

MOBILE
PROGRAMMING
SERIES



APACHE CORDOVA API COOKBOOK

JOHN M. WARGO

Foreword by BRIAN LEROUX

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



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

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw.

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

Copyright © 2015 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

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

Foreword	xiii
Preface	xv
Acknowledgments	xvii
About the Author	xix
1 Introduction to Apache Cordova	1
Introduction to Apache Cordova	1
Supported Platforms	5
Coding Cordova Applications	5
Configuring a Cordova Development Environment	6
Building Cordova Applications	6
Anatomy of a Cordova Application	9
Resources	15
Wrap Up	17
2 Accelerometer	19
Measuring Device Orientation	20
Watching a Device's Orientation	28
What Can Go Wrong	33
3 Camera	35
Taking a Picture	36
Configuring Camera Options	44
allowEdit	46
cameraDirection	47
correctOrientation	47
destinationType	48
encodingType	51
mediaType	51
quality	52
saveToPhotoAlbum	52

sourceType	53
targetHeight and targetWidth	57
iOS-Only Capabilities	58
What Can Go Wrong	58
4 Capture	61
Using the Capture API	62
Configuring Capture Options	66
duration	66
limit	66
Capture in Action	66
Capturing Audio	72
Capturing Images	74
Capturing Video	76
What Can Go Wrong	77
5 Compass	79
Getting the Device Heading	79
Watching the Device Heading	86
What Can Go Wrong	95
6 Connection	97
Using the Connection API	97
An Example	99
What Can Go Wrong	103
7 Contacts	105
Creating a Contact	106
Searching for Contacts	115
Cloning Contacts	124
Removing Contacts	124
What Can Go Wrong	124
8 Device	127
Using the Device API	127
An Example	128
What Can Go Wrong	133

9 Events 135

- Managing Event Listeners 136
- deviceready Event 137
- Application Status Events 138
- Battery Status Events 141
- Button Events 144
- Network Status Events 149
- What Can Go Wrong 152

10 File 153

- Storage Locations 154
- Accessing the Device's File System 156
- Reading Directory Entries 159
- Accessing `FileEntry` and `DirectoryEntry` Properties 163
- Writing Files 167
- Reading Files 170
- Deleting Files or Directories 172
- Copying Files or Directories 173
- Moving Files or Directories 173
- Transferring Files 174
 - File Uploads 175
 - File Downloads 178
- What Can Go Wrong 179

11 Geolocation 181

- Getting a Device's Current Location 182
- Watching a Device's Location 189
 - Setting a Watch 190
 - Canceling a Watch 192
- What Can Go Wrong 197

12 Globalization 201

- Example Applications 202
- Using the Globalization API 204
 - Success Callback 205
 - Error Callback 206

Currency	206
getCurrencyPattern	206
Date	208
getDatePattern	208
getDateNames	209
getFirstDayOfWeek	211
isDayLightSavingsTime	212
dateToString	212
stringToDate	215
Language and Locale	217
getPreferredLanguage	217
getLocaleName	218
Number	218
getNumberPattern	218
numberToString	220
stringToNumber	221
What Can Go Wrong	223

13 InAppBrowser 225

Example Application	225
Managing an InAppBrowser Window	227
open, show, and hide	227
InAppBrowser in Action	229
InAppBrowser Events	236
Executing Scripts in an InAppBrowser Window	237
Insert CSS	238
What Can Go Wrong	239

14 Media 241

The Media Object	241
Creating a Media Object	242
Current Position	246
Duration	246
Releasing the Media Object	247
Playing Audio Files	247
play	247
pause	248

stop	248
seek	248
Set Playback Volume	248
Media Playback in Action	249
Accessing Local and Remote Media Files	253
Implementing Play, Pause, and Stop	255
Updating the Application's UI	257
Recording Audio Files	259
Start Recording	259
Stop Recording	259
Media Recording in Action	259
What Can Go Wrong	263
15 Notification	265
Visual Alerts	266
Getting Input	269
Confirm	269
Prompt	272
Audible and Tactile Notifications	275
Beep	275
Vibrate	275
What Can Go Wrong	275
16 Splashscreen	277
Using the Splashscreen API	277
Using Custom Splash Screen Images	281
What Can Go Wrong	284
Index	285

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.

This page intentionally left blank

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

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.

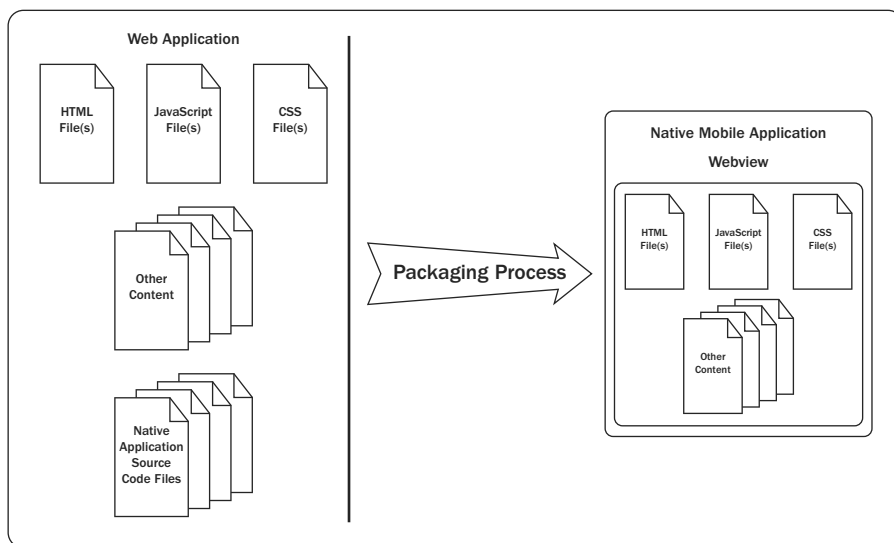


Figure 1.1 Apache Cordova Application Packaging Process

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.

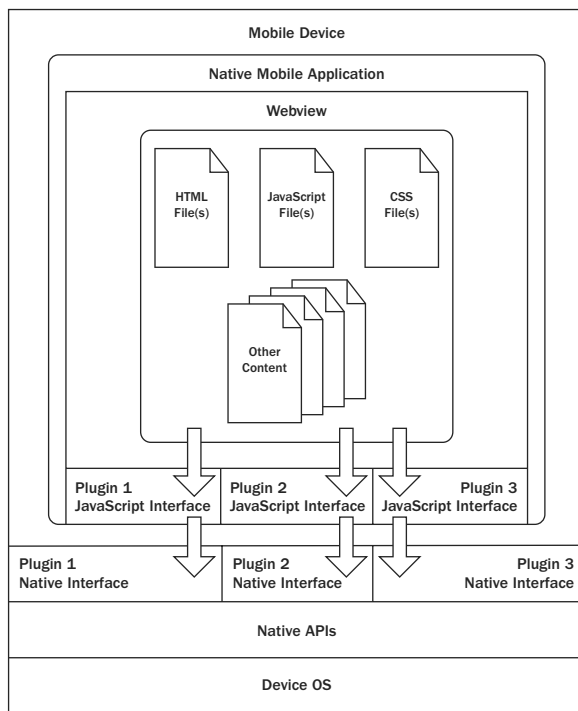


Figure 1.2 Apache Cordova Native Application Architecture Post-3.0

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).

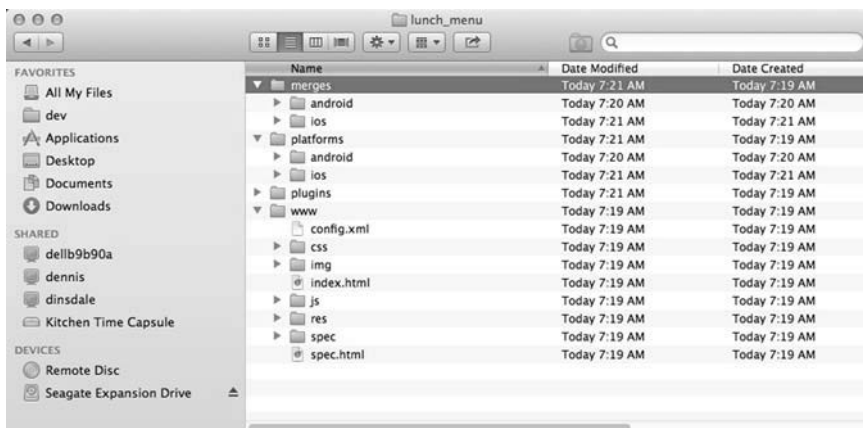


Figure 1.3 Cordova Application Project Folder Structure

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.

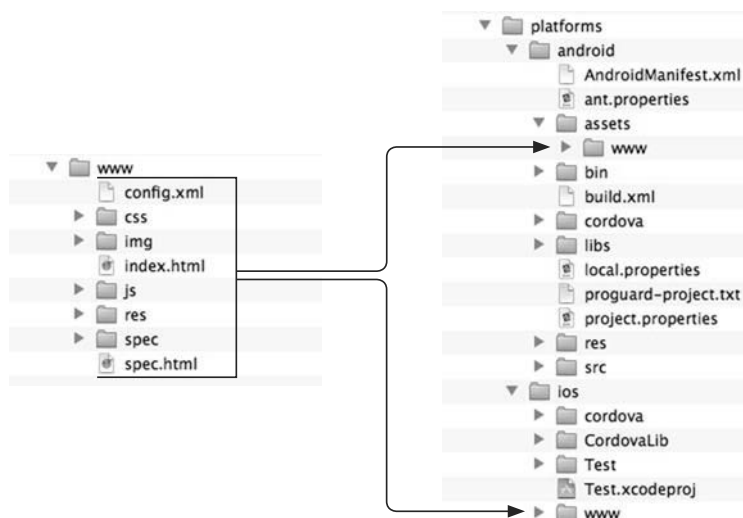


Figure 1.4 Copying Web Content to the Platform Projects Folders

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.



Figure 1.5 Hello Cordova 1 Running on an Android Emulator

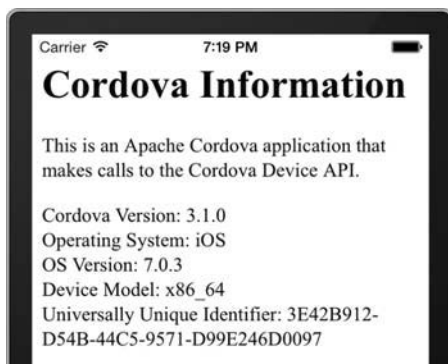


Figure 1.6 Hello Cordova 1 Running on an iOS Simulator

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.



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.



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.

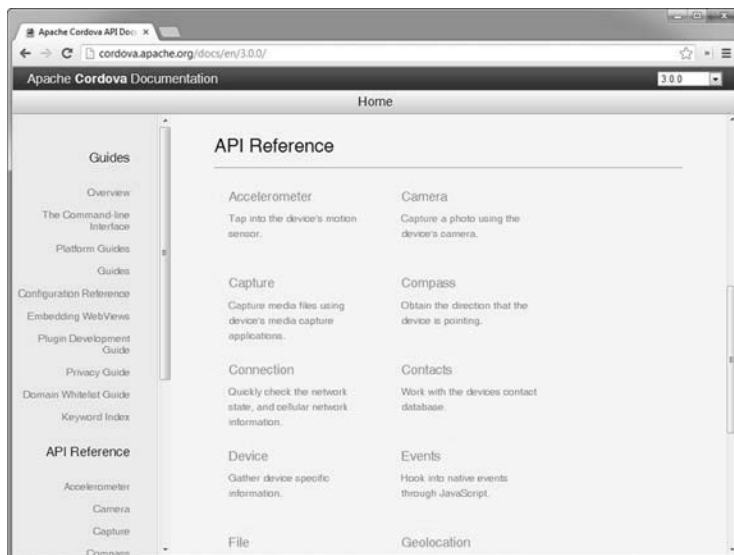


Figure 1.9 Apache Cordova Documentation

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.

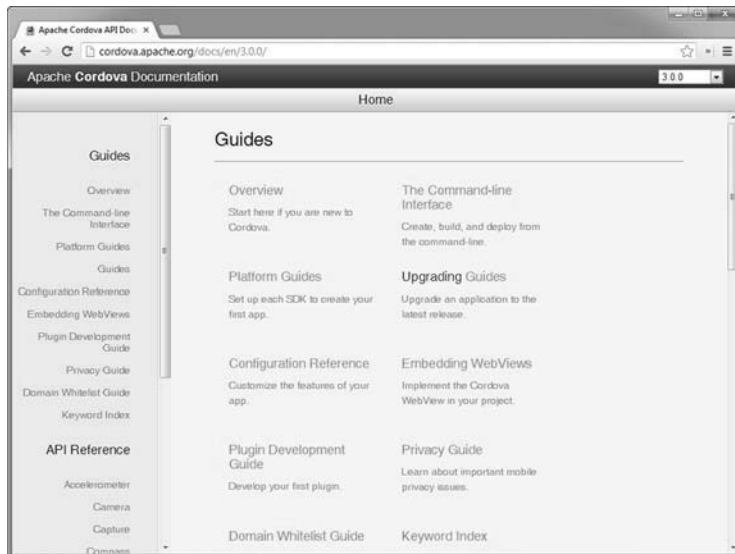


Figure 1.10 Cordova Documentation—Guides Section

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

A

Accelerometer API

- Geolocation API and, 181
- introduction to, 19–20
- measuring device orientation in, 20–28
- troubleshooting, 33–34
- watching device orientation in, 28–33

Access

- to device file systems, 156–159
- to file/directory properties, 163–166
- to media files, 253–255
- to network connections, 103

Addresses of contacts, 106–108

Adobe Brackets, 5, 8

Adobe Dreamweaver, 5

Adobe Topcoat. *See* Topcoat

Albums for photographs, 53–56

Alerts

- in Notification API, 265–269, 276

`allowEdit`, **46**

`allowInlineMediaPlayback`, **228**

Anatomy of Cordova applications, 9–15

Android devices

- Accelerometer API on, 20, 25, 32–33
- battery status events on, 144
- building Cordova apps for, 7–9

Android devices, (continued)

- Camera API on, 38–40, 44, 49–58
- Capture API on, 67, 72–76
- Compass API on, 84–85, 92
- Connection API on, 102
- Contacts API on, 110, 114, 117, 122
- copying web content folders to, 8
- core web content files on, 7
- Device API on, 129–130
- Emulator for, 11–12, 14, 25
- Enabling GPS on, 199
- Events API on, 140, 144, 148
- file details on, 165
- file readers on, 171–172
- Geolocation API on, 188, 197–199
- Globalization API on, 203–204
- GPS Disabled Error on, 199
- InAppBrowser API on, 225–228, 230–234, 238–239
- insertCSS on, 238
- Media API on, 243, 249–254, 260
- Notification API on, 266, 270, 272–273
- operating systems for, 6
- persistent storage locations in, 162
- Splashscreen API on, 283–284
- supported platforms for, 5
- Timeout Error messages on, 198

Apache Cordova. *See* **Cordova**

Apache Cordova 3 Programming

- on command-line interface, 7, 9
- on console exposure, 27

API Reference, 16

Apple iOS devices. *See* **iOS devices**

Application status events, 135, 138–141

Aptana Studio, 6, 8

Araxis Merge, 123

Audible notifications, 275

Audio files

- capture of, 61–62, 72–74
- playing, 247–253
- recording, 259–262

B

backbutton events, 135, 144

Back-facing cameras, 47

Bacon Ipsum generators, 234

Battery events, 135–137, 141–144

Beep tones, 265–266, 275

Birthdays, 106

BlackBerry devices

- accelerometer values on, 20
- button events on, 144
- Cordova APIs on, 5–6
- physical compasses in, 79

_blank, 227, 235

Browser windows, 229–234

Building applications, 6–9

Buttons

- in Events API, 135–136, 144–148
- in InAppBrowser API, 228
- in Media API, 255–257
- in Notification API, 270–271

C

Callbacks

- in Compass API, 80, 84
- in Contacts API, 110, 112–113, 116
- error. *See* **error callbacks**
- failure, 24, 62

- in File API, 153
- in Globalization API, 205–206
- in Media API, 243–246
- in Notification API, 267–268
- optional, 243, 245
- status, 243–245
- success. *See* success callbacks
- updateUI, 257

Camera API

- allowEdit in, 46
- cameraDirection in, 47
- Capture API vs., 35, 61, 64–65
- configuring options in, 44
- correctOrientation in, 47
- DATA_URL on, 48–50
- destinationType in, 48–51
- encodingType in, 51
- FILE_URI on, 48–49
- geolocation and, 181
- introduction to, 35
- iOS-only capabilities in, 58
- mediaType in, 51
- NATIVE_URI on, 48, 50–51
- quality in, 52
- saveToPhotoAlbum in, 52–53
- sourceType in, 53–56
- taking photographs with, 36–44
- targetHeight/targetWidth in, 57–58
- troubleshooting, 58–59

Cameras. *See also* Camera API

- back-facing, 47
- in Capture API, 35, 61, 64–65
- front-facing, 47
- video, 51

Capture API

- audio capture in, 61–62, 72–74

- Camera vs., 35, 61, 64–65
- configuring options in, 66
- duration in, 66
- image capture in, 74–75
- introduction to, 61
- limit in, 66
- Media API vs., 241, 259
- troubleshooting, 77
- using, 62–65, 66–72
- video capture in, 76

Cascading Style Sheets (CSS)

- in Cordova, 3
- in InAppBrowser API, 238–239
- in Topcoat, 12–14

Cellular connections, 98, 103. *See also* Connection API

CFW. *See* Continuous File Writer

ChildBrowser, 225

- clearcache, 228

- clearsessioncache, 228

CLIs. *See* Command-line interfaces

Cloning contacts, 124

- close method, 235

- closebuttoncaption, 228

- code for currencies, 207

Command-line interface (CLI), 3, 6–9

Compass API

- Geolocation API and, 181
- getting device headings in, 79–86
- introduction to, 79
- troubleshooting, 75
- watching device headings in, 86–95

- confirm capabilities, 265–266, 269–271

Connection API

- detecting current network type in, 99–103

Connection API, (continued)

- display alerts to users in, 99–103
- Events API vs., 149
- introduction to, 97
- troubleshooting, 103
- using, 97–99

Contacts API

- cloning contacts in, 124
- creating contacts in, 106–114
- introduction to, 105
- removing contacts in, 124
- searching for contacts in. *See* searching for contacts
- troubleshooting, 124–125

Continuous File Writer (CFW), 169–170**Coordinated Universal Time (UTC), 209****Copying files/directories, 173****Cordova. *See also* specific APIs**

- anatomy of applications for, 9–15
- building applications for, 6–9
- coding applications for, 5–6
- cookbook for. *See* cordovacookbook.com
- development environment
 - configuration for, 6
- introduction to, 1–5, 17
- resources for, 15–17
- supported platforms in, 5
- website on. *See* cordova.apache.org
- webviews in, 3–5

Cordova Domain Whitelist Guide, 255**cordova.apache.org**

- Accelerometer API documentation on, 20
- Camera API documentation on, 35, 45
- on command-line interfaces, 7

Cordova Domain Whitelist Guide on, 255

File API documentation on, 169

introduction to, 1

updates on, 15–16

cordovacookbook.com

Contacts code on, 105, 116

File code on, 154

InAppBrowser code on, 226

Media code on, 253

Notification code on, 265

Core APIs, 2

`correctOrientation`, 47

CRM. *See* Customer relationship management

CSS (Cascading Style Sheets). *See* Cascading Style Sheets (CSS)

Currency

- converting numeric values to strings, 220–221
- converting string values to numbers, 221–222
- `getCurrencyPattern`, 206–207
- in Globalization API, 201, 203–204
- ISO 4217 codes for, 206–207
- number settings and, 218

Current positions, 246**Custom images, 281–284**

Customer relationship management (CRM), 105

`cvaReady`, 24, 32, 84

D

`DATA_URL`, 48–50

Date settings, 202–204, 208–217

`dateToString`, 212–215

day, **215**

Daylight Savings Time, 209, 212

Days of week, 208, 211

Debug Console, 200

decimal

in converting numeric values to strings,
220–221

in converting string values to numbers,
221–222

in currency settings, 207

in number settings, 218–219

Degree-based watches, 93

Deleting files/directories, 172–173

destinationType, **48–51**

Detecting current network type, 99–103

Development environment configuration, 6

Device API

Hello Cordova 1 example of, 11

Hello Cordova 2 example of, 128–132

introduction to, 127

troubleshooting, 133

using, 127–128

deviceready events, 135, 137–138

DirectoryEntry

for copying files/directories, 173

for deleting files/directories, 172–173

for moving files/directories, 173–174

DirectoryReader, **159–160, 163**

displayName, **111, 115–116, 120**

Documentation site, 16–17

Downloading files, 178–179

Dreamweaver, 5

Dropbox, 53

duration, **66, 246**

E

Eclipse, 5–6

Editing images, 46

Email addresses, 106, 108–109, 114

enableHighAccuracy, **182**

enableViewportScale, **228, 231–232**

encodingType, **51**

endcallbutton **events, 135, 145**

English language, 201

Error callbacks. See also failure callbacks

in Accelerometer API, 24, 26

in Camera API, 40

in Capture API, 64–66, 73

in Compass API, 80, 84, 87

in Contacts API, 112–113

in File API, 157–160

in Geolocation API, 182, 187, 190,
198–199

in Globalization API, 204–206

in InAppBrowser API, 236

in Media API, 243–245, 247, 259

in uploading files, 176–177

Error messages

in Camera API, 51, 58–59

in Contacts API, 112–113

in File API, 158–159, 176–177

in Geolocation API, 198

in InAppBrowser API, 236

in Media API, 243

Escape buttons, 145

Event listeners

and battery status events, 144

button events and, 145–146

in Connection API, 102

in Events API, 135–137

Event listeners, (continued)

- in InAppBrowser API, 227, 236–238
- introduction to, 11
- network status events and, 151

Events

- application status, 135, 138–141
- backbutton, 135, 144
- battery, 135–137
- deviceready, 135, 137–138
- endcallbutton, 135, 145
- in `FileReader`, 171
- in `FileWriter`, 169
- in InAppBrowser API, 236–237
- menubutton, 136, 144
- network, 135
- network status, 149–152
- offline, 136
- online, 136
- onload, 11
- pause, 136, 138–141
- resume, 136, 138–141
- searchbutton, 136, 145
- startcallbutton, 136, 145
- volumedownbutton, 136, 145
- volumeupbutton, 136, 145

Events API

- application status events in, 138–141
- battery status events in, 141–144
- button events in, 144–148
- Connection API vs., 149
- deviceready events in, 137–138
- event listeners in, 136–137
- introduction to, 135–136
- network status events in, 149–152
- troubleshooting, 152

Executing scripts, 237–238**EXIF data, 47**

F

Failure callbacks, 24, 62. See also error callbacks**File API**

- accessing device file systems in, 156–159
- accessing file/directory properties in, 163–166
- copying files/directories in, 173
- deleting files/directories in, 172–173
- file downloads in, 178–179
- file uploads in, 175–178
- introduction to, 153–154
- Media API and, 261
- moving files/directories in, 173–174
- reading directory entries in, 159–162
- reading files in, 170–172
- storage locations in, 154–156
- transferring files in, 174–179
- troubleshooting, 179–180
- writing files in, 167–170

File formats, 252**File URIs (uniform resource identifiers). See URIs (uniform resource identifiers)****FileEntry**

- for copying files/directories, 173
- for deleting files/directories, 172–173
- for moving files/directories, 173–174

FileReader, 170–172**FileTransfer, 174–178****FileWriter, 167–170****Filter values, 116****Find method, 115, 118, 122**

Firefox OS

Device API on, 130–131

File API on, 180

GPS on, 199–200

Notification API on, 276

as supported platform, 5

First day of week settings, 211–212**Formats**

of dates, 208, 213, 215

of files, 252

`fraction`, **207, 219**

Front-facing cameras, 47

`fullPath`, **163**

G

Games

Accelerometer API and, 19–21

Events API and, 135

Media API and, 241

Geographic North Pole, 84**Geolocation API**

Accelerometer and, 181

canceling location watches in,
192–197

Compass and, 181

current location of devices in,
182–189

introduction to, 181–182

setting location watches in, 190–192

troubleshooting, 197–200

watching location of devices in, 189

`getCurrencyPattern`, **201, 203–204, 206–207**

`getCurrentAcceleration`, **21–24, 27**

`getCurrentHeading`, **79–86**

`getCurrentPosition`, **182–183**

`getDateNames`, **209–211**

`getDatePattern`, **208–209**

`getFile`, **163, 167, 170**

`getFirstDayOfWeek`, **211–212**

`$.getJSON` **method**, **110**

`getLocaleName` **in**, **218**

`getNumberPattern`, **218–220**

`getPicture`, **36–38, 44–46**

`getPreferredLanguage`, **205, 217**

GitHub

Camera code on, 45

Contacts code on, 105, 116

File code on, 154

Globalization code on, 202

InAppBrowser code on, 226

Media code on, 253, 260

Notification code on, 265

Globalization API

currency settings in, 206–207

date settings in, 208–217

`dateToString` **in**, 212–215

error callbacks in, 206

example applications in, 202–204

`getCurrencyPattern` **in**, 206–207

`getDateNames` **in**, 209–211

`getDatePattern` **in**, 208–209

`getFirstDayOfWeek` **in**, 211–212

`getLocaleName` **in**, 218

`getNumberPattern` **in**, 218–220

`getPreferredLanguage` **in**, 217

introduction to, 201–202

`isDayLightSavingsTime` **in**, 212

language settings in, 217

locale settings in, 217–218

Globalization API, (*continued*)

- number settings in, 218–223
- numberToString in, 220–221
- stringToDate in, 215–217
- stringToNumber in, 221–223
- success callbacks in, 205
- troubleshooting, 223
- using, 204–205

Google Android. *See* **Android devices**

GPS services. *See* **Geolocation API**

grouping, **207, 219**

Guides for Cordova, 17

H

Hello Cordova 1 example, 9–10

Hello Cordova 2 example, 13–14, 128–132

hidden **browser windows, 228–229**

hide **InAppBrowser windows, 227–229**

hour, **215**

HTML (HyperText Markup Language)

- anchor links in, 227
- Brackets for, 8
- break tags in, 170
- for buttons in File API, 155
- in Cordova, 1–4, 12
- deviceready events and, 152
- in Hello Cordova 1 example, 9–10
- image tags in, 42
- for InAppBrowser events, 236
- index.html files in, 161, 225
- jQuery items and, 71
- media content in, 228–229
- unordered lists in, 120, 160–161
- W3C Geolocation API and, 182

<http://cordova.apache.org>.

See cordova.apache.org

Hybrid applications, 1

HyperText Markup Language (HTML). *See*
HTML (HyperText Markup Language)

I

IDE. *See* **Integrated development environment**

IM. *See* **Instant Messaging**

Images

- in Camera API. *See* **Camera API**
- in Capture API, 61–62, 74–75
- of contacts, 107–108

InAppBrowser API

- events in, 236–237
- example applications in, 225–226
- executing scripts in, 237–238
- hide windows in, 227–229
- inserting CSS in, 238–239
- introduction to, 225
- open windows in, 227–229
- show windows in, 227–229
- troubleshooting, 239–240
- using, 229–236
- windows in, 227–229

Index.html files, 9–10. *See also* **HTML (HyperText Markup Language)**

Inserting CSS (Cascading Style Sheets), 238–239

Instant Messaging (IM) addresses, 107–108

Integrated development environment (IDE), 6–9

International Organization for Standardization (ISO), 206–207

INVALID_ARGUMENT_ERROR, **113**

iOS devices

- Accelerometer API on, 20–21, 26–27, 32–33
- application status events on, 138
- battery status events on, 144
- building Cordova apps for, 7–9
- button events on, 144
- Camera API on, 40–42, 47–51, 55–58
- Capture API on, 73–76
- Compass API on, 86, 93–94
- Contacts API on, 113–114, 123
- copying web content folders to, 8
- Cordova APIs for, 11
- core web content files on, 7
- date settings on, 215
- Device API on, 131
- Events API on, 141, 152
- file details on, 166
- file downloads on, 178–179
- file uploads on, 177–178
- Geolocation API on, 187–188
- globalization functions on, 202
- InAppBrowser API on, 228–229, 231–232
- Media API on, 243, 250–252, 262
- operating systems for, 6
- Simulator for, 11–12, 14–15, 27
- Splashscreen API on, 277–282
- supported platforms for, 5

iPad

- Device API on, 127, 132
- globalization functions on, 202

iPhone

- accelerometer values on, 20–21
- Device API on, 130–131
- physical compasses on, 79

isDayLightSavingsTime, **212**

isDirectory, **163**

isFile, **163**

isOffline, **151**

isOnline, **151**

ISO (International Organization for Standardization), 206–207

J

JavaScript

- in Accelerometer API, 22, 24
- in Camera API, 36–38, 49, 51
- in Capture API, 67
- in Compass API, 81
- in Cordova, 2–5, 11–12
- FileReader and, 171
- in Globalization API, 203, 205, 212–213
- in InAppBrowser API, 227, 229, 237–238
- in Media API, 246
- in Notification API, 266–267, 276
- removeEventListener in, 136, 145
- in Splashscreen API, 277

JPEG files, 47, 51

jQuery

- \$ method, 72, 103
- for Capture API, 64, 71–72
- for Contacts API, 110, 116
- for File API, 161
- for InAppBrowser API, 237
- for network status updates, 151–152

jQuery Mobile

- for Contacts API, 110, 116, 119–120
- for File API, 155, 161
- for Notification API, 276
- Topcoat vs., 67

jQueryRotate, 91

JSON, 110

K

Kapsel Logon plugin, 225

`keyboardDisplayRequiresUserAction`,
229

L

Landscape mode, 204

Language options, 201, 205, 217

`limit`, **66**

`listview`, **119–120**

Load functions, 236

Locale settings, 217–218

Location. See Geolocation API

Location watches

canceling, 192–197

introduction to, 189

setting, 190–192

M

Magnetic Heading, 85

Magnetic North Pole, 84

Mailing lists, 15–16

Malware, 105

`maximumAge`, **182**

Media API

accessing media files in, 253–255

callback functions in, 243–245

Capture vs., 241, 259

creating Media objects in, 242–246

current positions in, 246

duration in, 246

File API and, 261

file format problems in, 252

file URIs in, 242–243

introduction to, 241

objects in, 241

pause function in, 248, 255–257

play function in, 247, 255–257

playing audio files in, 247–253

recording audio files in, 259–262

releasing Media objects in, 247

seek in, 248

`startRecord` in, 259

stop functions in, 248, 255–257, 259

troubleshooting, 263

updating UIs in, 257–258

volume in, 248–249

Media-capture plugin, 77

`MediaFile` **objects, 63**

`mediaPlaybackRequiresUserAction`, **229**

`mediaType`, **51**

`menubutton` **events, 136, 144**

Metadata, 164–165

**Microsoft Windows phones. See Windows
Phone devices**

`millisecond`, **215**

`minute`, **215**

`month`, **215**

Moving files/directories, 173–174

multiple values, 116–118

N

Names of contacts, 106–107

Names of files, 163

Native mobile applications, 2`NATIVE_URI`, 48, 50–51`navigator.contacts.create`, 106–107`navigator.contacts.save`, 106`navigator.network.connection.type`, 97–98, 103**Negative numbers, 219****Network connections. See Connection API****Network events, 135, 149–152****Nexus 7 tablets**

Capture API on, 73

cellular coverage on, 198

date settings on, 214, 216–217

Media API on, 249

Nicknames, 107**North Poles, 80****Notification API**

audible notifications in, 275

beep tones in, 275

`confirm` in, 269–271

getting user input in, 269–274

introduction to, 265–266

JavaScript vs. Cordova code in, 267–268

`prompt` in, 272–274

tactile notifications in, 275

troubleshooting, 275–276

vibrate mode in, 275

visual alerts in, 266–269

Number settings, 218–223`numberToString`, 220–221

O

Offline events, 136`onAccelFailure`, 21–24, 28`onAccelSuccess`, 21–24, 28`onBackButton`, 145**Online events, 136****Onload events, 11**`open windows`, 227–229**Operating systems, 6**

Optional error callbacks, 243

Optional status callbacks, 245

Organizations in Contacts API, 107–108

P

pattern

in currency settings, 207

in date settings, 209

in number settings, 219

Pause functions

in Events API, 136, 138–141

in Media API, 248, 255–257

percent

in converting numeric values to strings, 220–221

in converting string values to numbers, 221–223

in number settings, 218

Permissions

for accessing network connections, 103

in Contacts API, 113

in Geolocation API, 198

Persistent storage, 153–155, 161**Phone numbers, 107–108****PhoneGap, 15–16, 57****PhoneGap Essentials, 57****Photographs. See also Camera API**

in Capture API, 61–62, 74–75

of contacts, 107–108

PhotographsI, (continued)

saving, 52–56

selfies, 44

taking, 36–44

Play functions

for audio files, 247–253

buttons for, 255–257

in Media API, 247

PNG files, 51**Positive numbers, 219**

presentationstyle, 229

processEntry, 163–164

prompt, 265–266, 272–274

Q

quality, 52

Quirks, 45

R**Radios, 98, 103****Reading directory entries, 159–163****Reading files, 170–172****Rear cameras, 47****Recording audio files, 259–262****Releasing Media objects, 247**

removeEventListener, 145

Removing contacts, 124

requestFileSystem, 156–157, 159

Resources, 15–17**Resume events, 136, 138–141****Rotation of screens, 92–93**

rounding, 207, 219

S

sandboxSize, 156

SAP Kapsel SDK, 225**SAP Mobile Platform, 225****SAP U15, 225**

saveToPhotoAlbum, 52–53

Searchbutton events, 136, 145**Searching for contacts**on different mobile devices, 115, 117,
122–123

displayName for, 111, 115–116, 120

field values in, 115, 122–123

filter values in, 116, 118

find method in, 115, 118, 122

limiting number of contact fields in,
115

listview in, 119–120

multiple values in, 116–118

second, 215

seek, 248

_self, 227, 234, 235

Selfies, 44

show InAppBrowser windows, 227–229

showContact, 115, 120

Siri, 144**Smartphones. See also specific types**

application status events on, 138

cameras on. *See* Camera API

geolocation and, 181–182

in landscape mode, 204

network connections on, 98

physical compasses in, 79

storage on, 154

Source code, 1

sourceType, 53–56

Splashscreen API

- custom images in, 281–284
- introduction to, 277
- troubleshooting, 284
- using, 277–281

Start functions

- in InAppBrowser API, 236
- for recordings, 259
- startcallbutton events, 136, 145

Status callbacks, 245, 250**Status events**

- application, 135, 138–141
- battery, 141–144
- network, 149–152

Stop functions

- buttons for, 255–257
- in InAppBrowser API, 236
- in Media API, 248
- for recordings, 259

Storage, 153–156, 161–162**Strings**

- converting numeric values to, 220–221
- converting to numbers, 221–222
- numberToString for, 220–221
- stringToDate for, 215–217
- stringToNumber in, 221–223

Success callbacks

- in Accelerometer API, 24, 27
- in Capture API, 62, 64
- in Compass API, 80, 87
- in Contacts API, 111, 116
- currency settings and, 207, 220
- dates and, 209–217
- in downloading files, 178

- in File API, 157, 160, 163–164
- in Geolocation API, 182, 190, 192
- in Globalization API, 204–205
- in InAppBrowser API, 237–238
- language options and, 217
- locale names and, 218
- in Media API, 242–243, 257
- in Notification API, 267–268, 272
- number patterns and, 219–223
- in reading files, 170
- in uploading files, 175–176
- in writing files, 168–169

Supported platforms, 5**suppressesIncrementalRendering, 229**

symbol, 219

_system, 227

T

Tactile notifications, 275**Taking photographs, 36–44**

target parameters, in InAppBrowser, 227, 234–235

targetHeight/targetWidth

introduction to, 45

reducing image file size with, 50

specification of, 57–58

Temporary storage, 153–156, 162**Time-based watches, 86–93**

timeout, 182, 198

timezone, 209

Topcoat

CSS and, 12–14, 24

introduction to, 12–14

jQuery Mobile vs., 67

Transferring files

in File API, 174–179

file downloads in, 178–179

file uploads in, 175–178

transitionstyle, **229**

Troubleshooting

Camera API, 58–59

Capture API, 77

Compass API, 95

Connection API, 103

Contacts API, 124–125

Device API, 133

Events API, 152

File API, 179–180

Geolocation API, 197–200

Globalization API, 223

InAppBrowser API, 239–240

Media API, 263

Notification API, 275–276

Splashscreen API, 284

True Heading, 85

U

Uls. See User interfaces

Unicode Technical Standard #35, 207

Uniform resource identifiers (URIs). See URIs (uniform resource identifiers)

Uniform resource locators (URLs). See URLs (uniform resource locators)

updateStatus, **102**

updateUI **callbacks, 257**

Uploading files, 175–178

URIs (uniform resource identifiers)

in Camera API, 35

in configuring camera options, 45

as destination type, 48–51

in file downloads, 178–179

in file uploads, 175

for Media objects, 242–243, 247

in selecting photos, 54

in taking pictures, 38–44

URLs (uniform resource locators)

in Camera API, 40, 48–51

in Contacts API, 107–108

in File API, 171, 175, 178

in InAppBrowser API, 227, 231, 234, 236

U.S. locales, 209–211

User input, 269–274

User interfaces (UIs), 2, 110

UTC. See Coordinated Universal Time

V

Vibrate mode, 265–266

vibration **capabilities, 275**

Video

Camera API for, 51

Capture API for, 61–62, 76

Visual alerts, 266–269

Voice Recorder, 72–73

Volume

in Events API, 136, 145

in Media API, 248–249

W

W3C (World Wide Web Consortium)

Contacts API by, 105

File API, 153

Geolocation API Specification by, 182

Media Capture API by, 61

Watches

in Accelerometer API, 28–33

in Compass API, 86–95

degree-based, 93

device heading, 86–95

device orientation, 28–33

in Geolocation API, 189–197

location, 189–197

time-based, 86–93

Webviews, 2–5

Whitelists, 227, 255

Wi-Fi network connections, 98–99, 103. See also Connection API

Windows in InAppBrowser, 227–234

Windows Phone devices

Camera API on, 42–43, 55–58

Compass API on, 85

Device API on, 132

file details on, 166

Geolocation API on, 189, 198

InAppBrowser API on, 233

persistent storage locations in, 162

persistent storage on, 162

Timeout Error messages on, 198

WinMerge, 123

World Wide Web Consortium (W3C). See W3C (World Wide Web Consortium)

Writing files, 167–170

www.cordovacookbook.com

See cordovacookbook.com

www.cordovaprogramming.com

on command-line interfaces, 7, 9

on console exposure, 27

X

Xcode, 27, 282

Y

year, 215