

MOBILE
PROGRAMMING
SERIES



APACHE CORDOVA 4 PROGRAMMING

JOHN M. WARGO

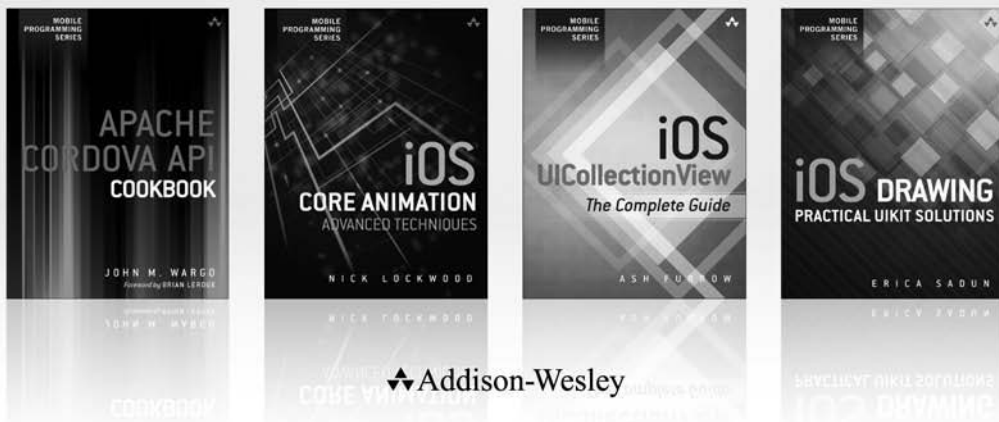
FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Apache Cordova 4 Programming

Addison-Wesley Mobile Programming Series



Visit informit.com/mobile for a complete list of available publications.

The Addison-Wesley Mobile Programming Series is a collection of programming guides that explore key mobile programming features and topics in-depth. The sample code in each title is downloadable and can be used in your own projects. Each topic is covered in as much detail as possible with plenty of visual examples, tips, and step-by-step instructions. When you complete one of these titles, you'll have all the information and code you will need to build that feature into your own mobile application.



Make sure to connect with us!
informit.com/socialconnect

informit.com
the trusted technology learning source

Addison-Wesley

Safari
Books Online

Apache Cordova 4 Programming

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 4 programming / John M Wargo.

pages cm

Includes index.

ISBN 978-0-13-404819-2 (pbk. : alk. paper)

1. Mobile computing—Computer programs. 2. Application program interfaces (Computer software)
3. Application software—Development. 4. Apache Cordova. I. Title.

QA76.59.W368 2015

004.167—dc23

2015003045

Screen captures © 2015 Adobe Systems Incorporated. All rights reserved. Adobe, PhoneGap and PhoneGap Build is/are either [a] registered trademark[s] or a trademark[s] of Adobe Systems Incorporated in the United States and/or other countries.

Apache, Apache Cordova, and the Cordova logo are trademarks of The Apache Software Foundation. Used with permission. No endorsement by The Apache Software Foundation is implied by the use of these marks.

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: 978-0-13-404819-2

ISBN-10: 0-13-404819-9

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, April 2015

*This is yet another book that couldn't exist except for
the unwavering support (and extreme patience)
of my wife, Anna. Crazy about you!*

*Not so much direct support, but still a lot of patience
from Elizabeth and August as well. I'm sorry I
wasn't able to sneak pictures of you guys
into the manuscript this time.*

This page intentionally left blank

Contents

Foreword xiii

Preface xv

Acknowledgments xxi

About the Author xxiii

1 The What, How, Why, and More of Apache Cordova 1

An Introduction to Apache Cordova 1

What Is Adobe PhoneGap? 3

A Little PhoneGap/Cordova History 4

Cordova Components 4

Access to Native APIs 5

Cordova User Interface Capabilities 10

Supported Platforms 12

Cordova License 13

Working with Cordova 13

 Designing for the Container 13

 Coding Cordova Applications 15

 Building Cordova Applications 16

Putting Cordova to Best Use 18

Getting Support 20

Resources 20

Cordova Going Forward 23

Hybrid Application Frameworks 25

Wrap-Up 25

2 Anatomy of a Cordova Application 27

Hello World! 27

Cordova Initialization 29

Leveraging Cordova APIs 35

Structuring Your Application's Code 38

The Generated Web Application Files 41

Responsive Design and Cordova 45

Wrap-Up 50

3	Configuring a Cordova Development Environment	51
	Installing the Cordova CLI	51
	Android Development Tools	52
	iOS Development Tools	63
	CLI Installation	65
	Installing Plugman	69
	Wrap-Up	70
4	Using the Cordova Command-Line Interfaces	71
	Troubleshooting	72
	Configuring Proxy Settings	72
	Enabling Verbose Output	74
	The Cordova CLI	75
	Cordova CLI Command Summary	76
	Using the Cordova CLI	76
	Upgrading Cordova and Cordova Projects	103
	The Plugman CLI	104
	Plugman CLI Command Summary	105
	Using the Plugman CLI	105
	Wrap-Up	120
5	The Mechanics of Cordova Development	121
	Cordova Development Issues	121
	Dealing with API Inconsistency	122
	Application Graphics, Splash Screens, and Icons	123
	Developing Cordova Applications	124
	Configuring a Cordova Application	131
	Testing Cordova Applications	134
	Leveraging Cordova Debugging Capabilities	135
	Using <code>alert()</code>	135
	Writing to the Console	136
	Debugging and Testing Using External Tools	139
	Weinre	139
	Ripple Emulator	145
	PhoneGap Developer App	148
	GapDebug	151
	Wrap-Up	156

- 6 Automation and the Cordova CLI 157**
 - Automating the Project Setup Step 157
 - Windows Command File 158
 - Bash Script 160
 - Cross-Platform Approach Using NodeJS 162
 - Automating the Cordova Process 164
 - Wrap-Up 167

- 7 Android Development with Cordova 169**
 - Using the Android Developer Tools 170
 - Managing the Android SDK 170
 - Using the Android Virtual Device Manager 172
 - Using the ADT IDE 178
 - Monitoring Application Activity Outside of the ADT IDE 191
 - Grabbing a Screen Shot 192
 - Testing on a Physical Device 192
 - Using the Chrome Debugging Tools 195
 - Wrap-Up 202

- 8 Firefox OS Development with Cordova 203**
 - Firefox OS Developer Tools 203
 - Debugging with the Firefox OS Simulator 207
 - Debugging Applications on a Firefox OS Device 218
 - Wrap-Up 220

- 9 iOS Development with Cordova 221**
 - Working with Xcode 221
 - Testing Cordova Applications in Xcode 225
 - Using the Safari Web Inspector 227
 - Wrap-Up 233

- 10 Ubuntu Development with Cordova 235**
 - Installing the Cordova CLI on Ubuntu 235
 - Debugging Ubuntu Applications 237
 - Wrap-Up 243

- 11 Windows Development with Cordova 245**
 - Windows versus WP8 Projects and Cordova 245
 - Windows Phone Limitations and Security Restrictions 247
 - JavaScript alert Not Supported 247
 - Application Security Model Limitations 248
 - Windows Development System Requirements 249
 - Windows Phone Development Tools 250
 - Windows App Store Setup 251
 - Configuring a Windows Phone Device for Application Testing 251
 - Cordova Development Workflow Using Visual Studio 254
 - Creating a Project 254
 - Opening a Cordova Project 256
 - Running a Cordova Application in Visual Studio 258
 - Controlling the Windows Phone Emulator 259
 - Debugging Cordova Applications Using Visual Studio 262
 - Using Visual Studio Tools for Apache Cordova 265
 - Wrap-Up 281

- 12 Using PhoneGap Build 283**
 - What Is PhoneGap Build? 283
 - Quick Prototyping 284
 - Collaboration 285
 - Content Refresh through Hydration 285
 - Using PhoneGap Build 286
 - A Quick Example 286
 - Configuring a PhoneGap Build Application 294
 - Adding Plugins to a PhoneGap Build Project 301
 - Deploying PhoneGap Build Applications 302
 - Wrap-Up 306

- 13 Using the PhoneGap CLI 307**
 - Getting Help 308
 - Project Management 309
 - Anatomy of the Default PhoneGap Application 310
 - PhoneGap CLI Workflow Differences 312
 - Interacting with the PhoneGap Build Service 312
 - Wrap-Up 315

14	Working with the Cordova APIs	317
	The Cordova Core APIs	317
	Working with the Cordova API Documentation	319
	Checking API Availability	320
	Catching Errors	321
	Setting Application Permissions	322
	Cordova Objects	324
	Connection Type	324
	device	326
	Alerting the User	326
	Hardware Notifications	326
	Visual Notifications	327
	Cordova Events	332
	Hardware APIs	334
	Accelerometer	335
	Compass	337
	Geolocation	339
	Camera	340
	Capturing Media Files	345
	Globalization	347
	Working with the Contacts Application	352
	Playing/Recording Media Files	358
	InAppBrowser	359
	Loading Content	360
	Browser Window Events	363
	Execute Scripts	364
	Insert CSS	365
	Splashscreen	367
	StatusBar	367
	Wrap-Up	371
15	Cordova Development End to End	373
	About the Application	373
	Creating the Application	374
	Using Merges	385
	Application Icons	387
	Testing the Application	389
	Wrap-Up	396

16	Creating Cordova Plugins	397
	Anatomy of a Cordova Plugin	397
	Creating a JavaScript-Only Plugin	398
	plugin.xml File	399
	The Plugin's <code>main.js</code> File	401
	Testing the Plugin	403
	Creating a Cordova Native Plugin	408
	Creating the Android Plugin	414
	Creating the iOS Plugin	424
	Publishing Plugins	431
	Wrap-Up	435
17	Using Third-Party UI Frameworks with Cordova	437
	Adobe Topcoat	439
	jQuery Mobile	444
	Bootstrap	450
	SAP OpenUI5	456
	Ionic Framework	459
	Onsen UI	464
	Wrap-Up	468
18	Using Third-Party Tools with Cordova	469
	Code Validation Tools	469
	JSLint	470
	JSHint	471
	Code Editors	473
	Adobe Brackets	473
	WebStorm	479
	Developer Productivity Enhancement Tools	485
	AppGyver	486
	Eclipse THyM	490
	Build Tools	494
	Gulp	494
	Grunt	500
	Wrap-Up	503
	Index	505

Foreword

It's great to have John Wargo in my classroom and in my teaching, both literally and figuratively!

Apache Cordova 4 Programming (AC4P) will be the fourth John Wargo title employed in my classroom. Surprisingly, this frenetic iteration happens in just two semesters and is testimony to the value of John's work.

The figurative: In preparing for my college's first offering of an upper-level mobile application course, it became evident that I should cover both native and hybrid mobile technologies. For a hybrid technology, PhoneGap immediately rose to the top of my candidate list. The search was on for a meaningful text. A survey of potential materials revealed that no formal college textbooks existed, and that John's *PhoneGap Essentials* was the de facto primer for technology professionals looking to learn about PhoneGap. Perfect, right? I order and review a copy, confirm the book's reputation, and place an order with my college bookstore.

Enter John Wargo. I engage John to explore the possibility of acquiring any supporting materials, the true value-add of a text or reference in any fast-paced course like this. John offers to assist but also immediately cautions that my choice of text is dated and that two newer texts now replace the first. I also learn that a fourth text is in the works [unhappy emoji]. Interactions with the college bookstore and publisher ensue, and the adjustments for text numbers two and three are made.

I'll spare you the unnecessary detail, but fast-forward to today. I anxiously await AC4P for inclusion in my course, later this semester.

Ah, I haven't yet shared the literal connection. Recalling my interactions with John, I add this anecdote. In addition to his assistance with the texts, John agrees to visit my campus when traveling to our area. He offers to visit my class as well as a college-wide venue to speak and to interact with our students. (We have a population of more than a thousand information technology students). What happened was marvelous; his words come off the pages and into life in these forums. This provides a tremendous learning opportunity for Georgia Gwinnett College's students. Conversely, we see that the narratives provided in print are his knowledge and experience captured in prose. My students engaged enthusiastically, commenting that we should do much more with PhoneGap (and its open-source cousin, Cordova) in future semesters. They were hooked!

Again, I welcome John into my classroom figuratively and hope that we can host him literally again, too.

—Bob Lutz, Ph.D.
Georgia Gwinnett College
January 2015

This page intentionally left blank

Preface

This is a book about Apache Cordova, the leading framework for building native mobile applications for multiple target platforms using HTML5 (HTML, JavaScript, and CSS). I created the book in order to help web developers and mobile developers understand how to use Apache Cordova to build hybrid applications for mobile devices. The book targets the specific capabilities provided in Apache Cordova 4 and subsequent versions.

As Adobe PhoneGap is just a distribution of Apache Cordova, this book is also about Adobe PhoneGap. You'll find any differences between the two clearly described herein.

The book is written for mobile developers who want to learn about Apache Cordova 4. If you're brand-new to Cordova, this book will be just what you need to get started. If you're experienced with an older version of Cordova, this book can act as a refresher, plus it will show you in detail how to use all of the new stuff that's in Cordova 4. You should have at least some experience with mobile development to directly benefit from this book. For web developers who want to get into mobile development using Apache Cordova, I've included content that shows you how to install and use the native SDKs, but I won't cover many native-specific topics.

What you'll find in the book:

- Lots of detailed information about Apache Cordova, what it does, how it works, and how to use the available tools and APIs
- Lots of examples and code; for even more code, be sure to check out my *Apache Cordova API Cookbook* (www.cordovacookbook.com)

What you won't find in this book:

- Mobile web development and mobile development topics; this is a book about Apache Cordova, not mobile development
- Expressions or phrases in languages other than English (I hate it when authors include expressions from Latin or French)
- Obscure references to pop-culture topics (although there is an overt reference to Douglas Adams's *Hitchhiker's Guide to the Galaxy* and one obscure reference to Monty Python)
- Pictures of my children or my pets

This is not a book for experienced Cordova 4 developers—if you consider yourself an experienced Cordova 4 developer, you probably should not buy this book.

Herein I try to provide complete coverage of Apache Cordova 4, covering enough detail that readers will leave with a complete understanding of what Cordova is, what it does, how it works, and how to use it for their mobile application projects. There's a whole lot more to Cordova—many advanced topics and more detailed coverage of the Cordova APIs, which can be found in the Cordova documentation or in blogs.

This book started many years ago as a book called *PhoneGap Essentials* (www.phonegapessentials.com); the book was all about PhoneGap 2.0 and was published right about the time the project name changed to Apache Cordova. The book came in at about 300 pages. The book's first 150 pages covered the available tools and everything a developer needed to know to configure a development environment, and then create, write, build, and test PhoneGap applications. The second half of the book provided a detailed deep dive into each of the (at the time) PhoneGap APIs. The cool part of this second half was that for each API it included at least one complete, functional sample application that demonstrated each aspect of the API. The framework's documentation was pretty useful in demonstrating how the API worked overall, but *PhoneGap Essentials* provided much more thorough examples.

The book went on to become the best-selling book on the topic, and it was used in university courses around the world. According to Amazon.com, people are still purchasing this book today.

With the release of Apache Cordova 3, I reworked the manuscript and published *Apache Cordova 3 Programming* (www.cordovaprogramming.com). This book also came in at 300 pages but was essentially a rewrite of just the first half of *PhoneGap Essentials* with only cursory coverage of the Cordova APIs provided. This allowed me to go into much more detail on the tools and development process.

Unfortunately, because *Apache Cordova 3 Programming* was available only as an ebook, it was hard to find, and many readers continued to buy *PhoneGap Essentials* even though it covered an older version of the framework.

In order to accommodate those readers who were more interested in the Cordova APIs, I reworked the second half of *PhoneGap Essentials* into another 300 pages called *Apache Cordova API Cookbook* (www.cordovacookbook.com). In this book, the complete example applications from *PhoneGap Essentials* were enhanced and expanded, and all of the book's content was updated for the newer version of Cordova. I'd not covered some topics as well as I would have liked to in the first book, so this update allowed me to really expand the coverage of some topics and include even more complete sample applications (32, I think it was).

Between *Apache Cordova 3 Programming* and *Apache Cordova API Cookbook*, I had written more than 600 pages of coverage of Apache Cordova 3. That's more than twice the size of the original book and a lot of good information for developers.

With this book, I've updated *Apache Cordova 3 Programming* for Apache Cordova 4, plus included new content on a bunch of topics. In my previous books, I focused primarily on PhoneGap and Apache Cordova; I didn't cover many third-party tools and left many mobile development topics uncovered as well. For this book, there were a bevy of additional tools available and some hybrid-focused HTML frameworks, so I decided to cover as many of them as I could in the space

available to me. Where this book's predecessor was 300 pages, this one should top out at more than 500 pages, so there's a lot of really good information here for all types of Cordova developers. When bundled with *Apache Cordova API Cookbook*, you'll have more than 800 pages of information about Apache Cordova.

Herein you'll find most of the same topics that were covered in *Apache Cordova 3 Programming*. The only missing topic is coverage of the BlackBerry platform. I wrote the first book on BlackBerry development and had pretty much always carried a BlackBerry device, but between books, BlackBerry experienced a dramatic drop in market share and I started carrying an Android device as my primary device. Additionally, in previous books I had the enthusiastic support of my former colleagues at BlackBerry, but when it came time to get feedback on the BlackBerry chapter in *Apache Cordova 3 Programming*, the development team stopped responding to my inquiries. Because of those two things I decided to drop support for BlackBerry from this book.

So, what new stuff have I added in this book? Coverage of

- Plugman and the PhoneGap CLI
- Cordova's support for Firefox OS and Ubuntu devices
- Automation (Grunt and Gulp) and Cordova CLI hooks
- Microsoft's hybrid toolkit for Visual Studio
- Third-party tools such as AppGyver, GapDebug, THyM, and more
- Third-party HTML frameworks such as Bootstrap, OpenUI5, Ionic, and Onsen UI

There's a lot more, but these are some of the highlights.

The one thing I cover in the book but not in tremendous detail is how to build custom Cordova plugins. I cover the topic and show you how to create two complete plugins, but this isn't a native mobile development book and that's a native mobile development topic. I've learned from my readers that the material I do provide is enough to help a lot of people get started with plugins and create their own plugins; I'll leave it up to another author to write a book dedicated to plugin development so it can get the attention it deserves.

Android Studio versus Android Developer Tools (ADT)

As I wrote the previous edition of this book, Google announced a new development tool called Android Studio. I expected then that Android Studio would be out before I started this manuscript and I'd be able to update the content for the new tool. As I worked through the book, Android Studio was still in beta and it was essentially incompatible with Cordova CLI-generated projects. I thought about hacking through it in order to provide updated content here, but after discussing my situation with Andrew Grieve from Google, I decided that it wasn't yet ready for prime time and I would stick with ADT for this book.

Wouldn't you know it, right after the book went into the editing process, Google finally released Android Studio. Sigh. At this point, I could make minor changes to the manuscript but couldn't rewrite a complete chapter. So, unfortunately, some of the content you'll find in Chapter 7, "Android Development with Cordova," refers to the older version of the SDK. The stuff around the SDK is still valid, but Android Studio installs everything in a different place from what I've shown. The incompatible stuff is everything I showed about using the Eclipse tools. Sorry.

University Use

One of the pleasant surprises you have when delivering technical books is when a book is picked up for use in university courses. From what I can tell, several universities around the world use my Cordova books for their PhoneGap/Cordova class work. I regularly receive emails from university professors asking me questions about the book as they prepare to use it in their classes.

I was fortunate enough to hear from Dr. Robert Lutz from Georgia Gwinnett College. They were using my books (*Apache Cordova 3 Programming* and *Apache Cordova API Cookbook*) in class and they were close enough that I could drive there and see how it was working for them. I arranged a visit to the campus, and Dr. Lutz was kind enough to arrange a campus Tech Talk at the university. I spent about an hour talking about mobile development and the role hybrid applications play in the market. After the session ended, I spent some more time with the class using my book and let the students pick my brain on a number of topics. It was quite a lot of fun and allowed me to learn more about how my work is being used by others. I even signed a few copies of my books.

After this book is finished, my goal is to work with Dr. Lutz to prepare classroom material that can be used in conjunction with the book. Stay tuned on that one.

Cordova as a Moving Target

One of the challenges in writing a book about open-source projects is that if the project is well staffed and busy, it gets regular updates. In Cordova's case, it's one of the fastest-moving open-source projects on the planet, so with their regular updates and yearly major releases, it is definitely a moving target.

I've worked very hard to structure and craft this book so that it can survive the rapid pace of the project, but only time will tell. You may find that something I've written here has changed and the book doesn't align with reality. There's nothing I can do about this except to stay on top of it and post updates to the book's web site (described below) when I find that something has changed enough that it breaks part of the book.

A Comment on Source Code

One of the things you'll notice as you look at the source code included in the book is that I've paid special attention to the formatting of the code so that it can be easily read and understood by the reader. Rather than allowing the publisher to wrap the source code wherever necessary, instead I've forced page breaks in the code wherever possible in order to structure it in a way that should benefit the reader. Because of this, as you copy the source code over into your Cordova applications, you will likely find some extra line breaks that affect the functionality of the code. Sorry.

All of the book's source code is available on GitHub (<https://github.com/johnwargo/ac4p>); there you'll find the complete application source code in a format that will allow you to quickly copy the code into your apps.

The Book's Web Site

The book has its own web site at www.cordova4programming.com. I will post there any updates and errata to the book. I'll also answer questions I receive from readers. Please feel free to use the contact form on the book's web site to provide feedback and/or suggestions for the next edition as well.

This page intentionally left blank

Acknowledgments

This book wouldn't exist without the help of others; as you can see below, I had a lot of help. Thank you to everyone who helped me craft this manuscript, including:

- The Cordova dev team for their patience and support as I asked all of my silly questions.
- Raman Sethi, Bobby Anchanattu, Changhoon Baek, Rob Close, Ashwin Desai, Alan Kinzie, Pete Kong, Jonathan Li, Marcus Pridham, Dan Van Leeuwen, and my other colleagues at SAP for continuing to teach me new things about Apache Cordova every day.
- Colleagues Marcus Pridham and Ashwin Desai for again helping me sort out issues with the plugin examples used in the book.
- Brian LeRoux, Steven Gill, Dave Johnson, and Michael Brooks from Adobe for helping me through some issues and reviewing the PhoneGap-related chapters.
- Andrew Grieve from Google for helping me decide whether to cover ADT or Android Studio in this version of the book. Turns out I made the wrong choice, but there's not much I can do about that now.
- Olivier Block, Eric Mittelette, Parashuram Narasimhan, and Sergey Grebnov from Microsoft for helping me through some issues, reviewing the Windows chapter, and providing me with a device to use for testing applications.
- Piotr Zalewa from Mozilla for answering my questions and reviewing the Firefox OS chapter.
- Gorkem Ercan from Red Hat for getting me started with THyM.
- David Pitkin, David Barth, Maxim Ermilov, and Jean-François Moy from Ubuntu for answering my questions, reviewing the Ubuntu chapter, and providing me with a device to use for application testing.
- Ashwin Desai for doing yet another excellent technical review of the manuscript. You've got to love it when the tech reviewer treats the book like his own work and even makes sure that the comments in the sample source code are correct.
- Greg Doench, Michelle Housley, and Chris Zahn for all their help with this project.
- Julie Nahil, Susan Brown Zahn, Anna Popick, and Barbara Wood for their help producing the book.
- My managers at SAP for supporting this endeavor.

Apologies to anyone I may have missed here.

This page intentionally left blank

About the Author

John M. Wargo is a professional software developer and a contributor to the Apache Cordova Project. John works for German software company SAP as part of the SAP Mobile Platform Product Management team. He is the product manager for the SAP Mobile Platform's Hybrid SDK, a set of enterprise plugins for Apache Cordova, and the SAP Fiori Client, a Cordova-based native application runtime for the SAP Fiori web application. He also works with the team at SAP building the SAP Web IDE Hybrid App Toolkit (HAT), a set of tools that add support for Apache Cordova applications to the SAP Web IDE (a cloud-based web application designer and editor based on the Apache Orion project).

This is his fourth book on Apache Cordova. He got started with mobile development while working at Research In Motion, now called BlackBerry, and eventually wrote the first book on BlackBerry development. He wrote a series of articles covering methods for mobilizing IBM Domino applications for *The View*, a magazine for IBM Lotus Notes and Domino developers, which was eventually published into an anthology.

You can find him online at www.johnwargo.com and on Twitter at [@johnwargo](https://twitter.com/johnwargo).

This page intentionally left blank

This page intentionally left blank

Anatomy of a Cordova Application

In the previous chapter, I provided you with an overview of Apache Cordova; before I start digging into all of the tools, capabilities, APIs, and so on, I want to give you a clear definition of what a Cordova application is. In this chapter, I show you what makes a web application a Cordova application and give you a tour of the sample application the Cordova team provides.

As mentioned at the beginning of the book, a Cordova application can do anything that can be coded in standard, everyday HTML, CSS, and JavaScript. There are web applications and Cordova applications, and the distinctions between them can be minor or can be considerable.

The sections in this chapter highlight different versions of the requisite HelloWorld application found in most any developer book, article, or training class. For the purpose of highlighting aspects of the applications' web content, rather than how they were created, the steps required to create the applications are omitted here (but covered in subsequent chapters).

Hello World!

As in any developer book, we're going to start with the default HelloWorld application, then build upon it to highlight different aspects of what makes a web application into a Cordova application. The HTML content shown in Listing 2.1 describes a very simple web page that displays some text; this application could easily run in a desktop or mobile browser.

Listing 2.1 Hello World #1 Application

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Hello World #1</title>
</head>
<body>
```

```

<h1>Hello World #1</h1>
<p>This is a sample Apache Cordova application.</p>
</body>
</html>

```

If you open the web page in the mobile browser on a physical device or on a device emulator or simulator, you will see something similar to what is shown in Figure 2.1 (here it's running in an Android emulator). The browser simply renders the page the best it knows how to, in this case, trying to render it as if it's a full web page scaled to fit the smaller screen of the mobile device. Since it's the browser, the window also displays the browser chrome, the address field, tab controls, and so on from the mobile browser.

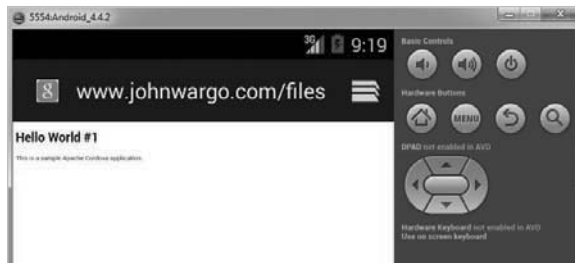


Figure 2.1 Hello World #1 Application Running in the Mobile Browser on an Android Emulator

This is not a Cordova application; it's just a web application running in a mobile browser.

If you package that same index.html file into a Cordova application (using the tools I will discuss throughout the book) and run it on a smartphone device or device emulator, the app will display something similar to what is shown in Figure 2.2.

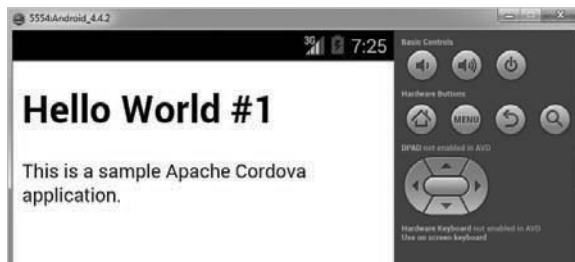


Figure 2.2 Hello World #1 Application Running on an Android Emulator

Here, the container seems to understand a bit about the view it's being rendered within and renders full size, not scaled down, so the whole page fits within the browser window.

In this example, this is a Cordova application because the web application has been packaged into the Cordova native application container. If I hadn't cropped the image, you would see that the web application consumes the entire screen of the emulated Android device. Even though

I'm running a web application, because it's running within a native application, there's no browser UI being displayed and no access to browser features. It's simply a native application rendering web content.

There is, however, nothing Cordova-ish about this application. It's running in the Cordova native container, but it isn't leveraging any of the APIs provided with the Cordova framework. Therefore, any web application can be packaged into a Cordova application—there's nothing forcing you to use the Cordova APIs. If you have a simple web application that simply needs a way to be deployed through a smartphone's native app store, for example, using Cordova is one way to accomplish that goal.

However, the app's not very interesting, is it? It's certainly not very pretty, but I'll show you how to fix that in Chapter 17, "Using Third-Party UI Frameworks with Cordova." For me, it needs to do some cool Cordova stuff before it becomes interesting.

Cordova Initialization

Now let's take the previous example application and add some Cordova-specific stuff to it.

Even though the Cordova container exposes native APIs to the web application running within it, in general (there are a few exceptions) those APIs are not available until the plugin that exposes the API has been added to the project. Additionally, the Cordova container has to do some prep work before any of its APIs can be utilized. To make it easy for developers to know when they can start using APIs exposed by the Cordova container, Cordova fires a specific event, the `deviceready` event, once it has finished its initialization and it's ready to go. Any application processing that requires the use of the Cordova APIs should be executed by the application only after it has received its notification that the Cordova container is available through the `deviceready` event.

The Hello World #2 application shown in Listing 2.2 has been updated to include code that uses a `deviceready` event listener to determine when the Cordova container application has completed its initialization. In this simple example, the application just displays an alert dialog when the event fires.

Listing 2.2 Hello World #2 Application

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World #2</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1,
    maximum-scale=1, minimum-scale=1, width=device-width,
    height=device-height" />
  <script src="cordova.js"></script>
  <script>
```

```

function onBodyLoad() {
    console.log("Entering onBodyLoad");
    alert("Body Load");
    document.addEventListener("deviceready", onDeviceReady, false);
}

function onDeviceReady() {
    console.log("Cordova is ready");
    navigator.notification.alert("Cordova is ready!");
}

</script>
</head>
<body onload="onBodyLoad()">
    <h1>Hello World #2</h1>
    <p>This is a sample Cordova application.</p>
</body>
</html>

```

Warning

If you copy the code from any of the listings in this chapter and try them in your own Cordova applications, you may notice that there are some extra carriage returns in the middle of some of the HTML. This was done to make the code render cleanly in the printed edition of the book. To download clean versions of all of the projects in this book, access the Code section of the book's web site at www.cordova4programming.com or get them from GitHub at <https://github.com/johnwargo/ac4p>.

On the iPhone simulator, the application will display the screen shown in Figure 2.3.



Figure 2.3 Hello World #2 Application Running on an iOS Simulator

Let's take a look at the sample application as there's a lot of new stuff in this example.

Within the `<head>` section of the web page are a few new entries, some meta tags that describe the content type for the application, and some other settings. For the most part, I pulled these meta tags from the default Cordova HelloCordova application described later in the chapter.

The `charset` tag identifies the character encoding used for the HTML document. What I've shown here is the default option; you would change this only if you were using a different character set for the HTML page.

```
<meta charset="utf-8" />
```

The next tag disables the embedded web browser's automatic processing of telephone numbers. With this option disabled, as shown below, the browser won't automatically turn phone numbers on the page into clickable links. You would need to change `telephone=no` to `telephone=yes` to enable this option.

```
<meta name="format-detection" content="telephone=no" />
```

Honestly, I'm really not sure why the Cordova team did this in their sample application; you would probably assume the user was running the application on a smartphone and would want phone numbers to be automatically enabled as links.

The viewport settings shown in the following 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" content="user-scalable=no, initial-scale=1,
  maximum-scale=1, minimum-scale=1, width=device-width,
  height=device-height" />
```

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're 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 Hello World #2 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 src="cordova.js"></script>
```

This loads the core Cordova API library and makes any core Cordova APIs available to the program. This file is also responsible for loading and initializing all of the plugins you have added to your Cordova application. You don't have to add the `cordova.js` file to your project; this is done for you automatically by the Cordova CLI (described in Chapter 4, "Using the Cordova Command-Line Interfaces"), but you do need to add this reference to your application.

To set up the `deviceready` event listener we need for Cordova, the application adds an `onload` event function to the application's body tag using 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 Cordova container is ready, when the Cordova application container has finished its initialization routines and fired its `deviceready` event:

```
function onBodyLoad() {
    document.addEventListener("deviceready", onDeviceReady, false);
}
```

In this example application the `onDeviceReady` function simply displays a Cordova alert dialog (which is different from a JavaScript alert dialog) letting the user know everything's OK:

```
navigator.notification.alert("Cordova is ready!");
```

In production applications this function could update the UI with content created through API calls or do whatever other processing is required by the application. (You'll see an example of this in Listing 2.4.)

Note

Cordova applications fail silently when they encounter typos or syntax errors in a web application's code, so when you're testing an application, sometimes nothing will happen and you'll have no clue why. If you look at the complete source code for the application, you'll notice that there are a few things I haven't described yet that I do in every Cordova application I write to help me troubleshoot the application. These tricks help me more quickly understand what's happening in an application as it runs.

One of the things I do during testing is use the web browser console to display status messages as the application runs using code similar to the following:

```
console.log("Entering onBodyLoad");
```

I'll show you how this works in Chapter 5, "The Mechanics of Cordova Development."

In the `onBodyLoad` function, I also make sure to make a call to the JavaScript `alert` function so I can easily tell that the `onload` event has fired:

```
alert("Body Load");
```


Note

Unfortunately, the JavaScript `alert()` function is not available in universal Windows apps, so you will have to adjust your code when running on that platform. This topic is discussed further in Chapter 11, “Windows Development with Cordova.”

As I mentioned earlier, the Cordova container fails silently when it encounters an error with the web application’s source code. So, if I have this alert in the code and it doesn’t fire, I know very quickly (in the very first code the application executes) that something is wrong with the application.

In the `deviceready` event handler, I always add a call to `navigator.notification.alert` as shown in the example code. This allows me to confirm visually that the `deviceready` event has actually fired, plus it allows me to confirm that the Cordova Dialogs plugin has been added to the project and that any other debug alerts I put into the code will be operational. I use the Cordova `alert` instead of the JavaScript `alert` because it’s better looking (I can set the title of the dialog, for example, although I didn’t do that here); it also gives me access to callback functions I can use to perform extra steps when something interesting happens.

Remember, most of the Cordova APIs have been removed from the container and implemented as plugins. So, to utilize the Cordova `alert` method, you must add the Dialogs plugin to your application by opening a terminal window to your Cordova project folder and issuing the following command:

```
cordova plugin add org.apache.cordova.dialogs
```

You’ll learn all about how to use the `cordova` command in Chapter 4. You’ll learn more about the Dialogs plugin in Chapter 14, “Working with the Cordova APIs.”

The Cordova Navigator

Many of the APIs implemented by Cordova are instantiated from the Navigator object. Unfortunately it’s not consistent; some APIs do it that way and some do not. Be sure to check the API documentation before calling an API.

The `deviceready` event will fire when the Cordova container finishes initializing, but it will also fire any time a new `deviceready` event listener is added by the application. Listing 2.3 shows this in action.

Listing 2.3 Hello World #3 Application

```
<!DOCTYPE html>
<html>
<head>
```

```

<title>Hello World #3</title>
<meta charset="utf-8" />
<meta name="format-detection" content="telephone=no" />
<meta name="viewport" content="user-scalable=no, initial-scale=1,
  maximum-scale=1, minimum-scale=1, width=device-width,
  height=device-height" />
<script src="cordova.js"></script>
<script>

  function onBodyLoad() {
    console.log("Entering onBodyLoad");
    alert("Body Load");
    document.addEventListener("deviceready", onDeviceReady, false);
  }

  function onDeviceReady() {
    console.log("Entering onDeviceReady");
    navigator.notification.alert("Cordova is ready!");
  }

  function addSecondDeviceReadyListener() {
    console.log("Entering addSecondDeviceReadyListener");
    document.addEventListener("deviceready", someOtherFunction, false);
  }

  function someOtherFunction() {
    console.log("Entering someOtherFunction");
    navigator.notification.alert("Second deviceready Function Fired.");
  }

</script>
</head>
<body onload="onBodyLoad()">
  <h1>Hello World #3</h1>
  <p>This is a sample Cordova application.</p>
  <button onclick="addSecondDeviceReadyListener()">Add deviceready Event Listener</
button>
</body>
</html>

```

In this example, I've added a button to the application's main page. When the button is tapped, an additional deviceready event listener is defined, and then the callback function for the new listener is immediately executed by the Cordova container. In this case, the onDeviceReady function executes once the container completes its initialization, and then the someOtherFunction function executes only after the second deviceready event listener has been added.

Leveraging Cordova APIs

Now that we know how to configure an application to wait until the Cordova APIs are available, let's build an application that actually uses some of the Cordova APIs. The Hello World #4 application shown in Listing 2.4 uses the Cordova Device API to allow the application to understand a bit about the environment it is running in.

Listing 2.4 Hello World #4 Application

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World #4</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1,
    maximum-scale=1, minimum-scale=1, width=device-width,
    height=device-height" />
  <script src="cordova.js"></script>
  <script>
    var br = "<br />";

    function onBodyLoad() {
      console.log("Entering onBodyLoad");
      alert("Body Load");
      document.addEventListener("deviceready", onDeviceReady, false);
    }

    function onDeviceReady() {
      navigator.notification.alert("Cordova is ready!");
      console.log("Cordova: " + device.cordova);
      //Get the appInfo DOM element
      var element = document.getElementById('appInfo');
      //replace it with specific information about the device
      //running the application
      element.innerHTML =
        'Cordova Version: ' + device.cordova + br +
        'Platform: ' + device.platform + br +
        'Model: ' + device.model + br +
        'OS Version ' + device.version;
    }

  </script>
</head>
<body onload="onBodyLoad()">
  <h1>Hello World #4</h1>
```

```

<p>This is a Cordova application that makes calls to the Cordova Device API.</p>
<p id="appInfo">Waiting for Cordova Initialization to complete.</p>
</body>
</html>

```

Figure 2.4 shows the Hello World #4 application running on the Windows Phone 8.1 simulator.



Figure 2.4 Hello World #4 Application Running on a Windows Phone Simulator

In this version of the HelloWorld application, the code in the `onDeviceReady` function has been updated so the program updates a portion of the application's content with an ID of `appInfo` with information about the device running the application and the version of Cordova used to build the application. Device-specific information is available via the Cordova Device API (<http://plugins.cordova.io/#/package/org.apache.cordova.device>), and this sample application uses a subset of the available properties in this API.

In order for me to be able to call the Device API, I had to add the Device API plugin to the project using the CLI command:

```
cordova plugin add org.apache.cordova.device
```

Note

Remember, Cordova fails silently when it encounters an error in a web application's code. So, if you forget to add the plugin to your application, the code will seem to execute, but nothing will happen. I can't tell you how many times I've tried to use the Device API's methods only to see them not work because I simply forgot to add the plugin to the project.

With the Device API in place, the application can access it using the following code:

```
var element = document.getElementById('appInfo');
element.innerHTML = 'Cordova Version: ' + device.cordova + br +
  'Platform: ' + device.platform + br +
  'Model: ' + device.model + br +
  'OS Version ' + device.version;
```

In the figure, you may have noticed that the Cordova version shows that I'm running Cordova 3.6.4. I actually ran this application using Cordova 4.0, but with this release the Cordova CLI, Cordova container, and Cordova APIs have all been broken out into separate releases. So, even though I'm actually running Cordova 4.0, some of the components may be at a different release.

Listing 2.5 shows a slightly modified version of the application; in this case I added some markup to make the device information into an unordered list so it would render more neatly on the page.

Listing 2.5 Hello World #5 Application

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World #5</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1,
    maximum-scale=1, minimum-scale=1, width=device-width,
    height=device-height" />
  <script src="cordova.js"></script>
  <script>
    function onBodyLoad() {
      alert("Body Load");
      document.addEventListener("deviceready", onDeviceReady, false);
    }

    function onDeviceReady() {
      navigator.notification.alert("Cordova is ready!");
      console.log("Cordova: " + device.cordova);
      //Get the appInfo DOM element
      var element = document.getElementById('appInfo');
      //replace it with specific information about the device
      //running the application
      element.innerHTML =
        '<ul><li>Cordova Version: ' + device.cordova +
        '</li><li>Platform: ' + device.platform +
        '</li><li>Model: ' + device.model +
```

```

        '</li><li>OS Version ' + device.version + '</li></ul>';
    }
</script>
</head>

<body onload="onBodyLoad()">
    <h1>Hello World #5</h1>
    <p>This is a Cordova application that makes calls to the Cordova Device API.</p>
    <p id="appInfo">Waiting for Cordova Initialization to complete.</p>
</body>
</html>

```

Just so you can see a Cordova application running on yet another device, Figure 2.5 shows the Hello World #5 application running on a Firefox OS simulator.

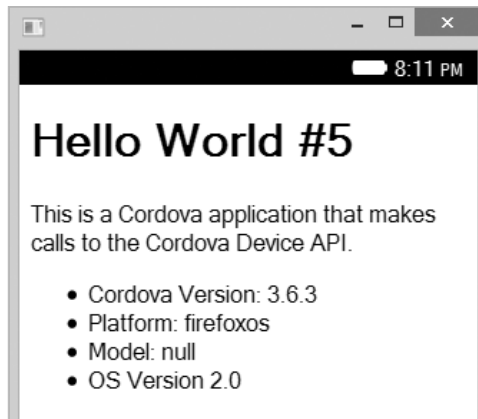


Figure 2.5 Hello World #5 Application Running on a Firefox OS Simulator

Structuring Your Application's Code

The way you structure the code for your web application is a matter of personal style, but for Cordova applications, and for some web applications, there may be a valid reason to use a particular approach. So far in this chapter I've set up my example applications so that everything, the HTML content as well as the JavaScript code, is in the same file. Additionally, I've broken things up a bit so the examples are simple and easy to read. There are a lot of things a developer can do to write more efficient and compact code—here I've deliberately not done them to make the examples as easy to read as possible.

A web developer will usually want to separate an application's HTML from its JavaScript code. In the simple applications I've shown here, there's not much of each, so it's not a big deal. But for more complicated applications, when there's a whole lot of code, separation of the two types of

code can make the code easier to maintain and allow multiple developers to work on different parts of the application (UI versus application logic) separately.

There is, however, a Cordova-specific reason why you will likely want to do this. Remember how I explained earlier that the Cordova container needed to initialize itself? Well, if you think about an application that has several Cordova plugins added to it, it might take some time for the Cordova container to initialize itself, and for all of the plugins to initialize themselves as well. What I've found in many sophisticated Cordova applications is that large web applications and/or a bunch of plugins can cause a Cordova application to time out during initialization. It takes so long to load and initialize everything that the Cordova container thinks something's wrong and fails with a timeout error. I've seen this happen most frequently with a large web application using jQuery Mobile.

So, what do you do to avoid this? You structure your web application projects so that the web content and the JavaScript code are separated, and then you take some extra steps to arrange the order in which things happen.

Another reason why you would want an application's JavaScript code broken out into a separate file is to more easily support JavaScript debugging. Throughout the book I'll show you many different tools you can use to test and debug your Cordova applications. What I found in my testing of these tools is that most of them are able to interact with an application's JavaScript code only when the code is not embedded inside the application's HTML content (the application's `index.html` file, for example).

Listing 2.6 shows a simple application I've created that is structured a little differently from all of the other examples I've shown so far. In this example, two things are different: the application loads all of its JavaScript code after all of the application's HTML has been defined, plus all of the application's logic has been split out into a separate JavaScript file called `index.js`.

Listing 2.6 Hello World #6 Application `index.html`

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World #6</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1,
    maximum-scale=1, minimum-scale=1, width=device-width,
    height=device-height" />
</head>
<body>
  <header>
    <h1>Hello World #6</h1>
  </header>
  <p>This is a simple Cordova application.</p>
  <script src="cordova.js"></script>
```

```

    <script src="index.js"></script>
</body>

</html>

```

When the earlier example applications started up, the `cordova.js` file was loaded before much else happened on the page. If the `cordova.js` took a while to load, on a slower device, for example, it might delay the rendering of the page's HTML content while it waited for the JavaScript to load. So, users of the application might see a blank page before the HTML displayed. If this was a large application, and several JavaScript files were being loaded, this might take some time, enough that the user would notice.

In the Hello World #6 application, all of the HTML loads within the browser context before the `cordova.js` file is loaded. If the `index.js` file were quite large, or I was loading jQuery Mobile and a bunch of other JavaScript stuff, the user would at least be looking at some sort of UI as the JavaScript was being loaded.

Listing 2.7 shows the application's `index.js`. It contains all of the JavaScript code the application is using. In this example, the file defines a simple function that self-initializes when the file is loaded, adds the event listener for the `deviceready` event, and provides a function that is executed when the event fires.

Listing 2.7 Hello World #6 Application `index.js`

```

var cvaReady;

var someOtherFunction = function () {
    if (cvaReady) {
        //do something

    } else {
        //tell the user why they can't do that
    }
};

(function () {

    var onDeviceReady = function () {
        console.log("Entering onDeviceReady");
        //Let the user know that the deviceReady event has fired
        navigator.notification.alert("Cordova is ready", null,
            "Device Ready", "Continue");
        //Set the variable that lets other parts of the program
        //know that Cordova has initialized
        cvaReady = true;
    };

```



```

//=====
//Do whatever other stuff you want to do on startup
//=====

    console.log("Leaving onDeviceReady");
};

//add an event listener for the Cordova deviceReady event.
document.addEventListener('deviceready', onDeviceReady, false);

}());

```

I've added a new feature in this example as well, a `cvaReady` object that the application can use to tell whether the `onDeviceReady` function has executed. If you don't want to wait to do everything until the `deviceready` event has fired, you can ignore it and check the `cvaReady` object as needed to see if you are able to do Cordova stuff. I know this is a clunky way to do this; I'm just trying to give you different options for your applications.

When you run into an issue where the Cordova container times out before loading all of your stuff, what some people recommend doing is setting up a timer in your `deviceready` event listener that waits a few seconds before loading a new page that then loads your application's JavaScript files. This allows all of the Cordova initialization to complete before anything else is done by the application. This is supposedly one way people have gotten around timing issues with using jQuery Mobile with a large Cordova application, but I've never had the need to use this approach.

The Generated Web Application Files

Now that I've shown you how a Cordova application is crafted, let's take a look at the default application generated by the Cordova CLI. In Chapter 4 you'll see that when the CLI creates a new application project, by default it creates a simple HelloCordova web application and places it in the project's `www` folder. You can override this behavior if you want, but this is the default.

The project folder contains a web application folder structure that is designed to separate the different types of files into separate folders. For example, the web application's CSS files should be placed in the `css` folder, JavaScript files in the `js` folder, and so on.

The application's `index.html` file is shown in Listing 2.8; it contains many of the same HTML elements and attributes as the other examples shown throughout the chapter. What the application does is display a simple page with the Cordova logo and some blinking text, "Connecting to Device," centered beneath the logo.

Listing 2.8 Contents of the HelloCordova `index.html` File

```

<!DOCTYPE html>
<!--
    Licensed to the Apache Software Foundation (ASF) under one

```

or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

-->

```
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="format-detection" content="telephone=no" />
    <!-- WARNING: for iOS 7, remove the width=device-width and
      height=device-height attributes.
      See https://issues.apache.org/jira/browse/CB-4323 -->
    <meta name="viewport" content="user-scalable=no,
      initial-scale=1, maximum-scale=1, minimum-scale=1,
      width=device-width, height=device-height,
      target-densitydpi=device-dpi" />
    <link rel="stylesheet" type="text/css" href="css/index.css" />
    <meta name="msapplication-tap-highlight" content="no" />
    <title>Hello World</title>
  </head>
  <body>
    <div class="app">
      <h1>Apache Cordova</h1>
      <div id="deviceready" class="blink">
        <p class="event listening">Connecting to Device</p>
        <p class="event received">Device is Ready</p>
      </div>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
    <script type="text/javascript">
      app.initialize();
    </script>
  </body>
</html>
```

Notice that the application loads the `cordova.js` and other resources at the end of the file as I explained in the previous section. In this application initialization is done a little differently. Rather than having an `index.js` file that auto-initializes, the `index.js` exposes an `initialize` method that is called manually in a separate `script` tag in the file.

Listing 2.9 shows the contents of the application's `index.js` file.

Listing 2.9 Contents of the HelloCordova `index.js` File

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied. See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
var app = {
  // Application Constructor
  initialize: function() {
    this.bindEvents();
  },

  // Bind Event Listeners
  //
  // Bind any events that are required on startup. Common events are:
  // 'load', 'deviceready', 'offline', and 'online'.
  bindEvents: function() {
    document.addEventListener('deviceready', this.onDeviceReady, false);
  },
  // deviceready Event Handler
  //
  // The scope of 'this' is the event. In order to call the 'receivedEvent'
  // function, we must explicitly call 'app.receivedEvent(...);'.
  onDeviceReady: function() {
    app.receivedEvent('deviceready');
  },
}
```

```

// Update DOM on a Received Event
receivedEvent: function(id) {
    var parentElement = document.getElementById(id);
    var listeningElement = parentElement.querySelector('.listening');
    var receivedElement = parentElement.querySelector('.received');

    listeningElement.setAttribute('style', 'display:none;');
    receivedElement.setAttribute('style', 'display:block;');

    console.log('Received Event: ' + id);
}
};

```

The JavaScript code registers the `deviceready` listener you've seen in many of the other examples in this chapter. When the `onDeviceReady` function executes, it writes some information to the console (this will be discussed more in Chapter 5) and then updates the page content to indicate that the Cordova container is ready.

This application is much more complicated than it needs to be; as you can see from my previous examples, you can easily do the same thing with much less code. However, it's apparently the way the Cordova team wants to highlight how to build Cordova applications.

Note

In the examples I have provided throughout the chapter, I deliberately simplified the application code to make it easier to teach you what a Cordova application looks like. The sample application generated by the CLI is structured more like modern HTML5 applications.

The approach you take when building your web applications is up to you; there's no right or wrong approach. I think the CLI-generated application is more complicated than it needs to be, but as features are added to an application, it may be easier to use the approach highlighted in this section.

Figure 2.6 shows the default Cordova HelloCordova application running on an Android emulator. When building your Cordova applications, you can start with this sample application and add in your custom code, or you can rip out the HTML and CSS files and start from scratch.



Figure 2.6 HelloCordova Application Running on an Android Emulator

Responsive Design and Cordova

When a smartphone or tablet user rotates a device running a web or Cordova application, the browser needs to be able to react to the change and adjust the page's properties. If it didn't, when the browser window switches from a portrait to a landscape orientation, much of the available screen real estate would go unused. Designing a web application so it properly renders the application's content across varying display widths or changing orientations is called responsive design.

Dealing with responsive design is a mobile web development topic, and I've always tried to limit these books to Cordova-related subjects only, but in this case it seemed to make sense to cover this topic. It didn't fit in other areas of the book, so I decided to add it here.

There are several ways you can approach dealing with browser window size and orientation-related challenges. Bootstrap (<http://getbootstrap.com/>) and other front-end frameworks provide capabilities web developers can leverage to automatically scale and adjust their web applications' content based on the available screen real estate. Additionally, there are capabilities in CSS and JavaScript that the web developer can leverage directly to accomplish this. I'm not going to cover third-party frameworks in this chapter; I'll cover some of them in Chapter 17. What I will show you is how to build some of these capabilities into your own applications directly.

Using Cascading Style Sheets, an application has the capability to define specific CSS attributes that apply depending on the orientation of the device. In the following example, you see that

I've defined two `body` styles, one that applies when the content is rendered on a screen while the orientation is portrait and the other when rendered on a screen while the orientation is landscape.

```

/* portrait */
@media screen and (orientation: portrait) {
  /* portrait-specific styles */
  body {
    background-color: blue;
    color: white;
  }
}
/* landscape */
@media screen and (orientation: landscape) {
  /* landscape-specific styles */
  body {
    background-color: red;
    color: black;
  }
}

```

In this case, just so I could demonstrate the changes cleanly, if you add this code to your web application (I'll show you an example in a minute), you get white text on a blue background while in portrait orientation and black text on a red background in landscape orientation. For your own applications, you'll want to adjust margins, borders, and so on based on the space available to your application.

Sometimes you want to do a little more when things change; to accommodate this, the web browser exposes events you can listen for and update your application's UI as needed. Two events that matter for Cordova developers are `orientationchange` and `resize`. To add event listeners for these events to your Cordova applications, you can use the following:

```

//Set the orientation change event listener
window.addEventListener('orientationchange', onOrientationChange);
//For actions that don't fire the orientationchange event
window.addEventListener("resize", onResize, false);

```

With this code in place, when the device's orientation changes, the `onOrientationChange` function is executed, and when the browser window resizes, the `onResize` function is executed. All your application has to do then is populate those two functions with the code you want executed when those particular events happen. In this example, I simply wrote some screen measurements to the page when the events fire.

To see all of this in action, I've created Example 2.7 shown in Listing 2.10. This application implements both the CSS queries and JavaScript events to create a web application that reacts to changes that occur while the application is running.

Listing 2.10 Example 2.7 Application index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Example 2.7</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="viewport" content="user-scalable=no, initial-scale=1,
    maximum-scale=1, minimum-scale=1, width=device-width,
    height=device-height" />
<style>
  /* portrait */
  @media screen and (orientation: portrait) {
    /* portrait-specific styles */
    body {
      background-color: blue;
      color: white;
    }
  }
  /* landscape */
  @media screen and (orientation: landscape) {
    /* landscape-specific styles */
    body {
      background-color: red;
      color: black;
    }
  }
</style>
<script src="cordova.js"></script>
<script>
  br = "<br />";

  function onBodyLoad() {
    alert("Body Load");
    document.addEventListener("deviceready", onDeviceReady, false);
    //set the orientationchange event listener
    window.addEventListener('orientationchange',
      onOrientationChange);
    //for devices that don't fire orientationchange
    window.addEventListener("resize", onResize, false);
    //Fire this at the start to set the initial orientation on
    //the page
    updatePage();
  }

  function onDeviceReady() {
    navigator.notification.alert("Cordova is ready!");
  }
</script>
</body>
</html>
```

```

function updatePage(msg) {
    //Build an output string consisting of the different screen
    //measurement values
    var strongStart = "<strong>";
    var strongEnd = "</strong>";
    //var StrRes, or, sw, sh, ww, wh;
    or = strongStart + "Orientation: " + strongEnd +
        window.orientation + " degrees";
    console.log(or);
    strRes = or + br;
    sw = strongStart + "Width: " + strongEnd + screen.width;
    console.log(sw);
    strRes += sw + br;
    sh = strongStart + "Height: " + strongEnd + screen.height;
    console.log(sh);
    strRes += sh + br;
    ww = strongStart + "Inner width: " + strongEnd +
        window.innerWidth;
    console.log(ww);
    strRes += ww + br;
    wh = strongStart + "Inner height: " + strongEnd +
        window.innerHeight;
    console.log(wh);
    strRes += wh + br;
    document.getElementById('appInfo').innerHTML = strRes;
}

function onOrientationChange() {
    var msg;
    console.log("Orientation has changed");
    switch (abs(window.orientation)) {
    case 90:
        console.log("Device is in Landscape mode");
        break;
    default:
        console.log("Device is in Portrait mode");
        break;
    }
    updatePage();
}

function onResize() {
    console.log("Resize event fired");
    updatePage();
}
</script>
</head>

```



```
<body onload="onBodyLoad()">
  <h1>Example 2.7</h1>
  <p>This is a Cordova application that responds to device
    orientation and resize events.</p>
  <p id="appInfo">Waiting for Cordova Initialization to complete.</p>
</body>
</html>
```

Figure 2.7 shows the application running on an Android device in portrait orientation.

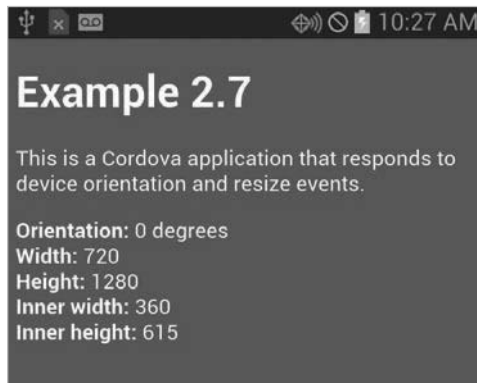


Figure 2.7 Example 2.7 Running on an Android Device in Portrait Orientation

Figure 2.8 shows the application running on an Android device in landscape orientation.

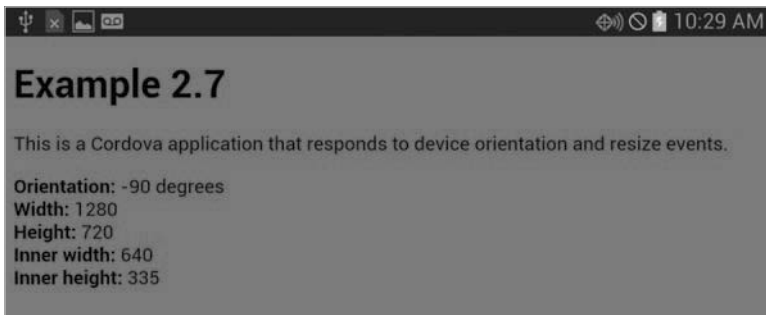


Figure 2.8 Example 2.7 Running on an Android Device in Landscape Orientation

There's a whole lot more that can be said about responsive design and the tools that address it, but that's way beyond the scope of this simple Cordova book. I'm a big fan of *Smashing*

Magazine, and they've published some nice articles on web design and responsive design that might help you with this topic:

- www.smashingmagazine.com/responsive-web-design-guidelines-tutorials/
- www.smashingmagazine.com/2010/07/19/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/
- www.smashingmagazine.com/2012/03/22/device-agnostic-approach-to-responsive-web-design/

There are a lot of web design and development books available that cover this topic in much more detail than I can. For example, take a look at the following Pearson titles:

- Dutson, Phil. *Responsive Mobile Design: Designing for Every Device*. Boston: Addison-Wesley, 2014.
- Kymin, Jennifer. *Sams Teach Yourself Responsive Web Design in 24 Hours*. Indianapolis, IN: Sams Publishing, 2014.

Wrap-Up

In this chapter, I've shown you what makes an application a Cordova application, plus I've shown you the sample application that the Cordova CLI creates for you. You now have the building blocks necessary to start building your own Cordova applications.

In the next chapters, I'll show you how to install Apache Cordova and use the Cordova CLI to create and manage your Cordova projects.

Notice that in this chapter, I didn't do anything to make my applications visually appealing. You can use CSS to do this, which is not something I really want to cover in a Cordova book. In Chapter 17 I'll show you how to use third-party frameworks, some of them specifically designed for hybrid applications, to add pizzazz to your Cordova web applications.

This page intentionally left blank

Index

A

- Aardwolf, 145**
- `about:app-manager`, **Firefox, 206**
- Accelerometer API, 259–260, 335–337**
 - `clearWatch` method, 335
 - `getCurrentAcceleration` method, 335–337
 - `watchAcceleration` method, 335, 337
- `access` **property, config.xml, 132, 134**
- `ACTION_GET_CARRIER_NAME` **constant, CarrierPlugin, 416**
- `ACTION_GET_COUNTRY_CODE` **constant, CarrierPlugin, 416**
- Active Scheme button, Xcode in iOS, 225**
- ADB (Android Debug Bridge) Helper, 206**
- Add Packaged App item, Firefox OS simulator, 207–208**
- `AddEventListener`
 - adding to applications, 46–47
 - Cordova events, 333
 - Cordova initialization, 30, 32, 34
 - debugging with, 137
 - generated web application files, 43
 - `InAppBrowser`, 363, 365–366
 - monitoring events, 333
 - working with Cordova APIs, 35, 37
- Additional Tools window, Windows Phone emulator, 259–261**

Address book. *See* **Contacts API**

`adduser` **command**, **Plugman**, 111

ADM (Android Device Monitor) utility, 191–192

Administrators

creating symbolic links, 82

Windows App Store setup, 251

Adobe

Brackets. *See* Brackets web code editor

Dreamweaver, 16

PhoneGap. *See* PhoneGap

Topcoat. *See* Topcoat UI framework

ADT (Android Developer Tools)

Android SDK installation, 59–65

Android SDK Manager, 170–172

Android Studio vs., 57, 169–170

Ant installation, 56–59

build tools used with, 494

JDK installation, 52–56

ADT IDE

editing application content files, 179–180

importing Cordova project, 180–185

monitoring applications outside of, 191–192

overview of, 178

running Cordova applications, 185–191

`ADT_HOME` **environment variable**, **Visual Studio**, 268–269

`alert()` **method**

Cordova API errors with `onError` vs., 322

Cordova initialization with, 32–33

debugging with Firefox OS simulator, 217–218

testing applications in Xcode, 227

unavailable in universal Windows apps, 33, 247–248

using for debugging, 135–136

as visual notification, 328–329

Alerts, **Windows security**, 101–102

Anatomy

of default PhoneGap application, 310–312

of plugins, 397–398

Anatomy of Cordova application

generated files, 41–44

Hello World! 27–29

initialization, 29–34

leveraging APIs, 35–38

overview of, 27

responsive design, 45–50

structuring code, 38–41

Android

adding platform to existing project, 83–85

content refresh via Hydration, 285

creating application icons, 387–389

creating Cordova native plugin, 410–414

creating native plugins, 414–423

creating Plugman project, 106–108

deploying PhoneGap Build applications, 302–305

executing applications with `run`, 99–100

GapDebug on, 151–156

Hello World using Bootstrap on, 455

Hello World using jQuery Mobile on, 447–448

Hello World using OpenUI5 on, 459

Hello World using TopCoat on, 442

InAppBrowser benefits, 362

PhoneGap Developer running on, 150

platform support with Plugman, 118

Plugin Development Guide, 397

StatusBar plugin for, 368–370

testing application on emulator, 389–392

using merges, 385–387

`android` **command**, 170

Android Debug Bridge (ADB) Helper, 206

Android Developer Tools. *See* **ADT (Android Developer Tools)**

Android development with Cordova

- ADT vs. Android Studio, 169–170
- Android SDK Manager, 170–172
- monitoring application outside ADT IDE, 191–192
- overview of, 169
- testing on physical device, 192–195
- using ADT IDE. *See* ADT IDE
- using Android Virtual Device Manager, 172–177
- using Chrome debugging tools, 195–202

Android Device Monitor (ADM) utility, 191–192**Android SDK**

- Firefox App Manager and, 206
- LogCat, 138
- managing Android Virtual Devices, 173–177
- managing with Android SDK Manager, 170–172
- Tools, 59–60

Android SDK Manager (ASM), 170–172**Android Studio (beta)**

- vs. ADT, 169–170
- Android SDK installation and, 59–60
- configuring Android SDK and, 60–62
- installer for, 60
- migration from ADT to, xvii–xviii

Android Virtual Devices. *See* AVDs (Android Virtual Devices)**Ant**

- adding Android to existing project, 83–85
- build tools used with ADT, 494
- installing for Android, 56–59

Apache Cordova 3 Programming, xvi–xviii, 74, 169, 220, 373, 469**Apache Cordova API Cookbook, xv–xviii, 7, 22, 154, 229, 317, 335, 353, 371, 439, 446****Apache Ripple. *See* Ripple Emulator (Apache Ripple)****APIs, Cordova**

- implementing, 23–24
- inconsistency of, 122–123

leveraging, 35–38

- loading library in initialization, 32
- native, 5–10
- overview of, 4–5

App Manager. *See* Firefox App Manager**AppGyver, productivity enhancement tool, 486–489****app.js file, Ionic framework, 463–464****Apple iOS Developer Program, 62****Application development, end-to-end**

- about, 373
- adding mobile device platforms, 374–375
- adding plugins, 375–376
- adding style, 378–380
- creating compass watch, 381–382
- creating new Cordova project, 374
- preparing for debugging, 380–381
- updating config.xml file, 376–377
- using merges, 385–387
- writing to console, 383

Apps List, Firefox App Manager, 214**apt-get update, 236****Aptana Studio, 16**

- editing HTML and JavaScript source code, 473
- keeping web content files open when working in Cordova, 126
- using Aptana IDE for editing JavaScript code, 469

Arguments

- bash script, 161
- Windows command file, 158

ARM-based Android emulators, 172**Arrays, contact object, 353****ASM (Android SDK Manager), 170–172****Attributes**

- Cordova initialization and viewport, 31
- jQuery Mobile, 445–446
- responsive design with CSS, 45–50

author property, config.xml, 132, 134**Automation. *See* Cordova CLI, automating**

Availability, Cordova API, 320–321

Available Packages window, WebStorm, 482–483

Available Software Wizard

Eclipse, 179–180

THyM, 490–493

AVDs (Android Virtual Devices)

creating, 174–175

creating/managing system emulator definitions, 191

defined, 172

Intel-based/ARM-based emulators for, 172–173

launch options, 176–177

naming, 174–175

selecting at runtime, 190

viewing device emulator settings, 174–175

wiping existing user data, 175

B

BackgroundColor preference, config.xml, 133

backgroundColorByName () method, StatusBar, 370

Bash scripts, automating project setup, 161

Basic Settings, PhoneGap Build, 291–293

Batch files, automating tasks, 158

Battery Status plugin, 333–334

Beautify extension, Brackets, 474–475

beep method, hardware notifications, 327

Bootstrap, 45–50, 450–456

Bower, Joe, 72

Brackets web code editor

Auto Close Braces option, 474

coding Cordova, 16

Cordova plugin, 476–479

editing web applications, 223

keeping web content files open when working in Cordova, 126

PhoneGap plug-in, 479

working with, 473–476

Breakpoints

debugging Ubuntu applications, 240–241

debugging with Visual Studio, 262–263

Firefox App Manager Debugger, 215–216

in JavaScript code with GapDebug, 154–155

remotely debugging iOS with Safari, 231–232

Browser

testing applications in desktop, 103

weinre debug server and client, 140–141

window size in responsive design, 45–50

build command, Cordova CLI, 98

Build process

Cordova application, 16–18

Cordova CLI managing, 96–98

PhoneGap, 3

Build tools

Grunt, 500–503

Gulp, 494–500

managing Android SDK, 170–172

overview of, 494

buttonLabel, 329

Buttons

adding to application's main page, 34

Help, 206

physical menu vs. virtual, 332–333

Web Inspector JavaScript debugger, 231–232

C

C#, Windows Phone 8 on Cordova, 245

Callback functions

alerting user, 329–330

browser window events, 364

camera, 345

Cordova APIs, 8

Cordova initialization, 33

Cordova native plugin, 413

Cordova native plugins, 412–413

- execute scripts, 364
- iOS plugin, 426
- Media API, 358
- mol.js file for plugins, 402–403
- onSuccess function, 366
- callback **parameter**
 - alert () method, 329
 - confirm () method, 329–330
 - prompt () method, 330
- callbackContext.success () **method, Android plugin, 416**
- Camera API, 319, 340–345**
 - cameraOptions, 343
 - Discard button, 341
 - getPicture method, 340–343
 - Save button, 341–342
 - Take Photo button, 341
 - targetHeight property, 343, 345
 - targetWidth property, 343, 345
- Camera plugin, 340**
- carrier.java file, CarrierPlugin class, **414–416**
- carrierName property, iOS plugin, 426
- Cascading style sheets. See CSS (cascading style sheets)**
- cd **command, changing directories for new project, 77–82**
- CDN (content delivery network)-hosted files, 448**
- cdva-create, **162–167**
- cdva-hooks, **166**
- CEF (Chromium Embedded Framework), Brackets, 473**
- charset tag, Cordova initialization, **31**
- ChildBrowser plugin, 360**
- Chrome DevTools**
 - Console, 197–198
 - Elements pane, 199
 - Inspect Devices Application List, 197
 - JavaScript error, 201–202
 - overview of, 195–202
 - Sources pane, 200–201
 - testing plugin with, 406–407
 - USB debugging with, 195–197
- Chromium Embedded Framework (CEF), Brackets, 473**
- Click chroot, Ubuntu, 242**
- CLIs (command-lines)**
 - configuring proxy settings, 72–74
 - Cordova. *See* Cordova CLI
 - enabling verbose output, 74–75
 - Ionic, 460–464
 - PhoneGap. *See* PhoneGap CLI
 - Plugman. *See* Plugman CLI
 - types of, 71–72
 - WebStorm supporting, 479–485
- close **method, InAppBrowser, 363**
- .cmd **extension, Windows, 158**
- Code**
 - adding style to project, 378–380
 - Cordova application, 15–16
 - downloading for projects in this book, 30
 - structuring application, 38–41, 380
 - third-party validation tools, 469–473
- Code editors, third-party**
 - Adobe Brackets, 473–479
 - Aptana, 126, 469, 473
 - overview of, 473
 - WebStorm, 479–487
- Collaboration, PhoneGap Build, 285, 292**
- Color, StatusBar, 370**
- Command lines. See CLIs (command-lines)**
- Command summary**
 - Cordova CLI, 76
 - Plugman CLI, 105
- Compass API**
 - clearHeading method, 338
 - getCurrentHeading method, 338

Compass API (continued)

- heading object, 338
- onSuccess function, 382
- overview of, 337–339
- results object, 337–339
- watchHeading method, 338–339
- watchOptions object, 381–382

Compass app

- creating application icons, 387–389
- creating compass watch, 381–383
- with jQuery mobile look, 380
- preparing application, 376–378
- rotating compass graphic, 378–379
- testing application, 389–392
- using merges, 385–387

compile **command**, **Cordova CLI**, **76**, **97**

Components, **Cordova**, **4–5**

Composer application, **AppGyver**, **486**

ConfigGAP

- Advanced Settings tab, 296–297
- General Settings tab, 296
- Permissions tab, 298
- PhoneGap Build, 295, 299–301
- Plugins tab, 298–299

config.json file, **CLI configuration**, **79**

config.xml file

- Cordova application options, 131–134
- creating new Cordova project, 79
- debugging with Firefox OS simulator, 209
- PhoneGap application anatomy, 310–312
- PhoneGap Build configuration, 294–301
- PhoneGap Build plugins added via, 301–302
- saving plugins, 92–94
- uninstalling plugins, 104
- updating information in new application, 376–377
- Visual Studio tools for Cordova, 271–272

confirm() method, **visual notifications**, **329–330**

connection object, **324–326**

Console

- ADT IDE, 185–187
- Chrome DevTools, 196–197
- debugging by writing to, 136–139, 226, 383
- Firefox App Manager, 214
- PhoneGap Server, 151
- viewing output in GapDebug, 155–156
- viewing output in weinre Debug Client, 145

Contacts API

- based on W3C Contacts API, 352
- contact object, 352–357
- contact properties, 353
- Contact picker, 356–357
- create method, 352–353
- find method, 355, 357
- methods, 354–355
- multiple property, 355
- newContact object, 352–353
- options object, 355
- phone numbers, 356
- pickContact, 356–357
- populating contact object, 354
- save method, 354–355

Container, **designing web applications for**, **13–15**

Content

- changing with weinre Debug Client, 142–145
- editing Cordova application, 179–180
- GapDebug highlighting HTML, 154
- loading with InAppBrowser, 360–363
- refreshing in PhoneGap Build, 285

content delivery network (CDN)-hostedfiles, **448**

content property, **config.xml**, **132**, **134**

coolMethod, **Javascript-only plugin**, **401–402**

--copy-from switch, **project create**, **81–82**

Cordova, introduction to

- accessing native APIs, 5–10
- Adobe PhoneGap, 3–4
- best uses of, 18–19
- building applications, 16–18
- coding applications, 15–16
- components, 4–5
- designing for container, 13–15
- getting support, 20
- going forward, 23–24
- hybrid application frameworks, 25
- license for, 13
- overview of, 1–3
- resources for, 20–23
- supported platforms, 12–13
- user interface capabilities, 10–12

Cordova APIs

- Accelerometer API, 335–337
- alerting user, 326–331
- Camera API, 341–345
- catching errors, 321–322
- checking availability, 320–321
- Compass API, 337–339
- Contacts API, 352–357
- Cordova events, 332–334
- Cordova objects, 324–326
- core, 317–319
- documentation, 319–320
- Geolocation API, 339–341
- Globalization, 347–352
- hardware APIs. *See* Hardware APIs
- InAppBrowser, 359–366
- Media API, 358–359
- Media Capture API, 345–347
- overview of, 317
- setting application permissions, 322–324
- Splashscreen API, 367
- StatusBar plugin, 367–370

Cordova CLI

- adding platforms, 82–85
- adding plugins, 87–90
- Android and. *See* ADT (Android Developer Tools)
- build management, 96–98
- building applications with, 18
- command summary, 76
- cordova.js file and, 32
- creating new Cordova project, 77–82, 124
- creating Plugman project, 105–106
- displaying project information, 95–96
- executing Cordova applications, 98–103
- generated web application files, 41–44
- help, 77
- installing on iOS, 66–67
- installing on Ubuntu, 235–236
- iOS requirements, 68–69
- listing platforms, 85
- listing plugins, 90
- overview of, 51–52, 75
- PhoneGap tools, 3
- Plugman. *See* Plugman CLI
- as primary CLI, 75
- removing platforms, 86
- removing plugins, 90–91
- restoring plugins, 94
- saving plugins, 92–94
- searching for plugins using, 91–92
- updating platforms, 86–87
- upgrading Cordova/Cordova projects, 103–104
- using, 76–77
- WebStorm supporting, 479–485

Cordova CLI, automating

- bash script for, 160–161
- of Cordova processes, 164–167
- NodeJS used across platforms, 162–164
- overview of, 157

Cordova CLI, automating (*continued*)

project setup, 157

Windows command file for, 158–160

cordova **command**, 33

cordova `compile`, 97

cordova `create`

new Cordova application, 124

new Cordova project, 77–82

Plugman project with Cordova CLI, 105–106

Cordova Device Motion API, 260–261

`.cordova` folder structure, CLI, 79–80

cordova-`icon`, 388

cordova `info` **command**, 95–96

cordova `platform add windows`, 246

cordova `platform add wp8`, 245

cordova `platforms check`, 86–87, 104

cordova `platforms up android`, 87

cordova `platforms update`, 87, 104

Cordova Plugin Registry. See CPR (Cordova Plugin Registry)

Cordova plugins

adding plugins in Cordova CLI, 89

adding to applications, 375–376

adding to projects, 403

anatomy of, 397–398

creating Android plugin, 414–423

creating iOS plugin, 424–430

creating JavaScript-only plugin, 398–399

creating native plugins, 408–414

`mol.js` file, 400–403

publishing, 109, 431–434

removing, 91

restoring, 94

testing, 403–408

cordova `prepare`

build management with, 96–97

copying `help.html` file into appropriate web content folders, 362

copying platform content into appropriate web content folders, 129, 207

creating project with Visual Studio, 254–255

developing Cordova applications, 125–127

getting information about prepare process, 74

importing Cordova project, 180, 183

testing application with device emulator, 389

working with Gulp and, 498–499

working with Xcode in iOS, 221–225

Cordova processes, automating, 164–167

cordova `run`

debugging Ubuntu applications, 237, 242–243

testing on Android emulator, 389–392

cordova.`exec` **method**, 412, 416

cordova.`js` file, 32, 39–40

CoreTelephony framework, testing iOS plugin, 427

CPR (Cordova Plugin Registry)

adding plugins from, 88, 431

adding user, 111

overview of, 110–111

plugin owner(s), 115–116

Plugman configuration information, 114–115

publishing plugins, 111–113

removing plugins installed from, 90–91

searching for plugins using, 113–114

unpublishing plugins from, 113

viewing plugin information, 114, 433

WebStorm, 482

`create` **command**, in Cordova CLI, 76

`create` **script**, Plugman project with shell scripts, 106–108

Cross-platform mobile development, 8

CSS (cascading style sheets)

adding style to project, 378

changing with weinre Debug Client, 145

inserting with InAppBrowser, 365–366

jQuery Mobile containing, 444

responsive design using, 45–50

Topcoat UI framework based on, 439–443

CSS libraries, jQuery Mobile, 445

`cssInfo` parameter, `InAppBrowser`, 366

`CTTelephonyNetworkInfo` class, `iOS` plugin, 426

`cordovaReady` object, structuring application code, 41

D

`-d` flag, 74–75, 80. *See also* `--verbose`

Dashcode, iOS, 223

`data-role` attribute, `jQuery Mobile`, 446

Debugging

Android applications, 179, 193–194

Cordova applications with Visual Studio, 262–265

Firefox OS applications with device, 218–220

Firefox OS applications with simulator, 207–218

iOS applications remotely with Safari, 227–233

preparing application for, 380–381

separating JavaScript code for, 39

on simulator vs. physical device, 224

Ubuntu applications, 237–243

Windows Phone 8, 245–246

Debugging tools

`alert()`, 135–136

`GapDebug`, 151–156

PhoneGap Developer app, 148–151

Ripple Emulator (Apache Ripple), 145–148

`weinre`, 139–145

writing to console, 136–139

Delete packages button, ASM, 170

`description` property, `config.xml`, 132, 134

Design

container, 13–15

responsive web application, 45–50

Develop menu, remotely debugging iOS, 228

Developer productivity enhancement tools

`AppGyver`, 486–489

`Eclipse THyM`, 490–493

overview of, 485–486

Developer Settings page, Firefox OS, 218–220

Developer types, 71–72

Development environment, configuring

Android development tools, 52–62

CLI installation, 65–69

Cordova CLI installation, 51–52

iOS development tools, 62–65

overview of, 51

`Plugman` installation, 69–70

Development mechanics

API inconsistency, 122–123

application graphics, splash screens, and icons, 123–124

configuring applications, 131–134

creating applications, 124–131

debugging capabilities. *See* Debugging tools overview of, 121

testing applications, 134–135

Development process, end-to-end

about the application, 373

application icons, 387–389

creating application, 374–385

testing application, 389–395

using merges, 385–387

Device Applications pane, Firefox, 211

Device Information page, Firefox, 219

Device Motion plugin, 335

`device` object, 326

Device Orientation plugin, 337–339, 376

Device Orientation, Xcode in iOS, 224

Device plugin, 326, 376`device.platform`, **API inconsistency, 123****deviceready event**

- checking API availability, 321–322
- Cordova applications responding to, 93, 95, 295
- Cordova initialization, 29, 32–34
- debugging with `alert()`, 135–136
- delay in event firing and, 407
- event listener for, 40–41, 43, 273, 467
- generation of device information list following firing of, 440
- Hello World using jQuery Mobile, 447
- setting `onDeviceReady` for, 381

Devices utility, Xcode, 227`device.version`, **API inconsistency, 123****Dialog plugin, 375–376****Directories, changing for new project, 77–82**`DisallowOverscroll` **preference, config.xml, 133****Displaying project information, Cordova CLI, 95–96**`displayName` **property, contact object, 353****Documentation**

- application permissions, 323
- Cordova, 5, 21–23
- Cordova API, 319–320
- Geolocation API, 318
- JSHint, 472
- PhoneGap Build, 295
- PhoneGap CLI, 308
- plugin, 431
- Plugin APIs, 319
- plugin.xml file, 400
- testing Android on physical device, 194–195
- weinre, 145

Domain Access tab, config.xml, 271**DOS command, automating tasks, 158****Dreamweaver, Adobe, 16****Dynamic pages, 13–14****E****Eclipse**

- adding ADT to, 178
- as Android Developer Tool, 59
- coding Cordova applications, 16
- editing Cordova application content files, 179–180
- THyM, 490–493

Edit Configuration, WebStorm, 482–483**Editing, Cordova application content files, 179–180****Elements pane**

- Chrome DevTools, 200
- Ubuntu, 239

emulate command

- in Cordova CLI, 76
- Ripple Emulator and, 147
- running application on device emulator, 100, 207, 464

Emulator

- Android. *See* AVDs (Android Virtual Devices)
- grabbing screen shot, 192
- running Cordova application in ADT IDE, 186–188
- running Cordova application on Ripple, 147–148
- testing Android plugin, 420–422
- testing application with Android, 389–392
- testing weinre, 142–145
- vs. simulator in this book, 135
- Windows Phone, 259–261

End-to-end process. *See* Development process, end-to-end**Environment variable overrides, Visual Studio, 268–269****errorCallback parameter, Cordova native plugin**

- error capture and, 7–8
- executed on failure of plugin method call, 412–413

Errors

- Camera API, 342
- catching Cordova API, 321–322
- code validation tools, 469–473
- Globalization API, 348
- InAppBrowser window events, 364
- JSHint code validation, 471–472
- JSLint code validation, 470–471
- Media Capture API, 347
- Ripple, 148

Events

- Cordova supported, 332–333
- InAppBrowser window, 363–364
- responsive design, 46

exec object, Cordova native plugin, 412–413

Execute scripts, InAppBrowser, 364–365

executeScript method, InAppBrowser, 364–365

exit event, InAppBrowser, 363

Extensions, searching for, 474–475, 477

external web content editor, 179–180

F

Files, plugin. See Plugins

Filter, ADT IDE LogCat panel, 188

filter property

- Contacts API, 355
- watchOptions object, 381–382

Firefox App Manager

- debugging on Firefox OS device, 218–220
- debugging with Firefox OS simulator, 207–218
- device compatibility issues, 220
- Firefox OS and, 203–207

Firefox OS

- debugging applications, 218–220
- debugging with simulator, 207–218

developer tools, 203–207

overview of, 203

simulator, 203, 207–209

Folders

- Android SDK installation, 61
- Bootstrap, 450
- Cordova native plugin structure, 410
- creating Android plugin, 419
- creating new Cordova project, 78–82
- developing Cordova applications, 124–126
- hybrid application, 269–270
- plugin, 90, 128
- ThyM, 492–493

frequency property, watchOptions, 381–382

Fullscreen preference, config.xml, 133–134

G

GapDebug, 151–156

Generated files, 41–44

Geolocation API, 318, 339–340

clearPosition method, 339–340

coordinates property, 340

getCurrentPosition method, 339–340

timestamp property, 340

watchPosition method, 339–340

getCarrierName method

Android device results, 423

Android plugin, 418

Cordova native plugin, 412

getCountryCode method

Android device results, 423

Cordova native plugin, 412

iOS plugin, 426–430

Getting Started page, Bootstrap, 450

Git

installing for iOS, 65–66

proxy settings for, 73

`git config`, 73

`GIT_HOME` environment variable, 268–269

GitHub

accessing plugin source code, 398

adding plugins from, 88

Cordova 3 storing plugins in, 432

removing plugins installed from, 90–91

Globalization API, 347–352

adding Globalization plugin to project, 347

`dateToString` function, 350–351

error codes, 348

examples of use of, 350–352

methods and parameters, 348–349, 351

Globalization plugin, 347

Google Chrome, 151–152, 195–202. See also Chrome DevTools

Google Gmail, 14

Google Maps, 14

Gradle

Ant vs., 57

Build tools used with ADT, 494

Graphics

application icons, 387–389

application issues and, 123–124

Splashscreen API and, 367

testing on physical devices, 393

Grunt tool

as build tool, 494

executing Grunt process, 502–503

`gruntfile.js` file, 500–502

installing, 500

overview of, 500

types of processes managed by, 167

`gruntfile.js` file, 500–502

Gulp tool

as build tool, 494

example of use in Cordova project, 495

installing and populating `gulpfile.js`, 494–495

Ionic and, 464–465

Ionic including configuration files for, 460–462

listing of `gulp.js` file, 495–497

tasks performed by, 498–500

types of processes managed by, 167

`gulpfile.js`, 495–497

H

Hardware APIs

Accelerometer API, 335–337

Camera API, 341–345

Compass API, 337–339

Geolocation API, 339–341

Globalization API, 347–352

Media Capture API, 345–347

overview of, 334–335

Hardware notifications, Cordova APIs, 326–327

HAXM (Hardware Accelerated Execution Manager) installer, 173

Heading watch, 381–382**Hello World!**

Creating Cordova project, 77–82

generated files, 41–44

initialization, 29–34

leveraging Cordova APIs, 35–38

responsive design and Cordova, 45–50

starting with, 27–29

structuring code, 38–41

updating with UI from Topcoat, 439–441

using Bootstrap, 454–455

using Ionic framework, 461–464

using jQuery Mobile, 444–447

using Onsen UI, 465–466

using OpenUI5, 456–457

Help

Firefox App Manager, 204

PhoneGap CLI, 308–309

help, **Cordova CLI**, 76–77

hide() **method**, **StatusBar**, 368–369

HideKeyboardFormAccessoryBar
preference, **config.xml**, 134

Highlighting

HTML content with GapDebug, 154

target content with weinre, 143–145

Home Screen, Firefox OS 2.0, 212–213

Hooks folders

adding, 269

complexity added to project with, 106

creating, 166–167

enhancing CLI commands with, 165

in new Cordova project, 79

options, 165–166

role in code execution, 131

role in execution of code validation tools, 469

HTML

designing for container, 13–14

generated web application files, 41–44

structuring application code, 38–41

updating content in Web Inspector, 230–231

web applications using HTML5, 14–15

http.proxy **setting**, **Git**, 73

https.proxy **setting**, **Git**, 73

Hybrid applications

best use of, 18–19

defined, 1

designing for container, 14–15

framework options, 25

Ionic framework for, 459–464

managing using AppGyver, 487–489

Monaca as, 464

for new project with ThyM, 490–492

WebStorm support for, 482–483

Hybrid Mobile project, ThyM, 490

Hydration, PhoneGap Build, 285

I

Icons, creating application, 123–124, 387–389

ID

adding plugin to project using, 433

removing plugins using, 90–91

searching for plugin with unknown, 91–92

IDE plugins, 18, 178–191

ImageMagick, 388

Import Wizard, ADT IDE, 181–183

Importing, Cordova project

copying files, 183–184

cordova prepare command and, 180

locating project folder, 182

using ADT IDE Import wizard, 181–183

viewing available software with Eclipse
wizard, 180

InAppBrowser API

execute scripts, 364–365

insert CSS, 365–366

issues with, 360

loading content, 360–363

overview of, 359–360

user interface, 12

window events, 363–364

index.html file

adding reference to file in, 378–380

ADT IDE, 184

checking API availability, 320–321

contents of HelloCordova, 41–44

Hello World using Ionic framework, 461–462

Hello World using jQuery Mobile, 444–445

Hello World using Onsen UI, 465–466

index.html file (continued)

- Hello World using OpenUI5, 456–457
- testing Android plugin, 420–421
- testing iOS plugin, 428
- testing plugin on Chrome DevTools, 406–407
- Visual Studio tools for Cordova and, 273

index.js file

- contents of, 43–44, 383–385
- Hello World updated with UI from Topcoat, 439–440
- Hello World using Bootstrap, 454–455
- Hello World using jQuery Mobile, 446–447
- Hello World using Onsen UI, 466–467
- Hello World using OpenUI5, 457–458
- JSHint code, 471–472
- JSLint code, 470–471
- splitting JavaScript into own, 380
- structuring application code, 40–41
- testing Android plugin, 421–422
- testing Javascript-only plugin, 404–405

info command, Cordova CLI, 76

Initialization

- creating Android plugin, 416
- creating Cordova applications, 29–34

insertCSS method, InAppBrowser, 366

Inspect Devices Application List, Chrome DevTools, 196–197

Inspector window, Firefox App Manager, 215

Install packages button, ASM, 170

Install page, PhoneGap Build, 302–303

Install Wizard, ADT, 179–180

Intel-based Android emulators, 172–177

Intel XDK, Apache Ripple for, 145

IntelliJ IDEA IDE, 59

Ionic UI framework

- installing, 460
- overview of, 459–464
- WebStorm supporting, 479–485

iOS

- adding platform to existing project, 83–85
- application icons for, 387–389
- automating project setup across platforms, 163
- build/deployment to simulator in Visual Studio, 275–281
- content refresh via Hydration, 285
- Cordova native plugin for, 410–414
- GapDebug on, 151–156
- Hello World using jQuery Mobile on, 447–448
- Hello World using Onsen UI on, 467
- Hello World using Topcoat on, 442–443
- InAppBrowser benefits for, 362
- installing, 68–69
- jQuery Mobile built to mimic, 444
- platform support with Plugman, 118
- plugin for, 424–430
- Plugman project with Cordova CLI for, 106
- StatusBar plugin for, 368–370
- testing application on simulator, 389–390
- using merges, 385–387

iOS development tools

- CLI requirements, 68–69
- Cordova installation, 66–67
- Git installation, 65–66
- Node installation, 65
- overview of, 62–65

iOS development with Cordova

- overview of, 221
- testing applications in Xcode, 225–227
- using Safari Web Inspector, 227–233
- working with Xcode, 221–225

isoCountryCode property, iOS plugin, 426

isOffline function, 333

J

`javac` command, installing JDK for Android, 52, 56

JAVA_HOME environment variable, Android development, 52–56

JavaScript

- access to Cordova native APIs, 5–10
- build tools used with Cordova, 494
- Chrome DevTools error, 201–202
- configuring application for `weinre`, 140–141
- Cordova `alert` results vs., 328
- Cordova application packaging process and, 2–3
- Cordova native plugin using, 411–412, 414
- failing silently in Cordova applications, 138
- jQM libraries, 444–445
- plugins containing source file for, 8–9, 398
- structuring application code by separating, 38–41
- try/catch block for API inconsistency, 123

JavaScript Dynamic Content shim, Windows Store apps, 249

JavaScript-only plugins, creating

- `mol.js` file, 401–403
- overview of, 398–399
- Plugin Development Guide for, 397
- `plugin.xml` file, 399–401
- testing, 403–408

JBoss, ThyM tool, 490–493

JDK (Java Development Kit), Android development, 52–56

JetBrains, 479

jQM (jQuery Mobile)

- adding style to project, 378–379
- rotating compass graphic on page, 378–379
- structuring application code for, 39
- as third-party UI framework, 444–450

`js-module` element, `plugin.xml`, 417–418

JSHint tool

- code editors supporting, 473
- installing, 471
- overview of, 471–473
- problems report in Brackets, 475–476
- WebStorm support, 473, 485

JSLint tool

- code editors supporting, 473
- installing, 470
- overview of, 470–471

JSON, 79, 80

`.jsproj` files, **Windows 8.1**, 246

K

Keywords, searching for plugins, 433–434

L

Landscape orientation

- Hello World using Bootstrap on Android, 455
- in responsive design, 45–50
- working with Xcode in iOS, 223–225

Lazy loading, Cordova CLI, 79

Libraries

- in Cordova initialization, 32
- downloading to use jQuery Mobile, 445
- jQM JavaScript, 444

Licenses

- Cordova, 13
- for generated web application files, 41–42
- Visual Studio for Windows Phone using, 250–251

`--link-to` switch, **folders**, 82

Links, symbolic, 82

Linux

- Cordova tools for Ubuntu supported on, 235
- creating application icons, 388

Linux (continued)

- executing CLI commands in, 76–77
- installing Cordova CLI on Ubuntu, 236
- installing Grunt in, 500
- installing Gulp in, 494
- installing Ripple in, 146
- updating Cordova CLI/Cordova projects, 103–104
- writing bash script to automate project setup, 160–161

Listing

- project platforms, 85
- project plugins, 90

ListView, jQuery, 445–447, 459**Live preview, Brackets, 473–474****loaderror event, InAppBrowser, 363****loadstart event, InAppBrowser, 363****loadstop event, InAppBrowser, 363****Local content, InAppBrowser, 362–363****Locals panel, Visual Studio, 264****Location**

- Geolocation API, 339–340
- Windows Phone emulator, 260–261

'location=yes' parameter, InAppBrowser, 361**LogCat**

- Android SDK, 138
- Message Filter Settings in ADT IDE, 186–188
- monitoring activity outside of ADT IDE, 191–192

M

Macintosh OS X

- Android SDK configuration on, 60–62
- application icons for, 388
- automating project setup across platforms, 162–164

automating project setup with bash script, 160–161

Brackets Cordova plugin on, 476–479

executing CLI commands on, 76–77

GapDebug on, 152

installing Ant for Android on, 56–59

installing Grunt on, 500

installing Gulp on, 494

installing JDK for Android, 52–56

installing JSHint on, 471

installing Ripple on, 146

installing weinre on, 140

iOS development, 65–69

remote agent installation, 276–278

remotely debugging iOS with Safari, 227–233

updating Cordova CLI/Cordova projects, 103–104

Windows development system requirements, 249

Xcode IDE installation using, 63–65

MakeAppicon, 388

makeListItem function, Topcoat UI, 440

Manifest Editor, 209

Manifest file, HTML5, 14

MarkdownPad, 431–432**MDHAE (Multi-Device Hybrid Apps Extension)**

folder structure, 269–270

installation of, 265–266

overview of, 265

warning page display, 268

Media API, 358–359

media object, 358–359

media plugin, 358–359

OnStatus function, 358–359

Media Capture API, 340, 345–347

captureAudio method, 346

captureVideo method, 346

duration property, 347

`getFormatData` method, 346

`options` parameter, 347

Media Capture plugin, 345

Merges folder

copying application content from, 96
managing platform-specific files with,
129–131

using merges, 385–387

`MessageDialog` class, troubleshooting

Windows, 247

Methods

Globalization API, 349

Media API, 358

Microsoft

configuring Windows Phone device for app
testing, 253

creating account, 251

registering Windows Phone device with, 251

Windows. *See* Windows

`mol.js` file, 401–403, 408

Monaca cloud-based development, 464

More Information page, Firefox OS Settings, 219

Multi-Device Hybrid Apps Extension. *See* MDHAE
(Multi-Device Hybrid Apps Extension)

`myplugin.js`, 117–118

`myplugin.m` file, 118

N

`name` property, `config.xml`, 132, 134

Naming conventions

Android Virtual Devices, 174–175

debugging application with `weinre`, 142

hooks files, 166

new Cordova project, 77

PhoneGap Build application, 288

plugins, 89–90

Windows Phone device, 251–252

Native APIs

access to, 5–6, 127

Cordova initialization and, 29

creating Cordova native plugins, 408–414

list of, 6–7

setting application permissions, 322–324

Native code

creating Android plugin, 414–423

creating Cordova native plugin, 408–414

creating iOS plugin, 424–430

debugging iOS with Xcode, 226

deploying PhoneGap Build applications,
302–303

not necessary in plugins, 398

Native developers, using `Plugman CLI`, 72

Navigator, Cordova initialization, 33–34

`navigator.notification.alert()`

in Cordova initialization, 30, 32–34

debugging with `alert()` vs., 136

leveraging Cordova APIs, 35, 37

structuring application code, 40

Network connectivity, loss of, 333

Network Information plugin, 324

New Project dialog, Visual Studio, 266–267

New Project wizard, `ThyM`, 490–491

Nitobi, and PhoneGap, 4

NodeJS (Node JavaScript runtime engine)

build tools used with Cordova, 494

Cordova CLI using, 51

cross-platform approach using, 162–164

installing `weinre`, 139–140

iOS development, 65

Norton Antivirus, 209

Notepad, 15

Notifications, user

hardware, 326–327

overview of, 326

visual, 327–331

npm config command, proxy settings, 73–74

NPM (Node Package Manager)

- adding Console plugin, 375
- adding mobile device platforms to application, 374–375
- automating project setup across platforms, 162–164
- configuring proxy settings for, 73
- creating application icons, 388
- Git installation for iOS development, 65
- installing Cordova CLI for iOS, 66–67
- installing Cordova CLI on Ubuntu, 236
- installing Grunt, 500
- installing Gulp, 494
- installing JSHint, 471
- installing JSLint, 470
- installing Ripple on Windows, 146
- Ionic including configuration files for, 460–461
- plugins now stored in, 432

npm-g

- cdva-create Node installation, 162
- Cordova icon Node installation, 388
- Cordova installation, 66–67, 236
- Git installation, 65–66
- Grunt installation, 500
- Gulp installation, 494, 497
- Ionic installation, 460
- iOS installation, 68–69
- JSHint installation, 471
- JSLint installation, 470
- Plugman installation, 69
- remote Macintosh agent installation, 276
- Ripple Emulator installation, 146
- updating Cordova, 103
- weinre installation, 139–140

O

Objective-C, iOS applications, 225

Objects, Cordova, 324–326

onBodyLoad function

- Cordova initialization, 32, 34
- debugging with alert(), 136

onDeviceReady function

- Cordova initialization, 32, 34
- generated web application files, 44
- leveraging Cordova APIs, 36
- setting deviceready event, 381
- structuring application code, 41
- Topcoat UI, 440

onError function

- catching Cordova API errors, 322
- Compass API, 382–384
- preparing application for debugging, 380–381
- testing on Android emulator, 392

OnFailure function

- Accelerometer API, 336–337
- Compass API, 338–339
- Contacts API, 354–355
- Globalization API, 348
- Media API, 358–359
- Media Capture API, 346–347

onload event function, Cordova initialization, 32

onOrientationChange function, responsive design, 46–48

Onsen UI framework, 464–468

OnSuccess function

- Accelerometer API, 336–337
- Compass API, 337–339
- Contacts API, 354–355
- Geolocation API, 339–340
- Globalization API, 348, 350

InAppBrowser, 364–365, 366
 Media API, 358–359
 Media Capture API, 346

Open Project link, Visual Studio, 256

OpenUI5 (SAP), 456–459

Orientation

- Accelerometer API, 337
- Hello World using Bootstrap on Android, 455
- responsive design and, 45–50
- Windows Phone emulator, 259–260
- working with Xcode in iOS, 223

Orientation preference, config.xml, 133–134

`overlaysWebView()` method, **StatusBar, 368–369**

P

Package Explorer, 183, 186

package.json file

- executing Grunt process, 502–503
- executing Gulp process, 497–500
- publishing plugin, 432

Packaging process, 2

Packaging tab, config.xml, 271–272

PATH environment variable. See also System path

- adding A nt to System path, 62
- Ant installation, 57–58
- Android JDK installation, 54–56
- Android SDK installation, 61–62
- editing Windows Path variable, 56

Pause button, Firefox App Manager, 215–216

Permissions

- Android application, 419–420
- ConfigiGAP tab for, 298
- Firefox App Manager Device, 212
- setting application, 322–324

Personal Package Archive (PPA), Ubuntu, 236

PhoneGap

- history of Cordova and, 4
- plugin for Brackets, 479
- resources for, 20–21
- understanding, 3
- used in this book, 3
- using with ThyM, 492
- WebStorm supporting, 479–485

PhoneGap Build

- adding plugins to project, 301–302
- collaboration, 285
- configuring application, 294–301
- content refresh via Hydration, 285–286
- deploying applications, 302–305
- forum for, 20
- new application, 287–293
- overview of, 18, 283
- PhoneGap CLI interaction with, 312–315
- quick prototyping, 284–285
- understanding, 283–284

PhoneGap CLI

- anatomy of default application, 310–312
- dealing with API inconsistency, 123
- getting help, 308–309
- interacting with PhoneGap Build, 312–315
- overview of, 307
- project management, 309
- rebirth of, 307–308
- workflow differences, 312

`phonegap create`, **308–309**

PhoneGap Developer app, 148–151

PhoneGap Enterprise, 20

PhoneGap Essentials, xvi

`phonegap remote build`
 android, **313**

Physical devices

- debugging Firefox OS with, 218–220
- debugging in Xcode with, 223–224
- debugging Ubuntu with, 242–243
- deploying PhoneGap Build applications to, 302–305
- deploying Windows Phone apps to, 251
- Firefox App Manager communicating with, 206
- grabbing screen shot of, 192
- Hello World using Bootstrap on Android, 455
- Hello World using jQuery Mobile on, 447–448
- Hello World using Onsen UI on iOS, 467
- Hello World using OpenUI5 on Android, 459
- Hello World using Topcoat on, 442–443
- running Cordova in Visual Studio, 258
- running Cordova in Xcode, 225
- testing Android on, 192–195
- testing Android plugin on, 423
- testing application on, 392–395
- testing iOS plugin, 430
- testing plugin on, 406–407
- testing with Windows Phone, 251–254

platform command, Cordova CLI

- adding platforms, 82–85
- in CLI command list, 76
- defined, 82
- listing platforms, 85
- removing platforms, 86
- updating platforms, 86–87, 104

Platform tools, Android SDK, 170–172**platformOverrides.js file, Visual Studio, 273–274****Platforms**

- adding support for mobile device, 374–375
- adding support with Plugman CLI, 118–120
- adding to existing project, 82–85
- automating project setup with NodeJS across, 162–164
- Brackets Cordova plugin, 476–477

- building Cordova applications for, 16–18

- Cordova API quirks across, 319–320

- Cordova support for Windows, 245

- creating Android plugin, 418

- creating Cordova native plugin, 410

- dealing with API inconsistency, 122–123

- deploying PhoneGap Build applications, 302–305

- developing Cordova applications for multiple, 124–131

- listing supported, 85, 102–103

- registering with manufacturers, 100

- removing with Cordova CLI, 86

- removing with Plugman, 120

- supported by Cordova, 12–13

- supported by GapDebug, 151

- updating, 86–87, 104

Play symbol, Visual Studio, 258**Plugin APIs, documentation, 319****Plugin Development Guide, 397****Plugin Registry. See CPR (Cordova Plugin Registry)****plugin command, Cordova CLI**

- in CLI command list, 76

- plugin search, 91–92

Plugins

- adding Device API to project, 36

- adding to application, 375–376

- adding with Cordova CLI, 87–90

- adding to PhoneGap Build project, 301–302

- Adobe PhoneGap API, 3

- anatomy of, 397–398

- architecture of, 5–8

- for Brackets, 476–479

- building Cordova application, 18, 127–131

- coding applications, 15–16

- ConfigGAP tab for, 299

- as Cordova components, 4

- creating Android native, 414–423

- creating Cordova native, 408–413
 - creating iOS, 424–430
 - creating JavaScript-only, 8–9, 398–403
 - creating with Plugman, 110, 116–118
 - documentation for Cordova, 23
 - forgetting to add critical project, 322
 - listing with Cordova CLI, 90
 - managing using CLI, 319
 - native developers using, 72
 - publishing, 431–434
 - registry for Cordova, 9–10
 - reinstalling in upgraded project, 104
 - removing with Cordova CLI, 90–91
 - restoring with Cordova CLI, 94
 - saving with Cordova CLI, 92–94
 - search path for, 80–81
 - searching for particular, 91–92
 - testing, 403–408
 - troubleshooting PhoneGap Build, 291
 - uninstalling with Plugman, 110
 - using Plugin Registry, 110–116
 - WebStorm Cordova, 482
- Plugins tab, config.xml, 271–272**
- plugin.xml file**
- creating Android plugin, 417–418
 - creating Cordova native plugin, 411
 - creating Javascript-only plugin, 399–401
 - overview of, 398
 - publishing plugins to Plugin Registry, 111–113
 - updated, 118–120
- Plugman CLI**
- command summary, 105
 - creating Javascript-only plugin, 399
 - creating plugins, 69–70, 116–118
 - creating Plugman projects, 105–108
 - installing, 69–70
 - installing plugins, 109–110
 - overview of, 104
 - platform support with, 118–120
 - Plugin Registry, 110–116
 - plugman adduser, 431
 - plugman config get, 114
 - plugman config ls, 114
 - plugman create, 116–117, 409–410
 - plugman info, 114
 - plugman install, 109, 113
 - plugman owner, 115–116
 - plugman platform, 118–120
 - plugman platform remove, 120
 - plugman publish, 431
 - plugman publish pluginfolder, 431
 - plugman search command, 113
 - plugman unpublish plugin_id, 113
 - uninstalling plugins, 110
 - used by Cordova CLI, 71
- Portrait orientation**
- for Hello World using Bootstrap on Android, 455
 - in responsive design, 45–50
 - working with Xcode in iOS, 223–225
- PPA (Personal Package Archive), Ubuntu, 236**
- Preferences**
- config.xml file, 133–134
 - default PhoneGap application, 310–311
 - remotely debugging iOS with Safari, 228
 - StatusBar plugin, 368
- prepare command, Cordova CLI**
- build command calling, 98
 - in build management, 96–97
 - copying code into platform subfolders, 126–127
 - copying platform content into appropriate web content folders, 129, 207–208
 - in Cordova CLI, 75–76
 - creating new Cordova project, 82
 - creating project with Visual Studio, 255

prepare command, Cordova CLI (*continued*)

- editing web application content and, 125
- getting information about prepare process, 74
- hooks folder exposing prepare events, 165–167
- importing Cordova projects, 179–180, 183
- project management with PhoneGap and, 309, 312
- role in managing `cordova.js` file, 108
- testing applications with device emulator, 389
- using with `serve` command to check that web content is current, 100
- working with Gulp and, 498–499
- working with Xcode and, 221–225

Problems report, Brackets, 475–476**Processes, automating Cordova, 164–167****Progress panel, ADT IDE, 185****Project, Cordova**

- build management, 96–98
- creating, 77–82
- creating new, 374
- creating with Visual Studio, 254–255
- displaying information, 94
- executing applications, 98–103
- importing, 180–185
- managing platforms, 82–87
- managing plugins, 87–94
- opening in Visual Studio, 256–258
- upgrading, 103–104

Project Options menu, Visual Studio, 257**Project setup, automating**

- bash script, 160–161
- cross-platform approach using NodeJS, 162–164
- overview of, 157
- Windows command file, 158–160

Projects

- creating Plugman, 105–108
- managing with PhoneGap CLI, 309
- Visual Studio options for new, 266–267

prompt () method, visual notifications, 330–331**Prototyping, PhoneGap Build quick, 284****Proxy settings, 72–74****publish , Plugman, 111–113****Publishing**

- plugins, 431–434
- plugins to Plugin Registry, 111–113
- testing app on physical devices before, 251–254

Q

QR (Quick Response) code

- AppGyver steroids application, 488–489
- PhoneGap Build applications, 302

R

readme.md file, plugins, 112–113, 431–432**Ready to Build button, PhoneGap Build, 288–289****Rebuild All button, PhoneGap Build, 289****Registration**

- Windows Phone Dev Center, 251
- Windows Phone developer, 251–253

Registry, Cordova Plugin, 9**Remote agent, iOS build to simulator in Visual Studio, 275–281****remove (rm) command, 86, 90****Removing**

- platforms, 86
- project plugins, 90–91

Resize function, responsive design, 46–48**Resource filters, ADT IDE, 184****Responsive design,**

- Bootstrap and, 450
- browser and window size and orientation, 45–49
- CSS use in, 45

- examples of portrait and landscape orientation, 49
- resources for, 50
- Responsive Mobile Design: Designing for Every Device (Dutson), 50**
- restore command, Cordova CLI**
 - in Cordova CLI command list, 76
 - `restore plugins`, 94, 104
- Ripple Emulator (Apache Ripple)**
 - debugging Cordova application, 274–275
 - debugging with, 145–148
 - installing, 146
 - viewing application on multiple devices, 380
- Run As menu, ADT IDE, 186, 191**
- Run button**
 - debugging Ubuntu applications, 240–241
 - running Cordova project in Xcode, 224
 - Visual Studio tools for Cordova, 274–275
- run command, Cordova CLI**
 - in Cordova CLI command list, 76
 - running applications on device emulator, 98–99, 207, 476–477
- Run Configurations, 188–190, 482–484**
- Run menu, Eclipse, 493**
- Running, Cordova application, 185–191**

S

- Safari Web Inspector**
 - debugger icons, 232
 - enabling remote debugging of iOS, 228–230
 - setting breakpoints/stepping through JavaScript code, 231–232
 - updating HTML content in, 230–231
 - viewing/editing variable values, 232–233
- Sams Teach Yourself jQuery Mobile in 24 Hours (Dutson), 450**
- Sams Teach Yourself Responsive Web Design in 24 Hours (Kymin), 50**

- SAP Mobile Platform (SMP), 285**
- SAP Mobile Platform Hybrid SDK**
 - author's work with, xxiii
 - over-the-air web content updates for production applications, 285
 - types of hybrid mobile applications, 25
- SAP (OpenUI5), 456–459**
- Sass, Ionic support for, 461–462**
- save command**
 - in Cordova CLI command list, 76
 - `save plugins`, 92–94, 104
- Scale, in responsive design, 45–50**
- Scanner application, Steroids, 486–489**
- Schemes, for Cordova project in Xcode, 224**
- Screen shots, grabbing, 192, 227**
- script tag, Cordova initialization, 31–32**
- Scripts, executing in InAppBrowser, 364–365**
- SDK (software development kit)**
 - building Cordova applications, 1, 16–18
 - installing Android, 59–65
- Search, for plugins using keywords, 433–434**
- Search paths**
 - installing local plugin, 89
 - searching for plugin, 91–92
- Security**
 - PhoneGap Build applications to Android, 302–303, 305
 - Windows app limitations, 247–248
- `_self` parameter, InAppBrowser, 361**
- serve command**
 - in Cordova CLI command list, 76
 - overview of, 100–101
 - PhoneGap, 148–151
- Session Filter, ADT IDE LogCat panel, 188**
- Shell scripts, Plugman, 106–108**
- `show()` method, StatusBar, 368–369**
- Signing keys, troubleshooting PhoneGap Build, 290–291**

Simulator

- debugging iOS with Xcode, 226–227
- debugging on physical device vs., 223
- debugging Ubuntu applications, 237–242
- debugging with Firefox OS, 207–218
- installing Firefox OS, 206–207
- testing application on iOS, 389–390
- testing iOS plugin, 430
- testing PhoneGap Build applications on, 304–305
- using weinre with device, 142–145
- vs. emulators in this book, 135
- Size, Windows Phone emulator, 259**
- Smashing Magazine, 49–50**
- SMP (SAP Mobile Platform), 285**
- Software development kit (SDK)**
 - building Cordova applications, 1, 16–18
 - installing Android, 59–65
- Solution Explorer, 257–258**
- `someOtherFunction` function, 34
- Source code**
 - accessing for plugin, 398
 - Cordova, 4
- Sources pane**
 - Chrome DevTools, 200
 - Ubuntu, 239–241
- Splash screens, creating, 123–124**
- Splashscreen API, 367**
 - `showSplash` function, 367
- Splashscreen plugin, 367**
- Stack Overflow, Cordova support, 20**
- Start Simulator button, Firefox OS simulator, 206**
- Static HTML pages, traditional web applications, 13–14**
- StatusBar** object, 368–370
- StatusBar** plugin, 367–370

Step Into button

- debugging iOS, 232
- debugging Ubuntu, 240–241
- debugging with Firefox OS simulator, 215–216

Step Out button

- debugging iOS, 232
- debugging Ubuntu, 241–242
- debugging with Firefox OS simulator, 215

Step Over button

- debugging iOS, 232
- debugging Ubuntu, 240–241
- debugging with Firefox OS simulator, 215–216

Steroids application, by AppGyver, 486–489

`steroids` command, 486–489

Stop button, Cordova project in Xcode, 224**Style, StatusBar, 370**

`styleDefault()` method, StatusBar, 370

`successCallback` parameter, Cordova native plugin, 7–8, 412–413

Support, Cordova, 20

`Switches`, run command, 99–100

Symbolic links, 82

`_system` parameter, InAppBrowser, 361

System path

- adding Ant to, 62
- ADT installation and, 60
- Android SDK installation and, 61–62
- Ant installation and, 57–58
- `cvs-create.sh` and, 161
- JDK installation and, 54
- Node.js installation and, 65, 76

System requirements, Windows development with Cordova, 249

T

Target platforms

- PhoneGap Build, 284–285
- universal Windows app project, 246
- Visual Studio Run menu, 275–276
- Xcode available device, 224–225

telephone=yes tag, Cordova initialization, 31

Telephony API, 414–423, 426

- Context class, 414
- TelephonyManager class, 414

Testing

- Android on physical device, 192–195
- Android plugin, 420–423
- application in desktop browser, 103
- application with device emulators, 389–392
- Cordova applications, 32, 134–135
- Cordova applications in Visual Studio, 258
- Cordova applications in Xcode, 225–227
- as debugging. *See* Debugging
- debugging tools. *See* Debugging tools
- Git configuration for iOS, 65–66
- iOS plugin, 427–430
- Node installation for iOS, 65
- plugins, 403–408
- separating JavaScript code for debugging, 39

TextEdit, 15–16

The Hitchhiker's Guide to the Galaxy (Adams), 398

Themes

- Topcoat, 439, 442–444
- UI5, 457

Third-party tools

- build, 494–503
- code editors. *See* Code editors, third-party

- code validation, 469–473
- developer productivity enhancement, 485–493
- overview of, 469

Third-party UI frameworks

- Adobe Topcoat, 439–443
- Bootstrap, 450–456
- Ionic framework, 459–464
- jQuery Mobile, 444–450
- Onsen UI, 464–468
- overview of, 437–439
- SAP OpenUI5, 456–459

THyM, productivity enhancement tool, 490–493

`tm.getSimCountryIso()` method, Android plugin, 414–416

`tm.getSimOperatorName()` method, Android plugin, 414–416

Tools

- building Cordova applications, 16–18
- Cordova components, 4–5

Tools menu, Android SDK installation, 59

Topcoat UI framework

- code examples, 404, 420–421
- CSS library for creating UIs for web applications, 439
- Hello World! examples, 440–441
- light and dark themes, 442–443

Troubleshooting

- CLIs, 72–75
- Cordova applications, 32
- creating project/adding platform, 79
- leveraging Cordova APIs, 36
- PhoneGap Build, 289–291

Twitter, creating Bootstrap, 450

U

Ubuntu development with Cordova

- debugging applications, 237–243
- installing Cordova CLI, 235–236
- overview of, 235

UI (User interface)

- building Cordova application with, 1–2
- capabilities of, 10–11
- frameworks. *See* Third-party UI frameworks
- updating in new project, 378–380

UIWebView, remotely debugging iOS, 227**Unknown Sources, enabling on Android, 302–303****Updates**

- Cordova CLI/Cordova project, 103–104
- platform, 86–87, 104
- plugin, 113

USB debugging, 193–197**User notifications, 326–331**

- beep and vibrate methods, 327
- overview of, 326
- visual, 327–331

V

Variables, Web Inspector, 232–233

`--verbose`, 74–75. *See also* `-d` flag

Version attribute, PhoneGap Build, 301–302

`vibrate` method, hardware notifications, 327

Vibration plugin, hardware notifications, 327**Viewport**

- changing settings on Bootstrap, 452
- Cordova initialization, 31, 34

Virtual buttons, 333**Visibility, StatusBar, 368–369****Visual notifications, Cordova APIs**

- `alert()`, 328–329
- `confirm()`, 329–330

overview of, 327–328

`prompt()`, 330–331

Visual Studio, 250–251**Visual Studio Express, 251****Visual Studio Professional, 251, 265****Visual Studio tools, for Cordova**

- `config.xml` file, 271–272
- `index.html` file, 273
- iOS build/deployment in Visual Studio, 275–281
- MDHAE and, 265–266
- New Project dialog, 266–267
- Options dialog, 268–269
- overview of, 265
- Run menu, 274–276
- viewing project, 269–270
- warning page display, 268

Visual Studio, workflow

- controlling Windows Phone emulator, 259–261
- creating project, 254–255
- debugging applications, 262–265
- opening Cordova project, 256–258
- running Cordova application, 258
- testing Cordova applications, 254
- Windows development with Cordova, 254–265

VMware Fusion, 249

W

W3C (World Wide Web Consortium)

- Contacts API, 352
- Cordova APIs, 24
- hardware APIs, 335
- Media Capture API, 345

Watch panel, Visual Studio, 264**Web 2.0 applications, 14**

Web applications

- anatomy of. *See* Anatomy of Cordova application
- building Cordova, 16–18
- building with IDE, 18
- coding Cordova, 15–16
- copying own files during project create, 81–82
- designing for container, 13–15
- editing Cordova content files, 179–180
- executing Cordova, 98–103
- generated files, 4145
- monitoring activity outside of ADT IDE, 191–192
- overview of, 8–9
- running Cordova, 180–185

Web content editors, 15–16**Web design. *See* Responsive design****Web developers**

- using Cordova CLI, 71–72
- Web Developer menu in Firefox, 205

Web Inspector, Safari. *See* Safari Web Inspector**Web Inspector, Ubuntu, 239–240****WebStorm, 473, 479–487****WebView, 2****weinre (Web Inspector Remote)**

- debugging iOS applications, 225–226
- debugging PhoneGap Build, 288–289
- debugging with, 139–145
- installing, 139–140

Whitelist, configuring application, 132**Window events, InAppBrowser, 363–364****Windows**

- alert function unavailable in, 33
- Android SDK configuration in, 60–62
- Ant installation for Android in, 56–59
- automating project setup in, 158–160, 162–164
- executing CLI commands in, 76–77
- Firefox OS simulator issues in, 209
- GapDebug for, 152

installing JDK for Android development, 52–56

installing JSHint on, 472

installing JSLint on, 470

installing Ripple on, 146

iOS development, 65–69

`serve` command security warning, 101

updating Cordova CLI/Cordova projects, 103–104

Windows 8 versions, 248–249**Windows App Store, 251–254****Windows development with Cordova**

- overview of, 245
- system requirements, 249
- Visual Studio tools, 265–281
- Visual Studio workflow, 254–265
- Windows App Store setup, 251
- Windows Phone development tools, 250
- Windows Phone device configuration, 251–254
- Windows Phone limitations/security, 247–249
- Windows vs. WP8 projects, 245–247

Windows Phone

- deploying apps to Windows Phone Store, 251
- development tools, 250
- device configuration, 251–254
- limitations/security restrictions, 247–249
- system requirements for Cordova, 249

Windows Phone 8

- controlling emulator, 259–261
- creating project with Visual Studio, 254–255
- debugging with Visual Studio, 262–265
- running Cordova application, 258
- supported by Cordova, 245
- Windows projects vs., 245–247

Windows Phone Dev Center, 250–253**Windows Phone SDK, 250****Windows platform**

- creating project with Visual Studio, 254–255
- supported by Cordova, 245–246

Windows Store apps, 248–249

Wipe user data checkbox, AVD, 175

Workflow

- Cordova developer, 72
- native development and plugin, 409
- PhoneGap CLI and PhoneGap Build, 313
- PhoneGap vs. Cordova CLI, 312
- Visual Studio. *See* Visual Studio, workflow

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

Writing to console, debugging by, 136–139

WWAHost process, Windows security, 248

www folder

- copying application content from, 96–97
- developing Cordova applications, 124–125
- importing Cordova project, 183

X

Xcode

- application console output in, 392
- iOS command line tools, 63–65

- not using to edit web applications, 223
- opening Cordova project in, 221–223
- testing iOS plugin in, 428–429
- Windows system requirements, 249

Xcode 6 *Start to Finish* (Anderson), 226

Xcode IDE, 63–65

xcodebuild, 63–65

.xcodeproj extension, iOS, 221–223

XHR (XMLHttpRequest) API, Web 2.0 applications, 14

Z

.zip files

- ADT installation, 60
- deploying custom plugins, 431
- PhoneGap Build, 288, 294, 314
- Plugman project with shell scripts, 106
- Topcoat, 439

Zoom, Windows Phone emulator, 259