*Ionic in Action*

by Jeremy Wilken

**Chapter 1**

# *brief contents*

# 1 Introducing Ionic and hybrid apps

**This chapter covers**

- Why you should choose Ionic and how it benefits you
- What Ionic is and how it uses Angular and Cordova
- Why hybrid apps are an ideal choice for mobile development
- Introduction and requirements for Android and iOS platforms

Building mobile apps has become an essential skill for many developers, and with Ionic you'll be able to build hybrid mobile apps that look and feel just like native mobile apps. A *hybrid app* is a type of mobile app that uses a browser window to display its interface. *Ionic* is a combination of tools and utilities that enables developers to quickly build hybrid mobile apps using the same technologies used to build websites and web applications, primarily HTML, CSS (Cascading Style Sheets), and JavaScript. Ionic works by embedding a web application inside of a native app by using Cordova. It's designed to work together with Angular to create

a web application for the mobile environment, and includes support for mobile features like user interface controls and responding to touch input.

This book aims to give developers the skills necessary to build Ionic mobile apps. I'll teach you the basics of setting up your projects correctly and how to build rich interfaces, and demonstrate with real-world style examples. I'll help you set up your build, testing, and deployment processes to get your app ready for production. But before we get too far along, we should dig deeper into Ionic and why it's a solid choice for building hybrid mobile apps.

## 1.1 What is Ionic?

Ionic is a combination of technologies and utilities designed to make building hybrid mobile apps fast, easy, and beautiful. Ionic is built on an ecosystem that includes Angular as the web application framework and Cordova for building and packaging the native app. We'll dig into each in more detail later, but figure 1.1 shows you an overview of these technologies and how they stack. Let's take a moment to cover the basics of how the technology stack works on a device.

In figure 1.1, the stack begins with the user opening the app from the device. Imagine this is an iPhone running iOS or a Nexus 10 running Android. Let's break down each of these pieces in more detail:

- *Device*—This loads the app. The device contains the operating system that manages the installation of apps that are downloaded from the platform's store. The operating system also provides a set of APIs for apps to use to access various features, such as the GPS location, contacts list, or camera.
- *Cordova app wrapper*—This is a native app that loads the web application code. Cordova is a platform for building mobile apps that can run using HTML, CSS, and JavaScript inside of a native app, which is known as a *hybrid mobile app*. It's a utility for creating a bridge between the platform and the application. It creates a native mobile app that can be installed (called the *app wrapper* in figure 1.1),

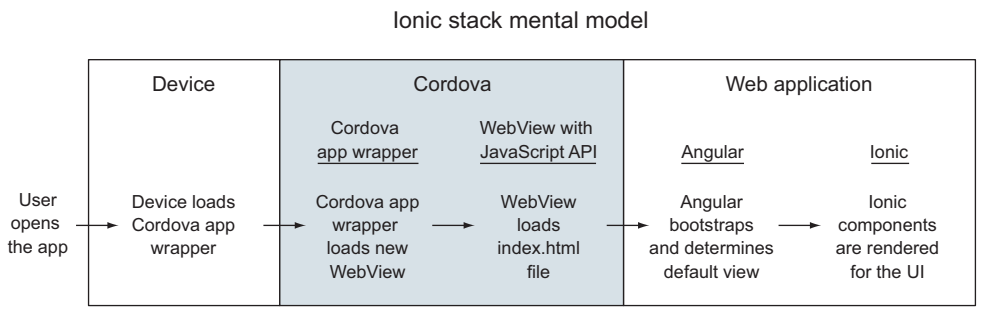


**Figure 1.1 The stack of technologies used with the Ionic framework, and how they fit together**

and it contains what's called a WebView (essentially an isolated browser window) with a JavaScript API that the web application will run inside.

- *Cordova JavaScript API*—This is the bridge that communicates between the app and the device. The app wrapper has access to both the web application and the native platform through the JavaScript API. This is primarily handled behind the scenes, and Cordova ultimately generates the native app for you.
- *Angular*—This is the web application that controls the app routing and function. The Angular web application runs inside of the WebView. Angular is a very popular framework for building powerful web applications. Angular is primarily used to manage the web application's logic and data.
- *Ionic*—This provides the user interface components rendered in the app. Ionic is built on top of Angular, and is primarily used to design the user interface and experience. This includes the visual elements such as tabs, buttons, and navigation headers. These interface controls are the heart of Ionic, and provide a near-native interface inside of a hybrid app. Ionic also includes a number of additional utilities and features that help manage your app from creation to previewing to deployment.

The combination of these technologies makes Ionic a very feature-rich platform for building your mobile apps. Now that you have a bird's-eye view of Ionic and the technology, let's look a little closer at three main types of mobile experiences and why Ionic's approach is beneficial.

## 1.2 Types of mobile experiences

It's important to understand there are several ways to build applications for mobile devices, and each has its strengths and weaknesses. There are three basic types: native apps, mobile websites, and hybrid apps. We'll look at each of these in detail to clarify the differences.

In figure 1.2, you can see how the three types compare in design and architecture. The figure also shows how each app would access a database or web service API to load data.

### 1.2.1 Native mobile apps

To create native apps, developers write code in the default language for the mobile platform, which is Objective C or Swift for iOS and Java for Android. Developers compile the app and install it on a device. Using the platform software development kit (SDK), the app communicates with the platform APIs to access device data or load data from an external server using HTTP requests.

Both iOS and Android provide a set of tools to enable developers to leverage the platform features in a controlled manner through predefined APIs. There are tools, both official and unofficial, that can aid in the development of native apps. It's common for developers to use frameworks in their native app to make development easier.

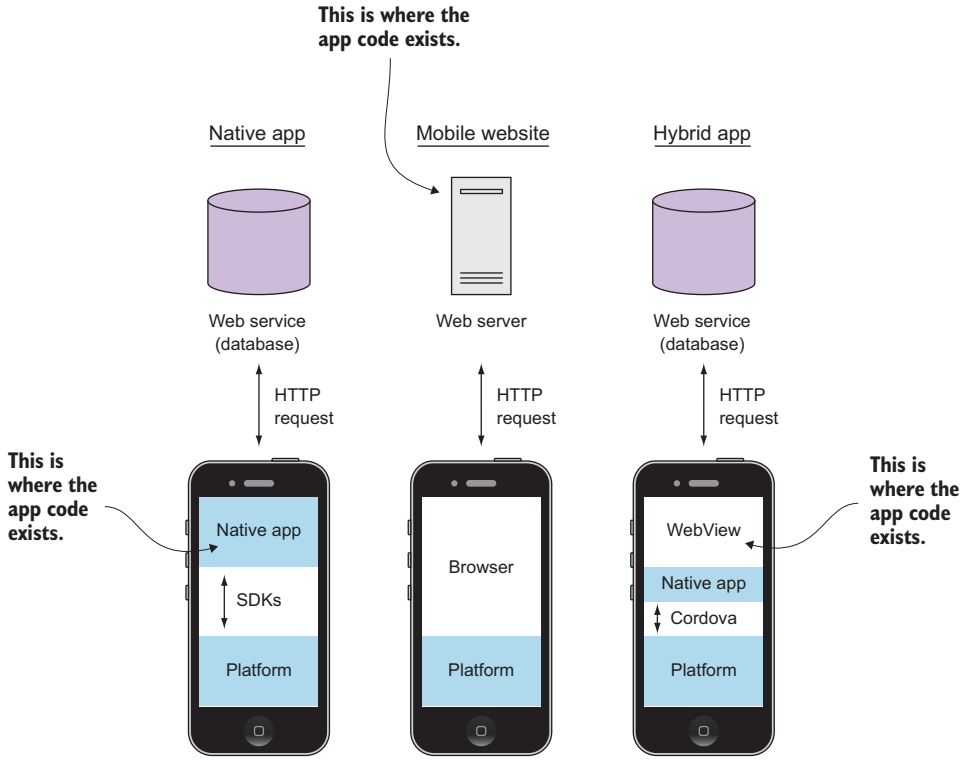


**Figure 1.2 Native apps, mobile websites, and hybrid app architectures compared side by side**

### NATIVE APP ADVANTAGES

Native apps come with a number of benefits over hybrid apps and mobile websites. The benefits revolve around being tightly integrated with the device platform:

- *Native APIs*—Native apps can use the native APIs directly in the app, making the tightest connection to the platform.
- *Performance*—They can experience the highest levels of performance.
- *Same environment*—They're written with native APIs, which is helpful for developers familiar with the languages used.

But there are also a number of disadvantages.

### NATIVE APP DISADVANTAGES

The disadvantages of native apps are generally the level of difficulty in developing and maintaining them:

- *Language requirements*—Native apps require developer proficiency in the platform language (for example, Java) and knowledge of how to use platform-specific APIs.
- *Not cross-platform*—They can only be developed for one platform at a time.

- *High level of effort*—Typically, they require more work and overhead to build, which increases costs.

Native apps may be best suited for developers who have a command of Java and Objective C, or for teams with extensive resources and a need for the benefits of native apps.

### 1.2.2 Mobile websites (web apps)

Mobile websites, or web apps, work well on a mobile device and are accessed through a mobile browser. Web apps are websites viewed on a mobile device in a mobile browser, designed specifically to fit a mobile device screen size. Figure 1.3 shows a couple of examples.

Some website designers develop a second version specifically for use on a mobile device. Perhaps you've used your mobile device to visit a website and were redirected to a version with limited features, such as visiting eBay and ending up on the http://m.ebay.com subdomain. On other websites, such as www.bostonglobe.com, you may

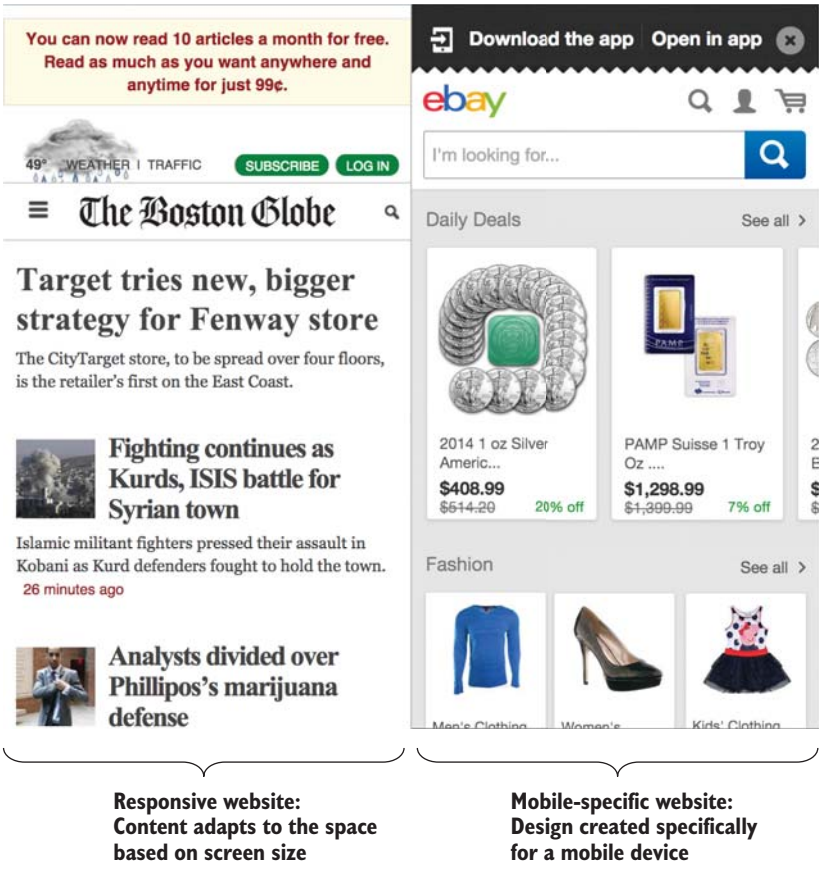


**Figure 1.3 Mobile websites: a responsive site from the *Boston Globe* (left) and a mobile-specific website from eBay (right)**

find that the design adjusts to your device's form factor and screen size. This is accomplished with a technique called *responsive design*. The website content will resize and flow according to the browser window size, and some may even be hidden.

#### MOBILE WEBSITE ADVANTAGES

Mobile websites enjoy a number of benefits, primarily in the level of effort and compatibility on devices:

- *Maintainability*—Mobile websites are easy to update and maintain without the need to go through an approval process or update installations on devices.
- *No installation*—Because they exist on the internet, they don't require installation on mobile devices.
- *Cross-platform*—Any mobile device has a browser, allowing your application to be accessible from any device.

As with native apps, there are also a number of disadvantages.

#### MOBILE WEBSITE DISADVANTAGES

Mobile websites run inside of a mobile browser, which is the major cause of limitations and disadvantages:

- *No native access*—Because mobile websites are run in the browser, they have no access to the native APIs or the platform, just the APIs provided by the browser.
- *Require keyboard to load*—Users have to type the address in a browser to find or use a mobile website, which is more difficult than tapping an icon.
- *Limited user interface*—It's difficult to create touch-friendly applications, especially if you have a responsive site that has to work well on desktops.
- *Mobile browsing decline*—The amount of time users browse the web on a mobile device is declining, while app use is increasing.

Mobile websites can be important even if you have a mobile app, depending on your product or service. Research shows users spend much more time using apps compared to the mobile browser, so mobile websites tend to experience lower engagement.

### 1.2.3 Hybrid apps

A hybrid app is a mobile app that contains an isolated browser instance, often called a *WebView*, to run a web application inside of a native app. It uses a native app wrapper that can communicate with the native device platform and the WebView. This means web applications can run on a mobile device and have access to the device, such as the camera or GPS features.

Tools that facilitate the communication between the WebView and the native platform make hybrid apps possible. These tools aren't part of the official iOS or Android platforms, but are third-party tools such as Apache Cordova, which is used in this book. When a hybrid app compiles, your web application transforms into a native app.

#### HYBRID APP ADVANTAGES

Hybrid apps have a few advantages over mobile websites and native apps that make hybrid apps a great platform for building apps:

- *Cross-platform*—You can build your app once and deploy it to multiple platforms with minimal effort.
- *Same skills as web development*—They allow you to build mobile apps using the same skills already used to develop websites and web applications.
- *Access to device*—Because the WebView is wrapped in a native app, your app has access to all of the device features available to a native app.
- *Ease of development*—They're easy and fast to develop, without the need to constantly rebuild to preview. You also have access to the same development tools used for building websites.

Hybrid apps provide a robust base for mobile app development, yet still allow you to use the web platform. You can build the majority of your app as a website, but anytime you need access to a native API, the hybrid app framework provides a bridge to access that API with JavaScript. Your app can detect swipes, pinches, and other gestures just like clicks or keyboard events. But there are a few disadvantages, as you might expect.

#### HYBRID APP DISADVANTAGES

Hybrid apps have a few disadvantages due to the restrictions that are placed on WebViews and limitations of native integrations:

- *WebView limitations*—The application can only run as well as the WebView instance, which means performance is tied to the quality of the platform's browser.
- *Access native features via plugins*—Access to the native APIs you need may not be currently available, and may require additional development to make a plugin to support it.
- *No native user interface controls*—Without a tool like Ionic, developers would have to create all of the user interface elements.

With Ionic, you can build hybrid apps so you can leverage the knowledge and skills with which web developers are already familiar.

## 1.3 Understanding how the Ionic stack works

There are several technologies that can be used when building hybrid apps, but with Ionic there are three primary ones: Ionic, Angular, and Cordova. Figure 1.4 outlines how these pieces can work in tandem to facilitate opening the camera from an Ionic app.

Let's break down each of the steps in figure 1.4:

1. The user taps on a button (which is an Ionic component).
2. The button calls the Angular controller, which calls Cordova through the JavaScript API.

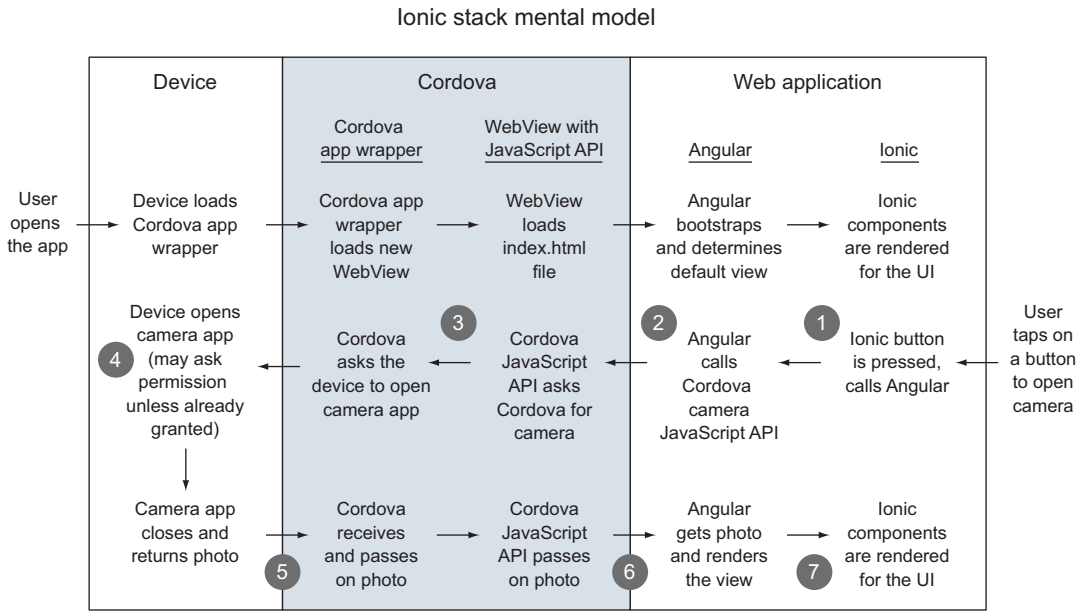


**Figure 1.4 How Ionic, Angular, and Cordova work together for a hybrid app**

3. Cordova communicates with the device using native SDKs and requests the camera app.
4. The device opens the camera app (or prompts for permission if necessary), and the user is able to take a picture.
5. When the user confirms the photo, the camera app closes and returns the image data to Cordova.
6. Cordova passes the image data back to the Angular controller.
7. The visual display of the image is updated inside of Ionic components.

This quick outline of how the pieces communicate can demonstrate how an Ionic app is really a stack of technologies that work in concert. Don't worry if some of these terms are unknown to you—we'll cover them throughout this book. The key here is to see how your app is able to leverage the power of the device. Let's look at each one more closely.

### 1.3.1 Ionic: user interface framework

Ionic's primary feature is a set of user interface controls that are missing from HTML but are common on mobile apps. Imagine a weather app that shows current conditions based on the user's location. Ionic provides a number of user interface components such as a slidebox that allows a user to swipe between several boxes of information like temperature, forecasts, and weather maps. These components are

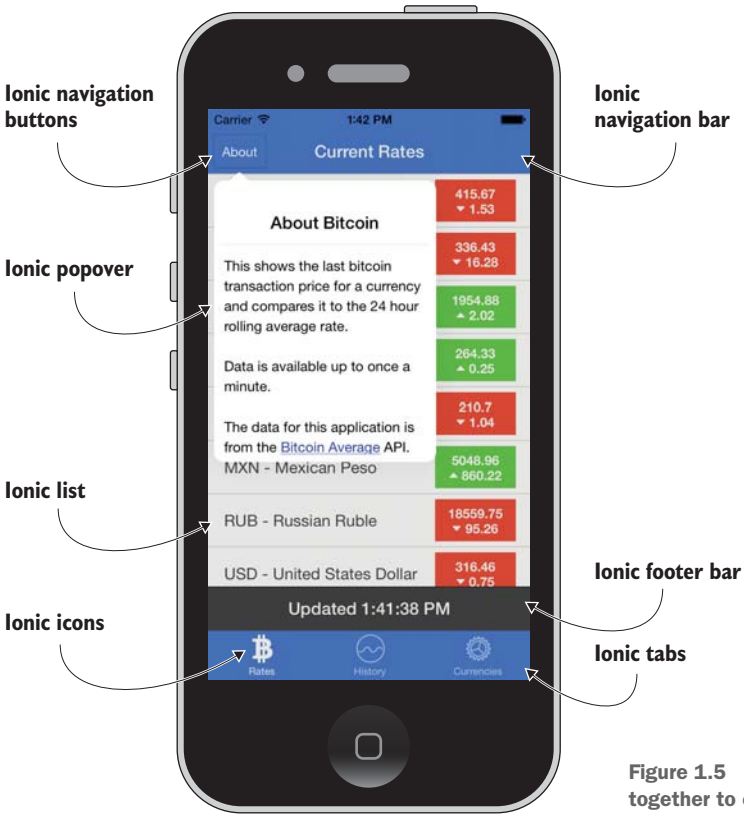


**Figure 1.5 How parts of Ionic work together to create a usable interface**

built with a combination of CSS, HTML, and JavaScript, and they behave like the native controls you're accustomed to using. Common examples include these:

- Side menus that slide in from the side
- Toggle buttons
- Mobile tabs

In figure 1.5 you can see a screenshot of one of the sample apps you'll build later in the book. It shows how many different Ionic components are used on the screen at once to create a powerful user interface.

Ionic is an open source project that's primarily developed by the Ionic team. Its popularity has grown very quickly since it was launched in November 2013; it has become a primary choice for building hybrid apps. Over 20,000 apps are launched with Ionic each month. Ionic is provided under the MIT license and is found at http://ionicframework.com.

Ionic also has a command-line interface (CLI) tool that provides some helpful developer tools. I'll refer to it as the CLI tool. This tool can help generate starter projects, and

preview, build, and deploy your app. I'll demonstrate most of the features of the CLI tool as we go through the examples.

Ionic also includes a font icon library that gives you access to a decent number of useful and common icons for your application. It's optional, but it's provided by default and we'll use it regularly in the examples.

Ionic also has a number of services that aid in mobile app development, such as a visual drag-and-drop app creator and deployment tooling, user tracking and analytics, and push notifications. You can learn more about the full Ionic platform at https://ionic.io.

These user interface controls are the primary features of Ionic, but the Ionic team has worked hard to ensure Ionic's tools and processes work well with Angular and Cordova, which are discussed next.

### 1.3.2 Angular: web application framework

Angular (also known as AngularJS) is a Google open source project that has become quite popular with web application developers. It provides web developers a good application structure and the ability to write complete applications quickly. In the weather app example in this book, you'll use Angular to help manage the user's data and load information from the weather service.

Miško Hevery and Adam Abrons started Angular in 2009. Eventually, Hevery joined Google and brought Angular with him. The project is immensely popular with developers today, and has been adopted by a number of large sites such as www.stackoverflow.com and www.nasa.gov. Angular is licensed under the MIT license and is available at http://angularjs.org.

You no longer have to use a server-based language (that is, PHP, Ruby, or Java) to build complex applications. Today, JavaScript web application frameworks like Angular allow you to build complex applications in the browser. Typically a server application also exists to help manage private data and secure any business logic. This is an obvious advantage for hybrid app developers because the browser is the platform you use to create your apps. If you're familiar with Angular (or other JavaScript application frameworks such as Ember or Backbone), you'll be able to easily apply your knowledge to developing mobile apps with Ionic.

In this book we'll also use additional Angular modules that have been developed by third-party developers. One notable example is a module called `ui.router`, which is an open source Angular module that provides better application routing and navigation than the default Angular routing module offers.

### 1.3.3 Cordova: hybrid app framework

In this book we'll use Apache Cordova as the hybrid app framework. This is the layer that takes care of managing the communication between the browser window and native APIs. The weather-app example needs access to the device's GPS information to know what location to load data for, and Cordova is able to bridge the gap between Angular and the device to retrieve that information.

You may also have heard of PhoneGap. Adobe contributed PhoneGap to the Apache Software Foundation under the name Cordova. Today, PhoneGap is a distribution of Cordova, or, in other words, PhoneGap is essentially Cordova with support for a few additional commercial features from Adobe. For the purposes of this book, we'll use Cordova, but you could use PhoneGap and its commercial features if you desire.

Cordova is an open source Apache project that has a large community around it. Adobe continues to be a major developer of the framework. Cordova is licensed under the Apache 2.0 license.

The core of Cordova provides a lot of features; it also provides a plugin system for developers to create new features such as native API integrations with the phone camera. It's actively maintained with regular releases of improvements and new features. You can find out more about Cordova at http://cordova.apache.org.

Ionic has also sponsored the creation of a project called ngCordova at http://ngcordova.com. ngCordova is a collection of nicely integrated Cordova plugins designed to work well with Angular. Chapter 8 covers more details about Cordova and plugins, and you'll see some examples from the ngCordova project.

## 1.4 Why Ionic?

Ionic brings a new and important set of improvements to hybrid apps that other tools like jQuery Mobile haven't been able to provide. Until recently, mobile devices were still relatively sluggish and only a native app could deliver the performance and experience many developers wanted or needed. Mobile platform makers hadn't made browsers as fast as the native platforms. All of that has changed as devices have become more powerful, platforms have improved, and new tools like Ionic have made it possible to build amazing hybrid apps.

### 1.4.1 Why Ionic is good for developers

Ionic is able to provide an experience—built into the hybrid app—that looks, feels, and performs like a native app. The long-standing argument that native apps are the only way to get fast and richly featured apps has been proven wrong. People expect their mobile apps to be fast, smooth, and intuitive, and Ionic apps can deliver:

- *Build apps with the web platform*—Using HTML, CSS, and JavaScript, you can make hybrid apps that behave like native mobile apps.
- *Built with Angular*—For developers familiar with Angular (or even another JavaScript framework like Ember), Ionic is a great choice. Because Ionic is built with Angular, you have access to all of Angular's features and third-party modules. Angular is designed to build major applications, and Ionic extends Angular for the mobile environment.
- *Uses modern techniques*—Ionic was designed to work with modern CSS3 features like animations. Mobile browsers generally have better support for the latest web platform specifications, which allows you to use those features as well.

- *Engaged community and open source spirit*—The Ionic community is very active on forums, with code contributions, and in sharing tips and tricks about the platform. The open source spirit is alive and well within the project.
- *Powerful CLI tool*—With the Ionic CLI tool, you can quickly manage development tasks such as previewing the app in a browser, emulating the app, or deploying an app to a connected device. It helps with setting up and starting a project as well.
- *Ionic services*—Ionic also provides services that make development much easier. The Ionic Creator service allows you to use a drag-and-drop interface to design and export an app. The Ionic View service allows you to deploy an app beta release to customers or test users. In short, Ionic is all about creating not just the basic tools for making hybrid apps, but also the development tools that will help you create them efficiently.
- *Ionic has a dedicated team*—Open source projects can be difficult to select because you can't be sure if they will be properly developed or supported. Ionic has a dedicated team that has a vested interest in keeping the platform on the leading edge.
- *Native-like experience*—With Ionic, you can create a look and feel that's like the native apps, making it easier for your customers to use the app.
- *Performance*—The performance with Ionic is comparable to a native app; the better the app performs, the happier app users will be.
- *Beautiful, flexible design*—The user interface components have been carefully designed to implement native style guidelines, but also allow for easy customization of any visual aspect of the app.

With Ionic, you can craft feature-rich apps for your customers that take you far less time and effort to create. This can provide great value for you, your team, and your app users.

### 1.4.2 Drawbacks of using Ionic

Ionic isn't always the right solution for your needs. It's important to evaluate the needs of each project to ensure Ionic is the right solution for you:

- *Limited platforms*—Ionic 1.0 only fully supports iOS and Android platforms. Other platforms such as Windows Phone or Firefox OS may be fully supported in the future but aren't guaranteed. Apps may still function on other platforms, but Ionic isn't actively supporting them.
- *Older platforms not supported*—Ionic supports iOS 7+ and Android 4+ properly. Older versions may work properly and aren't actively tested. This can be a challenge if your app needs to run on old or low-spec devices.
- *Not equal to native*—The native device APIs are only available if Cordova supports them. If you need deep integration with the device, it may be more difficult to achieve.

- *Not geared for heavy graphics*—This is a limitation of hybrid apps in general because they run in the browser. If you have a game app or heavy graphic requirements, the hybrid app environment has fewer abilities compared to a native app environment.

There may be situations where your app requirements force you to choose something other than Ionic, but even in those cases Ionic can be a very useful tool during the early prototyping phase.

## 1.5 Prerequisites for building apps with Ionic

To build hybrid apps, you should have a few skills that aren't covered in this book. You don't need to be an expert in any of the following areas, but you should be prepared to use them all together.

### 1.5.1 Experience with HTML, CSS, and JavaScript

If you've built a website, you've used the web platform. The browser is like the operating system that you'll use to develop the sample mobile apps in this book. HTML, CSS, and JavaScript are the key languages the browser understands. HTML gives structure to the content, while CSS provides the design. JavaScript then provides the interaction and logic necessary for the web application.

You'll need to be familiar with JavaScript syntax and concepts such as asynchronous calls, events, prototypical inheritance, and variable scoping.

### 1.5.2 Experience with web applications and Angular

You should have a fundamental understanding of web applications, because we'll build them inside of the sample mobile apps. There are a number of technologies and libraries that developers use to build web applications, and familiarity with the concepts will help you greatly.

In this book, web apps will be written in JavaScript using the Angular framework. Ionic is built specifically to work with Angular, and developers who have experience building applications with Angular will be able to apply their experience easily. You might have experience with another framework, such as Ember or Backbone, that can provide a foundation as you learn the Angular-specific approach.

We'll cover a bit about Angular in chapter 3 to get you up and running, but this isn't a book about Angular. You'll want to refer to the books *AngularJS in Action* (http://manning.com/bford) and *AngularJS in Depth* (http://manning.com/aden) to learn everything you want about Angular beyond the scope of this book.

### 1.5.3 Access to a mobile device

Having a mobile device is extremely important when building a mobile app. I recommend that you have at least one device for every platform to test on an actual device. There are emulators that let you see what your apps should look like on a mobile device, but they aren't full substitutes for the real thing.

You'll have to register these devices with your developer accounts as well, so it's not practical to borrow. If you need a device, you can check for a refurbished or used item online and use it just for development testing. The more types of devices you can test your app with, the better.

These three prerequisites will help you be more successful at designing, testing, and building mobile apps across multiple platforms. Let's take a look at the mobile platforms Ionic supports.

## 1.6 Supported mobile devices and platforms

A number of mobile platforms—OS, Android, Windows 8, Firefox OS, Tizen, Blackberry, and more—are in use. With Ionic, you can build for both iOS and Android. Support for Windows 8 and Firefox OS is planned for the future, but isn't currently available.

While it may be possible to develop an app by previewing only on a simulator, devices can act differently in the real world. Let's take a closer look at these two primary platforms and requirements.

### 1.6.1 Apple iOS

Apple makes the popular iPhone and iPad devices, and they share a common platform called iOS. Apple has strong control over the entire experience from the devices to the software to the apps, essentially making it a closed system. This has made iOS a strong platform from the perspective of users and developers.

Apple provides Xcode as the primary development program for iOS and OS X development. Xcode is free and available in the App Store if you don't already have it downloaded. We'll cover setting up for iOS development in the next chapter.

Xcode comes with a set of simulators that allows you to simulate different versions of iPhones and iPads. The simulators are fairly good at giving a realistic experience, which is helpful when targeting multiple versions of iOS with the same app.

Apple has one major requirement for building iOS mobile apps: you need a Mac computer. Apple has only designed its development tools to work on Apple's operating system, OS X, and it's also recommended that you run the latest version.

For those of you who aren't using a Mac, it's worth considering purchasing one if you plan to do iOS development. If you just need to build mobile apps, you'll be able to take advantage of any of the Mac computers. Any new Mac will have enough processing power to manage the simulation and build process. If you consider purchasing a used machine, you should verify that it's able run the latest version of OS X.

If you don't have a Mac, there are some options that can help build your apps. Ionic is building a service that will allow you to build mobile apps for any supported platform even if you don't have a Mac.

The Apple Developer Program has two types of membership: iOS and OS X development. You'll need to sign up at http://developer.apple.com and join the iOS program. It costs US $99 per year, but you only need to sign up when you're ready to sign and deploy your app to the App Store. You can download Xcode and work through

this entire book without an account until the point when I show you how to deploy an app to the App Store.

### 1.6.2 Google Android

Google created Android as an open source mobile platform, and has allowed mobile device makers to integrate Android into their devices. Compared to Apple's approach, Android has a very diverse set of devices because Google doesn't control every device Android is installed on. Older devices may also have Android forks specifically designed for a mobile carrier. This open system has encouraged adoption and also has been the leading platform in emerging markets by allowing lower-cost devices due to the absence of licensing fees for the operating system.

Android provides a number of tools for developing that are freely available for download from Android's site, http://developer.android.com/. Google has also been working on additional tools that are being built into Chrome, Google's browser, to provide useful development support for hybrid app developers. We'll cover how to set up your computer for Android development in the next chapter. The Android SDK has a simulator that can emulate the screen size and resolution of most Android devices.

Android development is supported on Mac, Linux, and Windows computers. You can review the exact requirements for Android developer tools at https://developer.android.com/sdk/index.html.

Google also has a Developer Program, which has a one-time fee of US $25. Just like with iOS, you don't have to sign up until you're ready to publish your app into the Play Store. You can register at https://play.google.com/apps/publish/signup/.

There are a few other Android app stores, notably the Amazon Web Store, which may also charge for a developer program. These aren't covered in this book. But you'll be able to build and deploy apps for any Android-based device, even if the app is distributed through a different store.

## 1.7 Summary

Throughout this chapter we've looked at details about how Ionic provides a powerful set of tools for building hybrid apps. Let's review the major topics covered in this chapter:

- Ionic is a solid choice that benefits developers, managers, and users.
- Hybrid apps are an advantage for developers who are already familiar with the web platform, and don't require learning additional programming languages.
- Hybrid apps use a WebView inside of a native app to run web applications that have access to native APIs.
- Ionic is designed to work with Angular for web application development and Cordova for integration with the device platform.
- Android and iOS are supported and require developer subscriptions. iOS development tools require a Mac.

In the next chapter, we'll review how to set up your computer to develop Ionic apps and set up a simple app to get started.