. The permission errors usually can be resolved by updating the file permissions of the Node.js installation directory.

After finishing installation of Ionic CLI and Cordova, we can use the command `ionic` to start developing Ionic apps.

You are free to use Windows, Linux, or macOS to develop Ionic 2 apps. Node.js is supported across different operating systems. One major limitation of Windows or Linux is that you cannot test iOS apps using the emulator or real devices. Some open source Node.js packages may not have the same test coverage on Windows as Linux or macOS. So they are more likely to have issues when running on Windows. But this should only affect the CLI or other tools, not Ionic 2 itself.

Yarn

Yarn (https://yarnpkg.com/) is a fast, reliable, and secure dependency management tool. After Facebook open sourced it, it quickly became popular in the Node.js community as a better alternative to npm. If you want to use yarn, follow the official instructions (https://yarnpkg.com/en/docs/install) to install it. After installing yarn, we can use the following command to install Ionic CLI and Cordova.

```
$ yarn global add cordova ionic
```

This book uses yarn instead of npm. If you didn't know about yarn before, read this guide (https://yarnpkg.com/en/docs/migrating-from-npm) about how to migrate from npm to yarn. Common yarn commands are listed below:

- yarn add [package] – Add packages as the project's dependencies. You can provide multiple packages to install. Version requirement can be specified following the Semantic Versioning spec (http://semver.org/).
- yarn upgrade [package] – Upgrade or downgrade versions of packages.
- yarn remove [package] – Remove packages.
- yarn global – Manage global dependencies.

The file yarn.lock contains the extract version of all resolved dependencies. This file is to make sure that builds are consistent across different machines. This file should be managed in the source code repository.

After Ionic CLI is installed, we can run `ionic info` to print out current runtime environment information and check for any warnings in the output; see Listing 1-1. The output also provides details information about how to fix those warnings.

Listing 1-1. Output of ionic info

```
Your system information:

Cordova CLI: 6.4.0
Ionic Framework Version: 2.0.1
Ionic CLI Version: 2.2.1
Ionic App Lib Version: 2.1.7
Ionic App Scripts Version: 1.0.0
ios-deploy version: 1.9.0
ios-sim version: 5.0.11
OS: macOS Sierra
Node Version: v6.9.4
Xcode version: Xcode 8.2.1 Build version 8C1002
```

iOS

Developing iOS apps with Ionic requires macOS and Xcode. You need to install Xcode and Xcode command-line tools on macOS. After installing Xcode, you can open a terminal window and type the command shown below.

```
$ xcode-select -p
```

If you see output like below, then command-line tools have already been installed.

```
/Applications/Xcode.app/Contents/Developer
```

Otherwise, you need to use the following command to install it.

```
$ xcode-select --install
```

After the installation is finished, you can use `xcode-select -p` to verify.

To run Ionic apps on the iOS simulator using Ionic CLI, package `ios-sim` is required. Another package `ios-deploy` is also required for deploying to install and debug apps. You can install both packages using the following command.

```
$ yarn global add ios-sim ios-deploy
```

Android

To develop Ionic apps for Android, Android SDK must be installed. Before installing Android SDK, you should have JDK installed first. Read this guide (https://docs.oracle.com/javase/8/docs/technotes/guides/install/) about how to install JDK 8 on different platforms. It's recommended to install

Android Studio (https://developer.android.com/studio/index.html) that provides a nice IDE and bundled Android SDK tools. If you don't want to use Android Studio, you can install stand-alone SDK tools.

> **Note** Android API level 22 is required to run Ionic apps. Make sure that the required SDK platform is installed.

Stand-alone SDK tools is just a ZIP file; unpack this file into a directory and it's ready to use. The downloaded SDK only contains basic SDK tools without any Android platform or third-party libraries. You need to install the platform tools and at least one version of the Android platform. Run `android` in `tools` directory to start **Android SDK Manager** to install platform tools and other required libraries.

After installing Android SDK, you need to add SDK's `tools` and `platform-tools` directories into your `PATH` environment variable, so that SDK's commands can be found by Ionic. Suppose that the SDK tools is unpacked into `/Development/android-sdk`, then add `/Development/android-sdk/tools` and `/Development/android-sdk/platform-tools` to `PATH` environment variable. For Android Studio, the Android SDK is installed into directory `/Users/<username>/Library/Android/sdk`.

To modify PATH environment variable on Linux and macOS, you can edit `~/.bash_profile` file to update `PATH` as shown below.

```
export PATH=${PATH}:/Development/android-sdk/platform-tools \
  :/Development/android-sdk/tools
```

To modify `PATH` environment variable on Windows, you can follow the steps below.

1. Click **Start** menu, then right-click **Computer** and select **Properties**.
2. Click **Advanced System Settings** to open a dialog.
3. Click **Environment Variables** in the dialog and find **PATH** variable in the list, then click **Edit**.
4. Append the path of `tools` and `platform-tools` directories to the end of **PATH** variable.

It is highly recommended to use Android Studio instead of stand-alone SDK tools. Stand-alone SDK tools are more likely to have configuration issues.

Genymotion

Genymotion (https://www.genymotion.com/) is a fast Android emulator. It's recommended to use Genymotion for Android emulation instead of the standard emulators.

IDEs and Editors

You are free to use your favorite IDEs and editors when developing Ionic apps. IDEs and editors should have good support for editing HTML, TypeScript, and Sass files. For commercial IDEs, WebStorm (https://www.jetbrains.com/webstorm/) is recommended for its excellent support of various programming languages and tools. For open source alternatives, Visual Studio Code (https://code.visualstudio.com/) and Atom (https://atom.io/) are both popular choices.

Create an App Skeleton

After the local development environment is set up successfully, it's time to create new Ionic apps. Ionic 2 provides four different types of application templates. We can choose a proper template to create the skeleton code of the app. Apps are created using the command `ionic start`. The first argument of `ionic start` is the name of the new app, while the second argument is the template name. The `--v2` flag is also required when creating Ionic 2 apps; otherwise, Ionic 1 apps will be created instead.

The source code of these templates can be found on GitHub (https://github.com/driftyco?query=ionic2-starter-). All these templates follow the same naming convention. The template names all start with `ionic2-starter-`. After removing the prefix `ionic2-starter-`, we can get the template name to be used in the command `ionic start`. For example, `ionic2-starter-blank` is the name of the template for blank apps.

Blank App

The template `blank` (https://github.com/driftyco/ionic2-starter-blank) only generates basic code for the app. This template should be used when you want to start from a clean code base; see Figure 1-1.

```
$ ionic start blankApp blank --v2
```