Hybrid Apps are a good choice in mobile app development in order to achieve cost-effectiveness and rapid development. However, they were not preferred to native apps until few years back due to poor performance and bad user experiences, but everything has changed with the release of Ionic.

Getting Started with Ionic is a fast-paced, practical book which explains all the important concepts of AngularJS and the Cordova framework that are required to develop apps, and then gives you a brief introduction to Hybrid Mobile Applications. It will guide you through setting up the environment for developing mobile apps, and through the multiple options and features available in Ionic so you can use them in your mobile apps. Features such as the side menu, tabs, touch interactions, and native features such as barcode, camera, and geolocations are all covered. Finally, we’ll show you how to use Cordova plugins and publish your apps.

Who this book is written for
This book is ideal for any web developer who wants to enter the world of mobile app development but has no clue where to start. This book will also be useful for Hybrid App developers who have not yet found the perfect framework that ensures users get a rich user experience from their apps.

What you will learn from this book
- Get to know about Hybrid Apps and AngularJS
- Set up a development environment for building Hybrid Apps
- Navigate around the components and routing in Ionic
- Use different Ionic directives for a mobile-specific experience
- Integrate an Ionic app with backend web services
- Work with plugins to include native functionality in your Hybrid Apps
- Test your apps on real devices
- Build an e-commerce app for iOS and Android from scratch

Community Experience Distilled
Get up and running with developing effective Hybrid Mobile Apps with Ionic

Foreword by Mike Harlington, Developer Advocate at Ionic
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'All About Hybrid Mobile Apps and Ionic Framework'
- A synopsis of the book’s content
- More information on Getting Started with Ionic
About the Author

Rahat Khanna is a techno-nerd experienced in developing web and mobile apps for many international MNCs and start-ups. He has completed his Bachelors in Technology with Computer Science & Engineering as specialisation. During the past 7 years, he has worked for a multinational IT service company and ran his own entrepreneurial venture also in his early twenties. He has worked on ranging projects from static HTML websites to scalable web applications and engaging mobile apps. Along with his current job as a Senior UI developer at Flipkart, a billion dollar e-commerce firm, he now blogs on the latest technology frameworks on sites www.airpair.com, appsonmob.com, and so on and delivers talks at community events. He has been helping individual developers and startups in their Ionic projects to deliver amazing mobile apps.
Preface

Hybrid Mobile Apps have become a promising choice in mobile app development to achieve cost effectiveness and rapid development. Ionic has been instrumental in setting the benchmark in this space as it focuses on performance. It has evolved as the most popular choice for Hybrid Mobile App development as it tends to match the native experience and provides robust components/tools to build apps.

*Getting Started with Ionic* equips any web developer with the practical knowledge required to use modern web technologies in building amazing Hybrid Mobile Apps using Ionic. This fast-paced, practical book explains all the important concepts of AngularJS and Cordova framework required to develop Ionic apps. Then, it gives you a brief introduction to Hybrid Mobile Applications. It will guide you through setting up the development environment for different mobile platforms and through the multiple options and features available in Ionic, so you can use them in your mobile apps. Features, such as the side menu, tabs, touch interactions, and native features, such as bar code, camera, and geolocations, are all covered. Finally, we'll show you how to use Cordova plugins and use Ionic cloud services to empower your mobile apps.

**What this book covers**

*Chapter 1, All About Hybrid Mobile Apps and Ionic Framework*, covers the introduction to Hybrid Mobile Applications and the technologies used to develop these apps. This chapter will also include all important concepts regarding AngularJS and Apache Cordova, which readers need to know before building Ionic apps.

*Chapter 2, Setting up the Environment the Right Way*, covers setting up the native development environments for required platforms and then installing Ionic and required dependencies to start building Ionic apps. It will include details about Ionic command-line interface and the important features it provides for rapid development.
Chapter 3, *Start Building Your First Ionic App*, includes instructions for starting a new project and leveraging Ionic starter templates to bootstrap easily. It also covers the explanation about the design of the app structure and how you can plan to develop your app on top of it.

Chapter 4, *Navigation and Routing in an Ionic App*, covers how to create routes and layouts for navigation. It will teach the reader about views, Ionic header/footer sections, and different layout components, such as tabs, side menu, and modals. It also includes the basic routes and navigation setup for our sample app.

Chapter 5, *Accessorizing Your App with Ionic Components*, covers different components, which will be used in building the mobile app. It has two major categories: one is CSS components, which include grid framework and some reusable CSS classes to be used to design UI. The other category is Javascript components, which will facilitate building the business logic and UI interactions for the app.

Chapter 6, *Integrating App with Backend Services*, includes explanation about creating Ionic services and factories, which will interact with backend services. Ionic Apps can be integrated with all kinds of web services, REST APIs, SOAP services, and even cloud-based mBaaS (*Mobile Backend as a Services*) such as Parse and Firebase.

Chapter 7, *Testing App on Real Devices*, covers tools and techniques used to test and debug the app on your actual devices.

Chapter 8, *Working with Cordova Plugins - ngCordova*, teaches the reader how to use ready-made Cordova plugins, which help the mobile app to talk to native device APIs, such as sensors, camera, and geolocation. It will also explain how to integrate an open source Angular wrapper named ngCordova for cordova plugins into an Ionic App.

Chapter 9, *Future of Ionic*, includes an introduction to various cloud Ionic services available online under the Ionic.io platform. It also covers about the future of Ionic v2 along with Angular v2 talking about new features and migration path.
All About Hybrid Mobile Apps and Ionic Framework

In this chapter, we are going to learn what a Hybrid Mobile Application is and the current technology ecosystems supporting it. We will also be introduced to Ionic Framework and the reasons that should lead you to decide on Ionic as your preferred choice. The topics covered in this chapter will be as follows:

• Introduction to a Hybrid Mobile Application
• Using web technologies to develop for mobile devices
• What is AngularJS?
• Why use Apache Cordova?
• Introducing Ionic Framework

The term 'website' has become a word of the past. 'App' is the new buzzword, and the world is moving away from old software systems to new jazzy apps. App, or application in terms of a software, is a more sophisticated system, which involves enabling a lot more features to the user rather than just providing static information like a traditional website.

Web apps have a lot of limitations such as requiring Internet connectivity all the time and restrictions on fully utilizing the hardware capabilities of the device on which you are accessing them. Mobile apps, on the other hand, defy all of these limitations and provide an engaging user experience.
Mobiles have emerged as the most popular channel for user engagement. The number of smartphone users is expected to grow to nearly 2.16 billion in 2016 (http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694), which is more than one quarter of the global population. The growth rate of smartphone usage has been tremendous and is expected to grow day by day.

Mobile adoption is associated with some of the following trivial points:

• Mobiles are the only device that we keep with us the whole day
• Kids (between the ages of two and five) know more about using a smartphone than tying their shoelaces
• Time spent on mobile phones is increasing 14 times faster than time spent on desktop media
• People check their smartphones first thing when they wake up instead of wishing good morning to their partners or family

After reading through these points, one must think that the perfect mobile strategy has become a necessity for every business. Going mobile is not a choice any more, rather everyone has to decide which way to go: mobile-first, mobile-only, or mobile-after. If you are a new start-up, an existing enterprise, or an individual developer, Native Mobile apps have a steep learning curve and high development costs for covering all platforms. Mobile ecosystems have become fragmented with multiple OSs such as iOS, Android, Windows, and numerous OEMs such as Samsung, LG, HTC, and others. A Hybrid Mobile Application, about which you will learn in this chapter, is the perfect savior for you. Ionic Framework is a popular hybrid app development framework that helps us in creating Native-looking Apps for multiple platforms using a single codebase.

**Introducing a Hybrid Mobile Application**

A common misconception is that a Hybrid Mobile Application cannot be installed on the device, but that is wrong. A Hybrid Mobile App is like any other Native Mobile App, which can be installed on devices and published using App stores. They can access the device hardware such as camera, accelerometer, GPS, and so on.
As we have discussed, there are multiple mobile platforms such as iOS, Android, Windows, plus many new ones such as Firefox OS and Tizen that have emerged lately. The development environment and programming languages are different for each of these. We have to code using Objective-C for iOS apps, Java (Android SDK) for Android apps, C#/VB.net with XAML for Windows Phone apps. If any entity requires its mobile presence across all these platforms, multiple teams and different codebases need to be maintained, which is too cumbersome.

Hybrid Mobile Apps can be developed for multiple platforms using a single codebase. However, some specialized code needs to be written for each platform to harness the native APIs for it.

Types of Hybrid Mobile Apps
There are broadly two categories of Hybrid Mobile Apps in the industry:

- WebView-based Hybrid Apps
- Cross-compiled Hybrid Apps

WebView-based Hybrid Apps
Each native mobile platform has a common control/component called a WebView, which is nothing but a Chromeless browser. This component is utilized to open locally hosted web content, for example, HTML pages, CSS files, and JavaScript code. HTML5 and CSS3 provide capabilities to develop responsive apps, which can render nicely on multiple screen sizes. The web technologies have evolved to handle the touch interactions that make them a perfect candidate for developing solutions for smartphones and tablets. The example of development platform and frameworks using this approach are Cordova, Ionic Framework, KendoUI Mobile, F7, Mobile Angular UI, Onsen UI, and many more.

Cross-compiled Hybrid Apps
Another category of hybrid app involves cross-compiling multiple native apps from a single programming language. For example, the developer will code using a single language, say, A, which can be converted at compile time or run time into native language components. Generally, these types of frameworks and platforms leverage creating a bridge or a mapping of native components to their custom constructs in the programming language intended for development. Examples for this category are Xamarin (C#), Kony (JS), Corona (C Lang.), Qt(C++), and many more.
We will be talking about WebView-based Hybrid Apps in this book as they are more suitable for large-scale and complex Mobile Apps. In this next section, we will learn more about the anatomy of Hybrid Apps and how they are capable of developing Native Apps for multiple platforms.

**Anatomy of a Hybrid Mobile App**

Hybrid Apps are no different from Native Mobile Apps that are installed on any mobile platform such as Android or iOS. On any platform, the core device APIs for hardware such as GPS, Camera, Accelerometer, and so on will be exposed by the Mobile OS. The following diagram shows the anatomy of Hybrid Mobile App:

![Diagram showing the anatomy of a Hybrid Mobile App]

These APIs are consumed by the native code of your Hybrid Apps. All the components of Hybrid Apps are discussed in detail in the following section.
Custom WebView

Each native platform development kit has a component called WebView, which is nothing but a Chromeless browser control. WebView has the capability to open local or remote web content, which is exploited by Hybrid App frameworks to display the UI of the app using web technologies. This is the most important component of a Hybrid App and has a significant role in deciding the performance of the app.

WebViews in popular platforms such as iOS and Android used to have a different rendering engine and JavaScript engine than the latest browser (Chrome or Safari). Last year, Apple released a new control in its SDK called WKWebView, which uses all the performance optimization such as the Nitro JavaScript engine used by the Safari browser on iOS. On similar lines, Google also released an updated WebView, which uses the rendering engine and JavaScript runtime of Chromium (Chrome browser). Google has also launched a new feature called Updatable WebView from Android (5.0) Lollipop, which enables you to upgrade only the WebView of your Hybrid App.

Crosswalk is an interesting open source project that enables app developers to embed custom WebView into your Hybrid App. With these advancements, Hybrid Apps have become capable of using the latest web features such as WebRTC for real-time multimedia communication, and WebGL for advanced graphics rendering.

Native library

All the Hybrid App development frameworks based on WebView have their own native library. It comprises some basic utility functions that support the Hybrid App such as creating app configs, bootstrap code for Native Apps, customization for the WebView, common error/exception handling logic, and so on. The native library is specific to the mobile platform as it involves interacting with the core OS APIs and components.

The most popular framework, for example, Apache Cordova/Phonegap has an architecture of dividing its native library into a core section and pluggable components called plugins. This helps in reducing the bare minimum size of a Hybrid App. Developers can use only the plugins they require for a specific app. A plugin will include native code for a particular feature and a JavaScript interface exposing the native functionality. For example, if you want to use the Fingerprint Authentication API for iOS, you can just include the plugin for iOS and use it apart from the core. It also enables communities to contribute by developing open source plugins.
Native to JS Bridge
In a WebView-based Hybrid App, the UI is always written using web technologies, and JavaScript is the language for writing logic and hence we need to call our native code from JS and get results to JS also. A bridge has two functions, one is to enable JS to call any native method and the other is to allow native methods to execute callbacks in JS. The bridge comprises different implementations in different platforms to call JS from native. For example, in Android, Java objects are marshalled into the WebView and can be called from the JS. In iOS, JS calls a specific URL scheme, which is interpreted by the native code. The reverse bridge is a simple global JS function that is called by the WebView passing special arguments such as callback results or specific commands.

Using web technologies to develop for mobile
After understanding what is happening inside a Hybrid App, it is important to know how web technologies are used to develop Mobile Apps. We can use simple HTML5, CSS, and JS to create mobile-specific UIs and enable them to be viewed in the WebView discussed previously. But any website, even if it is a mobile web app, should not be directly packaged into a Hybrid App. This is the most popular mistake developers make and then complain about the performance of the Hybrid App.

A Hybrid Mobile App UI needs to have proper separation of concerns and can be best developed using single page architecture (SPA) or MV* architecture. It helps in providing a seamless user experience and provide a Native App such as engagement. It also equips developers with segregated areas to code, for example, writing views using HTML5 markup templates, styling using CSS, and logic in JS.

Ionic Framework uses an open source MV* framework called AngularJS to build robust Native-looking Hybrid Mobile Apps. AngularJS is an extensive topic that cannot be covered here, but we will learn about some basics that are essential in utilizing Ionic Framework to its full potential.

What is AngularJS?
AngularJS is a JavaScript-based MV* framework that provides a strong backbone to scalable and complex web apps. It also enables developers to extend HTML and program their apps in a declarative paradigm in lieu of an imperative programming style. AngularJS provides us with a way of creating reusable components, setting standard templates in HTML, and reusable business logic with the ability to bind data dynamically to it.
AngularJS is a perfect fit for creating rich Mobile UI Apps as it provides a robust structure to the frontend, which is a reason why the Ionic team has chosen it as their core.

**Important concepts in AngularJS**

In order to build apps using AngularJS we need to understand the core concepts used in AngularJS and learn how to use them. The prerequisite for learning AngularJS is decent knowledge of HTML, CSS, and JavaScript. The core concepts that will be discussed include modules, directives, controllers, expressions, and filters.

**Modules**

In AngularJS, modules are at the core of everything because an AngularJS App is defined as a module itself. A module is a container for different sections of the app such as controllers, directives, services, and so on. A module can have other module dependencies injected at run time. This feature of AngularJS is called **DI (Dependency Injection)**. It provides super flexibility for unit testing as dependencies can be mocked and injected.

Each module has two important lifecycle hooks implemented as methods registering callbacks. The methods are *config* and *run*. The *config* method is used to set up or provide important configuration settings such as routes, states, and so on, whereas the *run* function is used like the *main* method for initiating the module inside the callback registered:
Directives

Directives are the most important and yet the most complex part of AngularJS. They can be easily described as a set of markers on DOM elements such as element name, CSS class, an attribute, or a comment, which lets the AngularJS compiler know that specified behavior needs to be attached there. It is advised to encapsulate any DOM manipulation logic into a directive while developing an AngularJS App.

There are plenty of in-built core directives that are part of the ng-module and used in each angular app. We will discuss the essential ones in order to understand the functioning of a directive.

ng-app is the core directive that bootstraps our app. The root angular module name needs to be passed to this directive and is generally used as an attribute, for example:

```html
<html ng-app="my-app">
```

ng-model is a directive used for binding models from controllers to the views. We will learn about the scope in the text ahead, which is used to hold the models as Plain Old JavaScript Objects. ng-model is used with input, select, and text area controls, for example:

```html
<input type='text' name='textField' ng-model="my-var">
```

There are many other directives such as ng-class, ng-show, ng-hide, and so on, which you will use while developing your app. Ionic Framework has built most of its components as custom directives and will be used frequently to develop Hybrid Apps using the framework.

Controllers

In AngularJS, a controller is a constructor function used to augment the view models. The controller can be initiated in two ways, either using the directive ng-controller or it can be associated with a route/state. According to Angular docs, controllers should be used for setting up the initial state of the $scope (view model) object and adding behavior to it.

We should refrain from using it for the following logic:

- DOM manipulations
- Formatting input
- Input validations
- Sharing of data should be done using services
The example code for a basic controller is:

```javascript
var newApp = angular.module('NewApp',[]);
newApp.controller('FirstController',['$scope',function($scope) {
    $scope.modelObj = { name:'MyDummyObject' };
    $scope.updateName = function(newName) {
        $scope.modelObj.name = newName;
    };
}]);
```

We define the dependency injection twice, first as a string and then as an argument to avoid problems during the minification of your JS files. Read more details at https://docs.angularjs.org/guide/di.

### Services

In AngularJS, services are used to store business logic and organize your code into logical units or entities. Angular services are lazily loaded and are wired together using Dependency Injection, as discussed earlier. AngularJS services are singletons and thus instantiated only the first time they are encountered as a dependency to any controller or module.

AngularJS has multiple built-in services out of which $http is the most important one. The $http service provides a wrapper on the browser's XMLHttpRequest object, or more popularly known as Ajax requests.

We can create AngularJS services using two module factory methods. The methods used are .service or .factory. The former is used when we create the service instance using a constructor function, and the latter is used when a factory function returns the instance of the service. Conceptually, we should use .service when we are integrating to any external API and use .factory if we are creating objects representing app models.

The code sample to create services using both methods is given as follows:

```javascript
// Using Factory Method
var newApp = angular.module('NewApp',[]);
newApp.factory('MyService',[function() {
    var serviceInstance = {};
    var privateMethod = function() { return 'result'; };
    serviceInstance.exposedMethod = function() {
        return privateMethod();
    };
    return serviceInstance;
}]);
```
In the given code, a factory function available on the `angular.module` object is used to return a service instance that contains public methods. The service can encapsulate private methods to include logic that should not be exposed:

```javascript
// Using Service Method
var newApp = angular.module('NewApp',[]);
newApp.service('MyService',[function() {
    var privateKey = function() { return 'result'; };
    this.exposedMethod = function() {
        return privateKey();
    };
}]);
```

Services in Angular can be used in the following ways:

- Representing business entities/models
- Sharing data across controllers
- Interface to external web service calls or Ajax requests

## Templates

In AngularJS, templates are associated with a route/state to display HTML elements and Angular-specific elements. Template code can be directly passed in JS or TemplateURL, for example, URL to the template file can be passed to any object. Angular combines the template to the controller and models to display dynamic content to the user on the browser. `$scope` is used to bind the controller to the template.

Templates can use Angular directives and expressions that are compiled by the `$compile` service. The following code shows a simple Angular template:

```html
<html ng-app>
  <head>
    <title>My First Angular Template</title>
  </head>
  <body>
    <h1>Main Section</h1>
    <div ng-controller='MyCtrl'>
      <p>{{ contentStr }}</p>
      <p>Date : {{ dateStr | dateFormat }}</p>
    </div>
    Name: <input type='text' ng-model='name'>
  </body>
</html>
```
Expressions

Expressions are code snippets put in AngularJS templates to create bindings between controllers and templates. The expressions are, by default, represented by `{{ }}` in the templates. Angular expressions are different from JavaScript expressions as they have some restrictions. They can contain basic arithmetic or concatenation operations, but no control flow statement or function declarations. Angular expressions run against the context of the scope, for example, variables in the bindings are evaluated on the scope object for each respective controller.

Filters

Filters are used in expressions to format data before displaying it. Filters are applied using the `|` operator. There are in-built filters such as currency, number, date, lowercase, and so on.

Example of expressions with a filter in a template:

```
<p>Total: {{ amount * 32 | currency }}</p>
```

amount is a scope model, using the `*` arithmetic operator and the `currency` filter.

Why use Apache Cordova?

Apache Cordova is a WebView-based Hybrid App development framework used to build cross-platform Native Apps. It is one of the most popular frameworks that has been open sourced by Adobe and is maintained by the Apache Foundation. Adobe maintains another branch with added features named Phonegap. They also have a cloud-based service called Phonegap Build (http://build.phonegap.com), which generates the native builds on the fly so that you do not need to install native SDKs.

It follows the same architecture discussed previously, having a minimal core and ability to add plugins for extra functionality to your app. Apache Cordova has a high number of open source plugins that provide excellent capability to Hybrid Apps. Any developer can also create a custom plugin to expose unique native functionality in a Hybrid App. It also consists of a CLI interface to provide commands for managing plugins and automating builds for multiple platforms. Apache Cordova is a widely tested and accepted framework and is recommended for building Hybrid Mobile Apps from web content.
Introducing Ionic Framework

Ionic Framework is a Hybrid App development framework that enables developers to build Native-looking Mobile Apps using web technologies (HTML5, CSS3, and JS). Ionic Framework is completely open source so that developers can build and publish their apps to the marketplace without any cost.

Ionic is built on top of the AngularJS framework and uses Apache Cordova for building apps from web content. Ionic Framework includes a set of amazing Angular directives that makes it very easy to develop for mobile. For example, ListView, Optimized Touch gestures, Side Menus, Popup, Tabs, and mobile-specific input elements.

Ionic has ready-made UI for mobile components, which helps in rapid application development for Hybrid Mobile Apps. Ionic has native-looking stylesheets for Android and iOS, which automatically get applied based on the platform build.

Ionic Framework has evolved into an ecosystem with a suite of mobile development tools along with the framework itself. Ionic CLI has amazing options such as Ionic Lab and Live Reload, which helps developers save lot of development time. Ionic view is a Native App for iOS and Android where developers can deploy and test their apps on the fly without packaging. Ionic.io is a complete cloud-based backend service platform where developers can manage their app data, view analytics, and manage push notifications from a single console.

Summary

We have learnt all about Hybrid Mobile Applications and how easy it is for a web developer to start building Mobile Apps using Ionic Framework. The concepts mentioned in this chapter will suffice in building apps using AngularJS, Cordova, and Ionic. In the coming chapters, we will build amazing Hybrid Mobile Apps that can be deployed to public app stores. In the next chapter we will learn about setting up the development environment and starting an initial project to bootstrap our development journey.
Where to buy this book

You can buy Getting Started with Ionic from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.