MOBILE PROGRAMMING SERIES

APACHE CORDOVA API COOKBOOK

JOHN M. WARGO

Foreword by BRIAN LEROUX

Apache Cordova API Cookbook

This page intentionally left blank

Apache Cordova API Cookbook

John M. Wargo

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data
Wargo, John M.
Apache Cordova API cookbook / John M. Wargo.
pages cm
Includes index.
ISBN 978-0-321-99480-6 (pbk. : alk. paper)
1. Mobile computing—Computer programs. 2. Application program interfaces (Computer software)
3. Apache Cordova. 4. JavaScript (Computer program language) 5. Smartphones—Programming.
6. Application software—Development. I. Title.
QA76.59.W3685 2015
005.2’762–dc23
2014017378

ISBN-13: 978-0-321-99480-6
ISBN-10: 0-321-99480-9

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.
First printing, July 2014

To my wife, Anna;
crazy about you!

This page intentionally left blank

Contents

This page intentionally left blank

Foreword

In the late summer of 2011 I first received news that Nitobi Software was being acquired by Adobe Systems to continue our work on the fast-growing, open-source PhoneGap project. The future was bright, with a happy and growing developer community and a mission bigger than ourselves making it possible to create native mobile apps using HTML, CSS, and JavaScript. To ensure the project stayed true to our open source roots we, with Adobe, donated the source code to the Apache Software Foundation. After some initial thrashing, the project formerly known as *PhoneGap* became *Apache Cordova*.

Apache Cordova thrives today. At the time of this writing, Apache Cordova was installed roughly 100,000 times in the last 30 days. It has a rather large ecosystem of code, with more than 50 repositories hosted by Apache and an even larger developer community with more than 200 native plugins on the official registry. All this size does come with some complexity, and this book will help you navigate that.

In principle, Apache only recognizes individual contributors to a project. In practice, many organizations sponsor individuals to collaborate. Adobe employees are joined by Google, Microsoft, Mozilla, BlackBerry, LG, Intel, IBM, and SAP in this mildly bizarre, neutral ground of collaboration made possible by Apache. Organizations choose to collaborate and contribute for a variety of reasons. Sometimes it is to create downstream distributions such as Adobe PhoneGap or just a set of Cordova plugins like what is found in the SAP Mobile Platform. However, at Apache only individuals can participate as contributors. This book will help you understand how everything works so you can consider the opportunity of contributing back to a large, open-source effort. In any case, by choosing to work with Apache Cordova you are investing upstream, meaning that your skills investment will be applicable to all the downstream distributions aforementioned. This is a subtle benefit of Apache Cordova's open-source design.

John Wargo is one of the individuals contributing to Apache Cordova. He's been tireless, keeping up with our dev mailing list that pushes over a thousand messages a month. (Which is nothing compared to our developer community mailing list!) He has meticulously reviewed our documentation and helped clarify countless parts of the API surface with the devs and the dev community. He is a stand-up example of a hacker making things better for all of us.

Cordova has grown beyond a simple toolkit for compiling web bits into native bits. The code has been completely refactored into a "Swiss army knife" for managing applications that target embedded web views. Understanding the structure and implementation of Cordova-based apps will make you a better developer, period. The modern developer needs to understand native platforms and the web platform. Apache Cordova unifies these concepts without hiding the underlying operating systems we work with. You will be imbued with superpowers to manage the complexity of moving between Android, iOS, and the browser. You will understand how native interfaces can be created from the humble web view. You will have the tools to participate with agency on any operating system with any web technology stack you choose.

We have always wanted to give open web standards a fighting chance against native operating systems. The original goal for the source code now known as Apache Cordova was to cease to exist. This was not a nihilistic statement but an acknowledgment that all technology deprecates. Our goal is to provide an alternative to proprietary client treadmills using HTML, CSS, and JavaScript as our vehicles. Today, I think these lines are sufficiently blurry. There is no web versus native; neither won. The future is somewhere in between. Sometimes people call this "hybrid." Hybrid is really just another way of saying *Apache Cordova*.

Have fun hacking, and if these principles seem right to you, consider joining the developer mailing list and introducing yourself. The Apache Cordova community is very friendly and always welcomes fellow mobile web hackers.

—Brian LeRoux

Preface

This is a book about the Apache Cordova APIs. Apache Cordova is a very popular open-source framework for building cross-platform native mobile applications using HTML5. Developers code their application content (UI and application logic) using HTML, CSS, and JavaScript, then that content is packed into native applications targeting multiple popular (and some not-so-popular) mobile device platforms.

Web applications running on a mobile device don't typically have access to device-side capabilities such as the camera, address book, compass, and so on. While there are initiatives within the Internet community to add these capabilities to the mobile browser, they are not implemented consistently across mobile device platforms today. The Cordova APIs described in this book provide an interface a developer can use to access those device-side capabilities today, as device manufacturers add those capabilities to their browsers. This book teaches you how to use those APIs in your Cordova applications.

This book is for mobile developers who have at least some experience with web development and Apache Cordova. If you're new to mobile development, note that a lot of the general-purpose mobile development background information you will need to understand the topics in this book won't be found here.

If you've not yet worked with Apache Cordova, this book isn't going to help you set up a Cordova development environment, understand the ins and outs of the Cordova development process and the Cordova CLI, or use the mobile device platform tools to build and test your applications. You'll likely want to spend some time with this book's companion, *Apache Cordova 3 Programming* (or its successors), before digging in here.

Inside the Book

What you'll find herein is complete coverage of each Apache Cordova API. For each API, I describe what it does, how it behaves, and how to use it in your applications (with code). Each chapter includes at least one complete example application you can use that exercises every aspect of each API covered in the chapter. There are more than 30 complete applications described in the book with source code available on GitHub (see the "Resources" section for the exact location).

The example applications highlighted in the book are built using either Adobe Topcoat (topcoat.io) or jQuery Mobile (jquerymobile.com). I did this to give the applications a more professional look. It also allowed me to let those frameworks take care of the applications' user interface and user interaction activities so the Cordova-related code could be as clear and distinct as possible.

What You Won’t Find Here

Well, as with all of my other books, you won’t find any pop culture references anywhere in the book. The chapter on the Contacts API does include the names of members of the Monty Python comedy troupe as sample contact names for the example application, but if you don’t know the Pythons, you likely wouldn’t even notice this.

The book does not include any content in languages other than English, HTML, and JavaScript. I’m assuming you’re OK with English. As this is a software development book I’m assuming you will also be OK with HTML and JavaScript.

As this is a book about the Apache Cordova APIs, you won’t find any discussion of web development or mobile development topics. Pearson has some excellent books on those topics. Visit InformIT.com if you are interested.

Resources

I’ve created a web site for the book; it’s located at www.cordovacookbook.com. On the site you will find information about the book, and as readers let me know of any omissions or errors in the text, I’ll post the information to the site’s errata area.

The book’s example application source code can be obtained from the book’s GitHub repository at https://github.com/johnwargo/apache-cordova-api-cookbook-code. I will update the code there as bugs are reported and fixed.

You can find my personal tech blog at www.johnwargo.com. On this site I publish articles on topics that interest me. Most often, I write about mobile development topics and will post updates on Cordova as they come up.

Acknowledgments

Many people helped with the creation of this book; I would like to thank:

Brian LeRoux and the Cordova dev team for making such a great product and for patiently answering my questions as they came up while I wrote this book.

My colleagues at SAP for continuing to teach me new things about Apache Cordova.

Ashwin Desai for doing such an excellent job on the technical review of the manuscript; he even corrected my source code comments.

Greg Doench, Chris Zahn, Michelle Housley, and the rest of the team at Addison-Wesley for helping me create this book.

About the Author

John M. Wargo has been a professional software developer for the entirety of his professional career. He got into the mobile space when he accepted a job at Research In Motion (now called BlackBerry) and became a subject-matter expert on BlackBerry development for a US carrier and its customers.

Using his experience at RIM he wrote the first book on BlackBerry development (*BlackBerry Development Fundamentals*) and from there he was hooked. He is the author of the bestselling *PhoneGap Essentials* and *Apache Cordova 3 Programming*. He also penned the majority of the articles on mobile development for *Mastering Mobile for Notes/Domino*, an anthology of articles from *The View*, a magazine for IBM Lotus Domino developers.

John is currently a product manager for SAP, working with the SAP Mobile Platform. He is the product manager for Kapsel, a set of enterprise plugins for Apache Cordova, and the SAP Fiori Client (a mobile application built using Apache Cordova), available in the Google Play Store and the Apple App Store.

In his spare time he stays caught up on mobile development trends and tools and thinking about his next book.

This page intentionally left blank

1 Introduction to Apache Cordova

This chapter is your introduction to the Apache Cordova framework and Apache Cordova application development. In the chapter, I describe what Cordova is, how it works, and how to develop applications using Cordova. It's clear from many of the support forum posts that developers who are just getting started with Apache Cordova don't really "get" what they're working with. This chapter should answer many of the initial questions you have related to Apache Cordova. If you are already familiar with Apache Cordova, you can skip this chapter if you want and jump right away into the Cordova application programming interfaces (APIs).

Introduction to Apache Cordova

Apache Cordova (http://cordova.apache.org/) is a free, open-source framework for building cross-platform native applications using HTML5. The creators of Apache Cordova wanted a simpler way of building cross-platform mobile applications and decided to implement it as a combination of native and web application technologies. This type of mobile application is called a Hybrid application.

The initial benefit of Apache Cordova is the native capabilities above and beyond what is normally supported in the mobile browser. At the time all of this started, the best way to build a mobile application that worked on multiple mobile devices was to build it using HTML. Unfortunately, though, for mobile developers, many mobile applications needed to do more than HTML and web browsers could support. Building a web application that interacted with the device camera or the local Contacts application simply wasn't possible. To get around this, Cordova implements a suite of APIs that extend native device capabilities (such as the camera, accelerometer, Contacts application, and so on) to a web application running within the native container. The rest of the book beyond this introductory chapter is all about those APIs.

Apache Cordova consists of the following components:

- Source code for a native application container for each of the supported mobile device platforms. The container renders the Cordova web application on the device.

- A set of Core APIs (delivered as plugins) that provide a web application running within the container access to native device capabilities (and APIs) not normally supported by a mobile web browser.
- A set of tools used to manage the process of creating application projects, managing plugin lifecycle, building (using native software development kits—SDKs) native applications, and testing applications on mobile device simulators and emulators.

To build a Cordova application, you create a web application, package the web application into the native container, test and debug the application, and then distribute it to users (typically through an app store). The packaging process is illustrated in Figure 1.1.

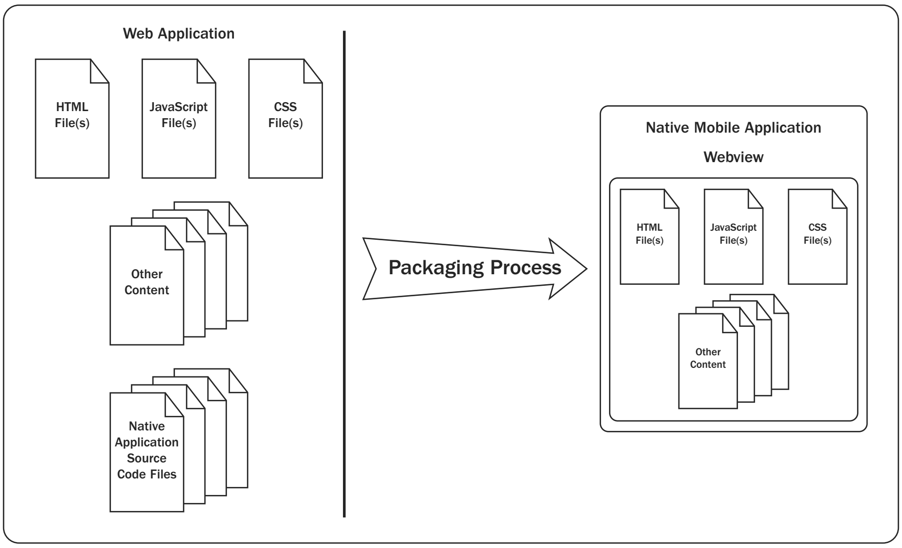


Figure 1.1 Apache Cordova Application Packaging Process

> **Note**
>
> When many developers first learn about this technology, they immediately assume that the web application is somehow translated into the native language for each supported mobile device platform—converted into Objective-C for iOS or Java for Android, for example—but that's not what's happening here. There are some mobile application frameworks that take that approach, but for Cordova, the web application simply runs unmodified within a native application shell.

Within the native Cordova application, the application's user interface (UI) consists of a single screen that contains nothing but a single web view that consumes the available screen space on the device. When the application launches, it loads the web application's start-up page (typically index.html but easily changed by the developer to something else) into the web view, then passes control to the web view to allow the user to interact with the web application. As the user interacts with the application's content (the web application), links or JavaScript code within the application can load other content from within the resource files packaged with the application or can reach out to the network and pull content down from a web or application server.

About Web Views

A web view is a native application component that is used to render web content (typically HTML pages) within a native application window or screen. It's essentially a programmatically accessible wrapper around the built-in web browser included with the mobile device.

The web application running within the container is just like any other web application that would run within a mobile web browser. It can open other HTML pages (either locally or from a web server sitting somewhere on the network), and JavaScript embedded within the application's source files implements needed application logic, hiding or unhiding content as needed within a page, playing media files, opening new pages, performing calculations, retrieving content from or sending content to a server. The application's look-and-feel is determined by any font settings, lines, spacing, coloring, or shading attributes added directly to HTML elements or implemented through Cascading Style Sheets (CSS). Most anything a developer can do in a web application hosted on a server can also be done within a Cordova application.

A typical mobile web browser application does not usually have access to device-side applications, hardware, and native APIs. For example, the Contacts application is not accessible to web applications, nor can a web application typically interact with the accelerometer, camera, microphone, and more or determine the status of the device's network connection. The typical native mobile application, on the other hand, will make frequent use of those capabilities. An interesting mobile application (interesting to prospective application users anyway) likely needs access to those native device capabilities.

Cordova accommodates that need by providing a suite of JavaScript APIs that a developer can leverage to enable a web application running within the Cordova container to access device capabilities outside of the web context. Essentially these APIs are implemented in two parts: a JavaScript library that exposes the native capabilities to the web application, and the corresponding native code running in the container that implements the native part of the API. This is implemented essentially as one JavaScript library but with separate native implementations on each supported mobile device platform.

Beginning with Cordova 3.0, each of the Cordova APIs has been broken out into separate plugins; you can use the Cordova command-line interface (CLI) or plugin manager (plugman) to add and remove plugins from your Cordova project. This approach provides the architecture illustrated in Figure 1.2, an application with discrete code for each plugin and where only the needed plugins are packaged with the application.

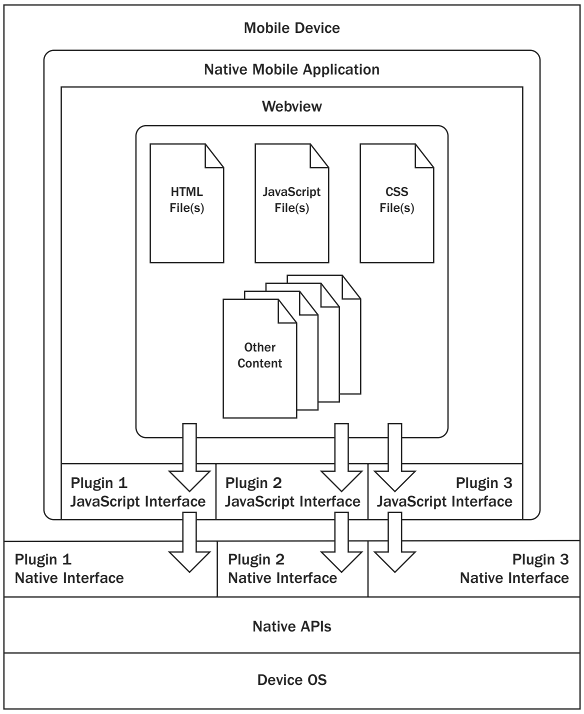

Mobile Device
Native Mobile Application
Webview
HTML File(s)
JavaScript File(s)
CSS File(s)
Other Content
Plugin 1 JavaScript Interface
Plugin 2 JavaScript Interface
Plugin 3 JavaScript Interface
Plugin 1 Native Interface
Plugin 2 Native Interface
Plugin 3 Native Interface
Native APIs
Device OS

Figure 1.2 Apache Cordova Native Application Architecture Post-3.0

Cordova currently provides the following APIs:

- Accelerometer
- Camera
- Capture
- Compass
- Connection
- Contacts
- Device
- Events
- File
- Geolocation
- Globalization
- InAppBrowser

- Media
- Notification
- Splashscreen

Each of these APIs is described in detail in Chapters 2 through 16. At least one complete sample application is provided for each.

Supported Platforms

As of this writing, the Apache Cordova web site lists that it supports Google Android, Samsung bada, BlackBerry, Apple iOS, Palm WebOS, Symbian, and Microsoft Windows Phone platforms. The Cordova download contains folders for Android, BlackBerry, Firefox OS, iOS, Windows Phone 8, Windows 7, and Windows 8. Support for other operating systems is available through separate downloads.

Support for other mobile device platforms is available but through separate downloads, typically from GitHub. It appears from the traffic on the Cordova dev lists that support for other platforms, such as Amazon Fire OS and Ubuntu, is under development as well.

As you can see, the list of supported platforms is broad, but only a few are really popular. For this book, I cover primarily Android and iOS, plus some others that I find interesting such as Windows Phone 8 and Firefox OS.

Coding Cordova Applications

As mentioned previously, Cordova applications are built using normal, everyday web technologies such as HTML, CSS, and JavaScript. Whatever you want your application to do, if you can make it work using standard web technologies, you can make it work in a Cordova application. Cordova applications can do more than standard web applications, through the specialized JavaScript libraries provided with the framework that I discussed earlier.

The Cordova project doesn't provide any special editor for writing Cordova applications; you simply need to dig out your web content editor of choice and start coding. To keep things simple, you could use default tools like Notepad on Microsoft Windows or TextEdit on a Macintosh. You could even use something more sophisticated such as Adobe Dreamweaver (www.adobe.com/products/dreamweaver.html) or the Eclipse integrated development environment (IDE) (www.eclipse.org).

Adobe, however, offers a free, open-source code editor called Brackets (http://brackets.io) that I've been playing around with. It provides a nice, clean interface for coding web applications. As it's an Adobe product, I expect that you'll see Cordova and/or PhoneGap integration capabilities in it before long.

For this book, I primarily coded using the open-source Aptana Studio (www.aptana.com), an Eclipse-based IDE tailored for web development. It's lighter weight than Eclipse and allowed me to easily format the project source code for easy importing into this manuscript (using two spaces instead of tabs everywhere).

Configuring a Cordova Development Environment

Before you can build applications using Apache Cordova, you must set up the appropriate development environment. The challenge for Cordova developers is that you must install the native SDKs, the software components the Cordova CLI requires, and finally the Cordova CLI. There's a lot to install, and the required components come from a lot of different sources. The good news is that all of the tools you will need are free and just a download away.

Chapter 3 of *Apache Cordova 3 Programming* describes the whole installation process in detail; you will need to refer to the Apache Cordova documentation or the book for the complete installation details. There are a lot of moving parts to this, and for that reason many people find the initial setup to be the hardest part of Cordova development.

Building Cordova Applications

Each of the mobile device platforms supported by the Cordova project has its own proprietary tools for packaging or building native applications. To build a Cordova application for each supported mobile platform, the application's web content (the HTML, CSS, JavaScript, and other files that constitute the application) must be added to an appropriate application project for each platform and then built using the platform's proprietary tools. What's challenging about this process is that each mobile platform uses completely different tools, and application projects use different configuration files and most likely a different project folder structure.

Additionally, some of the supported mobile platform development tools will run only on certain desktop operating systems. For example:

- The Android SDK runs on Linux, Microsoft Windows, and Macintosh OS X.
- The BlackBerry tools (there are several) run on Microsoft Windows and Macintosh OS X.
- The iOS SDK runs only on Macintosh OS X (no surprise there).
- The Windows Phone SDK runs only on Microsoft Windows (no surprise there either).

In the old days of Cordova development, you would use IDE plugins (on Android, iOS, and Windows Phone), command-line tools (on Android and BlackBerry), or start by copying a sample application (on bada, Symbian, and webOS) to create a new project. You would start with one of the supported platforms, write the appropriate web content, then package and test the application using the selected platform's SDK. Once you had it all working correctly, you would copy the web content over to a new project for one of the supported platforms and repeat the process. There was little consistency in project folder structure, framework JavaScript files (they had different file names on some platforms and were markedly different for each), and build process across mobile device platforms.

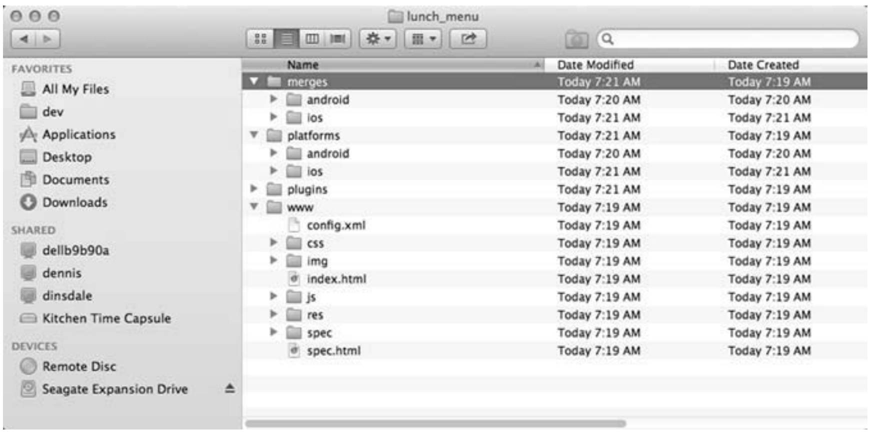
To make things easier, in later versions of the framework, the Cordova development team scrapped the IDE plugins and implemented a command-line interface for projects across a wider range of supported mobile device platforms. You use the command-line tools to create new projects, manage (add, remove, list, update) plugins, build, and then test applications using the device emulators and simulators. You can still do all of this by hand if you want to, but the command-line tools make it much easier.

Now, as this is a book about the Cordova APIs, I'm not going to spend too much time talking about the CLI and the development process. That particular topic is covered in great detail (about 200 pages' worth) in *Apache Cordova 3 Programming* (www.cordovaprogramming.com), but you can also find details in the Cordova Command-line Interface guide on the Cordova documentation site at http://cordova.apache.org/docs/en/3.0.0/guide_cli_index.md.html#The%20Command-line%20Interface and in the Platform Guides at http://cordova.apache.org/docs/en/3.0.0/guide_platforms_index.md.html#Platform%20Guides.

If you are building an app for Android and iOS, you would open a terminal window and execute the following:

```
cordova create lunch_menu
cd lunch_menu
cordova platform add android ios
```

At this point, what you'd have is a new Cordova project folder called lunch_menu with a bunch of subfolders, as shown in Figure 1.3. There's a platforms folder that contains native application projects for Android and iOS. Additionally, there's a folder called www that contains the application's core web content files, the content files that will be shared across the Android and iOS projects (or whatever platforms you want to use for your application).



Name	Date Modified	Date Created
merges	Today 7:21 AM	Today 7:19 AM
android	Today 7:20 AM	Today 7:20 AM
ios	Today 7:21 AM	Today 7:21 AM
platforms	Today 7:21 AM	Today 7:19 AM
android	Today 7:20 AM	Today 7:20 AM
ios	Today 7:21 AM	Today 7:21 AM
plugins	Today 7:21 AM	Today 7:19 AM
www	Today 7:19 AM	Today 7:19 AM
config.xml	Today 7:19 AM	Today 7:19 AM
css	Today 7:19 AM	Today 7:19 AM
img	Today 7:19 AM	Today 7:19 AM
index.html	Today 7:19 AM	Today 7:19 AM
js	Today 7:19 AM	Today 7:19 AM
res	Today 7:19 AM	Today 7:19 AM
spec	Today 7:19 AM	Today 7:19 AM
spec.html	Today 7:19 AM	Today 7:19 AM

Figure 1.3 Cordova Application Project Folder Structure

For your application, you will edit the web content stored in the www folder. When the web application content in that folder is ready for testing, you will use the CLI to copy the code into the platforms subfolders shown in the figure.

What I do while working on a Cordova project is keep my web content files open in an HTML editor like Adobe Brackets (www.brackets.io) or Aptana Studio (www.aptana.com) and then use the CLI to manage my mobile device platform projects for me. As I edit the files, I add the web content to the .html file and my application's code to the application's .js files; when I'm ready to test (and debug) the application, I switch over to a terminal window that I keep open and pointed to the Cordova project's root folder (the lunch_menu folder I created a while back) and issue some commands. If I want to switch to the Android IDE and test the Android application, I issue the following command:

```
cordova prepare android
```

Or, if I will be testing and debugging both the Android and iOS versions of the application, I issue the following command:

```
cordova prepare android ios
```

I could just prepare all target operating systems for the project using the following:

```
cordova prepare
```

What this command does is copy all of the project files from the www folder into the appropriate place for each platform project folder as shown in Figure 1.4. In this example, it copies the web content folder (www) to the Android project's assets/www folder and the iOS project's www folder.

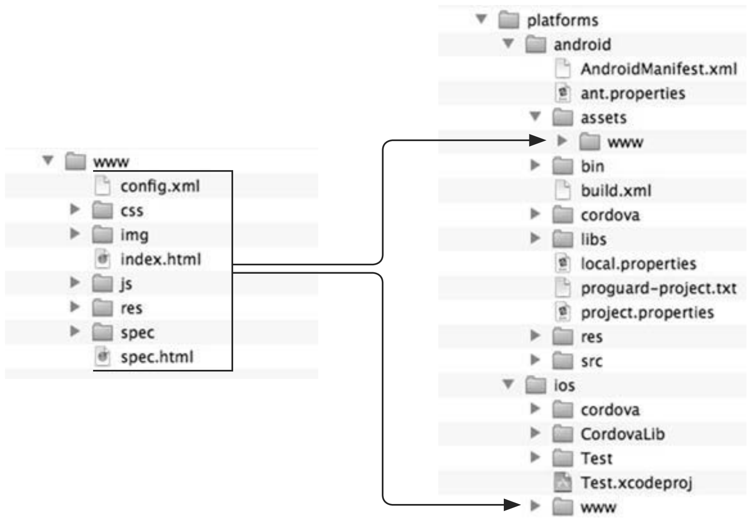


Figure 1.4 Copying Web Content to the Platform Projects Folders

With the project's files prepared, you can use the CLI to launch the application in an emulator or on a physical device for testing. You can also open the appropriate IDE and test/debug the application directly in the IDE. You can learn a lot more about the testing and debugging process in *Apache Cordova 3 Programming* (www.cordovaprogramming.com), Chapters 6 through 10.

Anatomy of a Cordova Application

Now that you know a little bit about how to create a Cordova application project, it's time to show you what makes a Cordova application a Cordova application. In this section, I show how to create a Cordova web application that leverages one of the Cordova Core APIs.

To begin, I opened a terminal window and navigated to the folder where I wanted to create the project. Next, I issued the following commands:

```
cordova create hellocordova1
cd hellocordova1
cordova platform add android ios
cordova plugin add org.apache.cordova.device
```

This created a hellocordova1 project folder, added the Android and iOS platforms to the project, and then added the code for the Cordova Device API. Next, I navigated to the project's www folder and pasted the code from Listing 1.1 into the project's existing index.html file.

Listing 1.1 Hello Cordova 1

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <meta name="viewport" id="viewport" content="width=device-width,
      height=device-height, initial-scale=1.0, maximum-scale=1.0,
      user-scalable=no;" />
    <script type="text/javascript" charset="utf-8" src="cordova.js">
    </script>
    <script type="text/javascript" charset="utf-8">
      function onBodyLoad() {
        document.addEventListener("deviceready", onDeviceReady, false);
      }

      function onDeviceReady() {
        var br = "<br />";
        //Get the appInfo DOM element
        var element = document.getElementById('devInfo');
        //Replace it with specific information about the device
        //running the application
        element.innerHTML = 'Cordova Version: ' + device.cordova + br +
          'Operating System: ' + device.platform + br +
          'OS Version: ' + device.version + br +
          'Device Model: ' + device.model + br +
          'Universally Unique Identifier: ' + device.uuid;
      }
    </script>
  </head>
  <body onload="onBodyLoad()">
    <h1>Cordova Information</h1>
    <p>This is an Apache Cordova application that makes calls to the
       Cordova Device API.</p>
    <p id="devInfo">Waiting for Cordova initialization to complete.</p>
  </body>
</html>
```

The index.html file shown in the listing is like any other HTML page you've seen, with some extra elements that enable it to understand how to interact with the Cordova container. The content-type setting is a standard HTML setting and should look the same as it would for any other HTML5 application. Within the `<Head>` section of the web page are two new entries, meta tags that describe the content type for the application and viewport settings.

The viewport settings shown in the following code tell the embedded web browser rendering the content how much of the available screen real estate should be used for the application and how to scale the content on the screen:

```
<meta name="viewport" id="viewport" content="width=device-width,
  height=device-height, initial-scale=1.0, maximum-scale=1.0,
  user-scalable=no;" />
```

In this case, the HTML page is configured to use the maximum height and width of the screen (through the `width=device-width` and `height=device-height` attributes) and to scale the content at 100% and not allow the user to change that in any way (through the `initial-scale=1`, `maximum-scale=1`, and `user-scalable=no` attributes).

Note

The viewport and associated attributes are not required. If they are omitted, the browser will revert to its default behavior, which may (or may not—who knows?) result in the application's content not consuming the full screen area available to it or zooming beyond it. Because there's not much content in the application, it could, for example, consume only the upper half of the screen on some devices. You may also find that on some platforms the settings have no effect—all the more reason to test your Cordova applications on a variety of mobile devices before release.

There's also a new script tag in the code that loads the Cordova JavaScript library:

```
<script type="text/javascript" charset="utf-8" src="cordova.js"></script>
```

This loads the Cordova API library and makes some Cordova capabilities available to the program.

The JavaScript code in a Cordova application does not have immediate access to any installed Cordova APIs after the web application has loaded. The native Cordova application container must complete its initialization process before it can respond to calls JavaScript made using the Cordova APIs. To accommodate this delay in API availability, a web developer building Cordova applications must instruct the container to notify the web application when the Cordova APIs have completed initialization. Any application processing that requires the use of the APIs should be executed by the application only after it has received notification that the APIs are available.

In the application, this notification is accomplished through the addition of an onload event defined in the page's body section as shown in the following:

```
<body onload="onBodyLoad()">
```

Within the `onBodyLoad` function, the code registers an event listener that instructs the application to call the `onDeviceReady` function when the device is ready, when the Cordova application container has finished its initialization routines and fired its deviceready event:

```
document.addEventListener("deviceready", onDeviceReady, false);
```

In this example application the `onDeviceReady` function updates the page rendered on the screen to display all of the available properties exposed by the Cordova Device API (described in Chapter 8) as shown in the following:

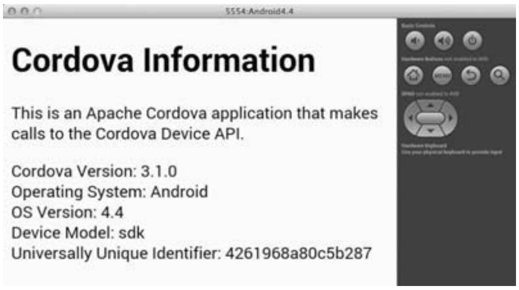
```
//Replace it with specific information about the device
//running the application
element.innerHTML = 'Cordova Version: ' + device.cordova + br +
  'Operating System: ' + device.platform + br +
  'OS Version: ' + device.version + br +
  'Device Model: ' + device.model + br +
  'Universally Unique Identifier: ' + device.uuid;
```

To run the application on an Android emulator, open a terminal window, navigate to the Cordova project folder, and issue the following command:

```
cordova emulate android
```

The default Android emulator will launch and display the application as shown in Figure 1.5.

When the application runs on an iOS simulator, it will display a screen similar to what is shown in Figure 1.6.

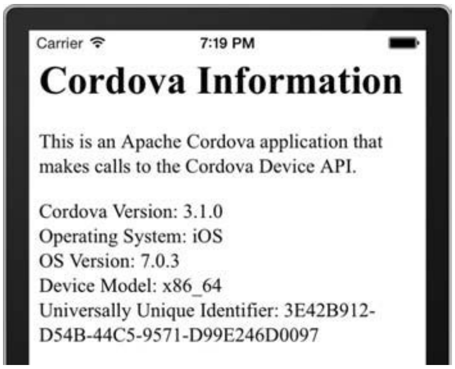


Cordova Information

This is an Apache Cordova application that makes calls to the Cordova Device API.

Cordova Version: 3.1.0
Operating System: Android
OS Version: 4.4
Device Model: sdk
Universally Unique Identifier: 4261968a80c5b287

Figure 1.5 Hello Cordova 1 Running on an Android Emulator



Cordova Information

This is an Apache Cordova application that makes calls to the Cordova Device API.

Cordova Version: 3.1.0
Operating System: iOS
OS Version: 7.0.3
Device Model: x86_64
Universally Unique Identifier: 3E42B912-D54B-44C5-9571-D99E246D0097

Figure 1.6 Hello Cordova 1 Running on an iOS Simulator

One of the common questions I get from people first learning Cordova is “Can I use HTML5 or JavaScript framework X with Cordova?” (substituting the name of their favorite HTML5 or JavaScript framework—jQuery Mobile, Sencha Touch, Dojo, and so on—into the question). The answer is unequivocally yes. The Cordova application simply renders whatever web content you pass to it using the native browser web view exposed by the mobile device OS.

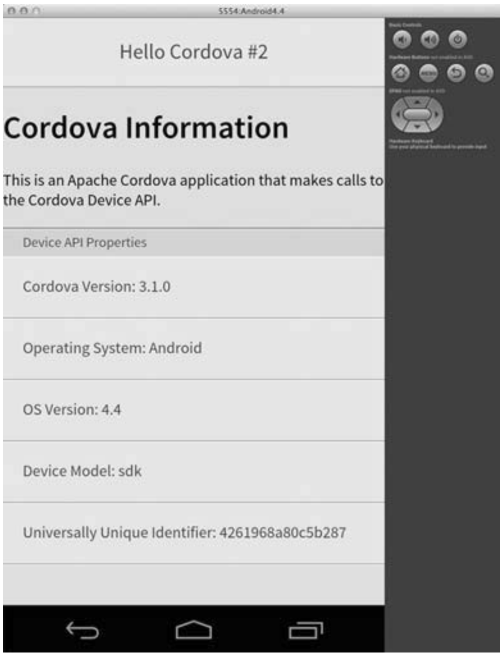
As a side project, to help developers easily build more beautiful mobile applications, Adobe created Topcoat (www.topcoat.io). Topcoat is a set of CSS files and open-source fonts that can be used to create fast, themeable, beautiful web sites. In an effort to make the sample applications highlighted in this book prettier, where appropriate I’m going to use Topcoat to apply styling to many of the applications.

So, once I downloaded the Topcoat files, I extracted them and copied over the font and CSS files to my project folder, then updated the application’s HTML to use the Topcoat styling. You can see an updated listing for the example application, now called Hello Cordova 2, in Listing 1.2.

Listing 1.2 Hello Cordova 2

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <meta name="viewport" id="viewport" content="width=device-width,
      height=device-height, initial-scale=1.0, maximum-scale=1.0,
      user-scalable=no;" />
    <link rel="stylesheet" type="text/css"
      href="css/topcoat-mobile-light.min.css">
    <script type="text/javascript" charset="utf-8" src="cordova.js">
    </script>
    <script type="text/javascript" charset="utf-8">
    function onBodyLoad() {
      document.addEventListener("deviceready", onDeviceReady, false);
    }
    function makeListItem(textStr) {
      return '<li class="topcoat-list__item">' + textStr + '</li>';
    }
    function onDeviceReady() {
      var tmpStr;
      tmpStr = '<ul class="topcoat-list__container">
        <h3 class="topcoat-list__header">Device API Properties</h3>';
      tmpStr+= makeListItem('Cordova Version: ' + device.cordova);
      tmpStr+= makeListItem('Operating System: ' + device.platform);
      tmpStr+= makeListItem('OS Version: ' + device.version);
      tmpStr+= makeListItem('Device Model: ' + device.model);
      tmpStr+= makeListItem('Universally Unique Identifier: ' +
        device.uuid);
      tmpStr+= '</ul>';
      //Get the appInfo DOM element
      var element = document.getElementById('devInfo');
      //Replace it with specific information about the device running
      //the application
      element.innerHTML =tmpStr;
    }
    </script>
  </head>
  <body onload="onBodyLoad()">
    <div class="topcoat-navigation-bar">
      <div class="topcoat-navigation-bar__item center full">
        <h1 class="topcoat-navigation-bar__title">Hello Cordova #2</h1>
      </div>
    </div>
    <h1>Cordova Information</h1>
    <p>This is an Apache Cordova application that makes calls to the
      Cordova Device API.</p>
    <div class="topcoat-list" id="devInfo">
      <h3 class="topcoat-list__header">
        Waiting for Cordova to initialize
      </h3>
    </div>
  </body>
</html>
```

I added the application's title to a title bar, assigning `class="topcoat-navigation-bar"`, `class="topcoat-navigation-bar__item center full"`, and `class="topcoat-navigation-bar__title"` to the elements of the header as shown in the listing. To render the list of Device API properties on the screen, I created an unordered list, with the appropriate class assignment, then applied `class="topcoat-list__item"` to each list item. When the modified application runs on an Android emulator, it displays a screen similar to what is shown in Figure 1.7.


Hello Cordova #2

Cordova Information

This is an Apache Cordova application that makes calls to the Cordova Device API.

Device API Properties

Cordova Version: 3.1.0

Operating System: Android

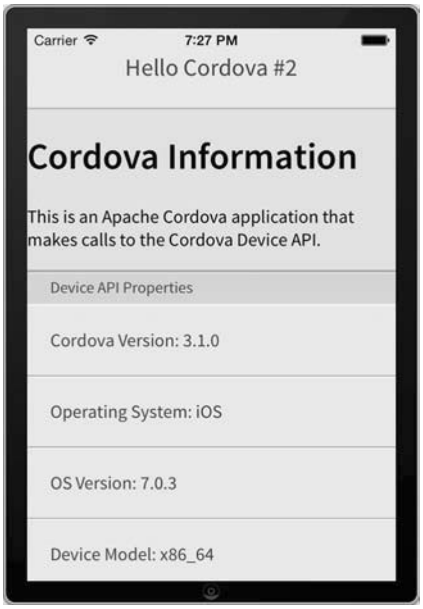
OS Version: 4.4

Device Model: sdk

Universally Unique Identifier: 4261968a80c5b287

Figure 1.7 Hello Cordova 2 Running on an Android Emulator

Notice how much better that looks? To prove it works as well on iOS, when the application runs on an iOS simulator, it displays a screen similar to what is shown in Figure 1.8.



Cordova Information

This is an Apache Cordova application that makes calls to the Cordova Device API.

Device API Properties

Cordova Version: 3.1.0

Operating System: iOS

OS Version: 7.0.3

Device Model: x86_64

Figure 1.8 Hello Cordova 2 Running on an iOS Simulator

Resources

There are many places online where you can find information about how to work with the Cordova framework. Table 1.1 lists the different web sites where you can find information about Apache Cordova. Adobe PhoneGap is Adobe's distribution of Apache Cordova with some extra capabilities added, so I have included some links to PhoneGap resources as well.

To stay informed about what's going on with the project, you can sign up for the mailing lists at http://cordova.apache.org/#mailing-list. If you have some extra time, it is fun to read through the emails as the development team discusses the implementation of a new feature or tracks down a bug.

The dev mailing list is used by the developers of the Cordova framework to discuss issues and make decisions about the Cordova implementation. The commits mailing list is for tracking commit logs for the Cordova repositories, when new or updated code is added to a version of the framework. The issues mailing list is for conversations around bug and feature requests submitted to the Cordova JIRA issue- and bug-tracking system at http://issues.apache.org/jira/browse/CB.

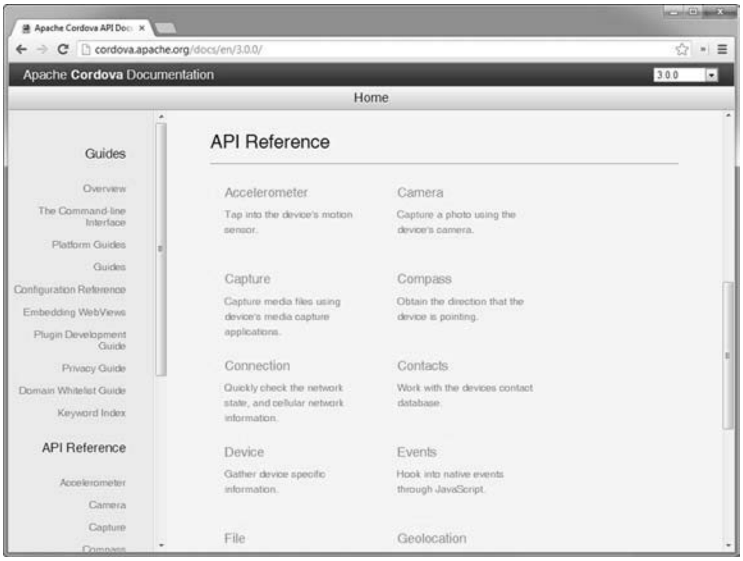
> **Caution**
>
> Please don't use the dev list to ask questions about Cordova development; use Google Groups instead.

Table 1.1 Available Resources

Resource	Link(s)
Cordova Web Site	http://cordova.io or http://cordova.apache.org (both point to the same site)
Cordova Documentation	http://docs.cordova.io
Cordova Wiki	http://wiki.cordova.io
Cordova Issue Tracker	https://issues.apache.org/jira/browse/CB
Cordova Mailing Lists	http://cordova.apache.org/#mailing-list
Cordova Twitter Account	http://twitter.com/apachecordova
PhoneGap Web Site	http://www.phonegap.com
PhoneGap Wiki	http://wiki.phonegap.com
PhoneGap Blog	http://www.phonegap.com/blog
PhoneGap Twitter Account	https://twitter.com/phonegap

You'll spend the majority of your time on the Apache Cordova Documentation site that is shown in Figure 1.9. The site contains a complete reference to all of the Cordova APIs plus additional guides you'll need as you work with the framework.

The API reference shown in the figure includes a complete reference for all of the methods, properties, and events for each of the Cordova APIs. On the API pages you'll also find sample source code and additional information you will need to make use of the APIs.



API Reference

Accelerometer
Tap into the device's motion sensor.

Camera
Capture a photo using the device's camera.

Capture
Capture media files using device's media capture applications.

Compass
Obtain the direction that the device is pointing.

Connection
Quickly check the network state, and cellular network information.

Contacts
Work with the devices contact database.

Device
Gather device specific information.

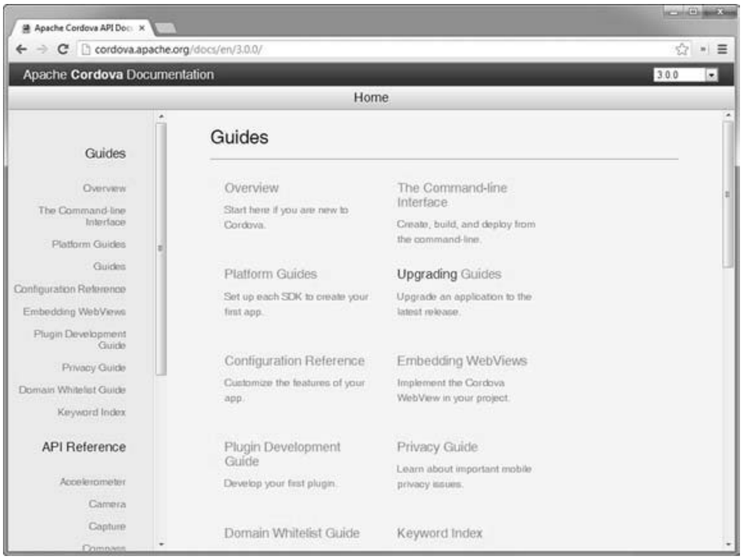
Events
Hook into native events through JavaScript.

File

Geolocation

Figure 1.9 Apache Cordova Documentation

While you're looking at the Documentation site, scroll down within either the left or the right side of the page to see the list of guides shown in Figure 1.10. These guides contain a lot of useful information a developer needs to work with the framework, including how to create plugins, using the command-line tools, and, most important, the getting-started guides for each of the supported mobile device platforms.



Guides

Overview
Start here if you are new to Cordova.

The Command-line Interface
Create, build, and deploy from the command-line.

Platform Guides
Set up each SDK to create your first app.

Upgrading Guides
Upgrade an application to the latest release.

Configuration Reference
Customize the features of your app.

Embedding WebViews
Implement the Cordova WebView in your project.

Plugin Development Guide
Develop your first plugin.

Privacy Guide
Learn about important mobile privacy issues.

Domain Whitelist Guide

Keyword Index

Figure 1.10 Cordova Documentation—Guides Section

Wrap Up

So, that's it—a quick overview of Apache Cordova with a quick development tutorial and some examples. With the information provided here, you have the background information you need to work through the remainder of the content in the book. The remainder of the book is dedicated to detailed instructions on how to leverage each of the Cordova APIs in your Cordova applications.

This page intentionally left blank

Index