John M. Wargo

PhoneGap Essentials

Building Cross-Platform Mobile Apps

PhoneGap Essentials

This page intentionally left blank

PhoneGap Essentials

Building Cross-Platform Mobile Apps

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.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Wargo, John M.
PhoneGap essentials : building cross-platform mobile apps / John M. Wargo.
p. cm.
Includes index.
ISBN 978-0-321-81429-6 (pbk. : alk. paper)—ISBN 0-321-81429-0 (pbk. : alk. paper)
1. PhoneGap (Application development environment) 2. Mobile computing. 3. Application software. I. Title.
QA76.59.W37 2012
004—dc23

2012010042

Copyright © 2012 Pearson Education, Inc.

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

ISBN-13: 978-0-321-81429-6
ISBN-10: 0-321-81429-0
Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, June 2012

To my wife, Anna.
This work exists because of your outstanding support.

This page intentionally left blank

Contents

Foreword by Bryce A. Curtis *xiii*
Foreword by Jim Huempfner *xv*
Preface *xvii*
Acknowledgments *xxiii*
About the Author *xxiv*

Part I PhoneGap 1

Chapter 1 Introduction to PhoneGap 3
A Little PhoneGap History 4
Why Use PhoneGap? 5
How PhoneGap Works 6
Designing for the Container 11
The Traditional Web Server (Web 1.0) Approach 11
The Web 2.0 Approach 11
The HTML 5 Approach 12
Writing PhoneGap Applications 13
Building PhoneGap Applications 13
PhoneGap Limitations 17
PhoneGap Plug-Ins 18
Getting Support for PhoneGap 19
PhoneGap Resources 19
Hybrid Application Frameworks 19
Appcelerator Titanium 20
AT&T WorkBench and Antenna Volt 21
BlackBerry WebWorks 21
Strobe 22
Tiggr 22
Worklight 22

Chapter 2 PhoneGap Development, Testing, and Debugging 23
Hello, World! 23
PhoneGap Initialization 25

Leveraging PhoneGap APIs 28
Enhancing the User Interface of a PhoneGap Application 30
Testing and Debugging PhoneGap Applications 35
Running a PhoneGap Application on a Device Simulator or Emulator 35
Running a PhoneGap Application on a Physical Device 36
Leveraging PhoneGap Debugging Capabilities 37
Third-Party PhoneGap Debugging Tools 43
Dealing with Cross-Platform Development Issues 49
API Consistency 50
Multiple PhoneGap JavaScript Files 51
Web Content Folder Structure 51
Application Requirements 52
Application Navigation and UI 52
Application Icons 53

Part II PhoneGap Developer Tools 55

Chapter 3 Configuring an Android Development Environment for PhoneGap 57
Installing the Android SDK 58
Eclipse Development Environment Configuration 64
Creating an Android PhoneGap Project 66
New Eclipse Project 67
Using Command-Line Tools 74
Testing Android PhoneGap Applications 77
Using the Emulator 78
Installing on a Device 78

Chapter 4 Configuring a bada Development Environment for PhoneGap 79
Downloading and Installing the Correct PhoneGap bada Files 80
Creating a bada PhoneGap Project 82
Creating a bada Application Profile 86
Testing bada PhoneGap Applications 95

Chapter 5 Configuring a BlackBerry Development Environment for PhoneGap 97
Installing the BlackBerry WebWorks SDK 98
Creating a BlackBerry PhoneGap Project 99
Building BlackBerry PhoneGap Applications 103
Configuring the Build Process 104
Executing a Build 107

Testing BlackBerry PhoneGap Applications 109
Testing on a BlackBerry Device Simulator 109
Testing on a Device 111

Chapter 6 Configuring an iOS Development Environment for PhoneGap 113
Registering as an Apple Developer 113
Installing Xcode 114
Creating an iOS PhoneGap Project 116
Testing iOS PhoneGap Applications 122

Chapter 7 Configuring a Symbian Development Environment for PhoneGap 125
Installing the Nokia Web Tools 125
Installing the Make Utility 126
Creating a Symbian PhoneGap Project 128
Configuring Application Settings 129
Modifying HelloWorld3 for Symbian 130
Packaging Symbian PhoneGap Projects 131
Testing Symbian PhoneGap Applications 132

Chapter 8 Configuring a Windows Phone Development Environment for PhoneGap 135
Installing the Windows Phone Development Tools 135
Creating a Windows Phone PhoneGap Project 136
Testing Windows Phone PhoneGap Applications 139

Chapter 9 Using PhoneGap Build 141
The Fit 142
Getting Started 142
Configuration 143
Creating an Application for PhoneGap Build 145
Creating a PhoneGap Build Project 146
Upload Options 146
New Project 147
The Build Process 148
Project Configuration 148
Dealing with Build Issues 150
Testing Applications 152
OTA Download 152
Via Camera 152
Debug Mode 153

Part III PhoneGap APIs 155

Chapter 10 Accelerometer 157
Querying Device Orientation 158
Watching a Device's Orientation 162

Chapter 11 Camera 165
Accessing a Picture 165
Configuring Camera Options 176
quality 177
destinationType 178
sourceType 179
allowEdit 180
encodingType 181
targetHeight and targetWidth 181
mediaType 181
Dealing with Camera Problems 182

Chapter 12 Capture 185
Using the Capture API 186
Configuring Capture Options 189
duration 190
limit 190
mode 190
Capture at Work 191

Chapter 13 Compass 205
Getting Device Heading 205
Watching Device Heading 209
watchHeading 210
watchHeadingFilter 213

Chapter 14 Connection 217

Chapter 15 Contacts 223
Creating a Contact 224
Searching for Contacts 236
Cloning Contacts 242
Removing Contacts 242

Chapter 16 Device 243

Chapter 17 Events 249
Creating an Event Listener 249
deviceready Event 250

Application Status Events 251
Network Status Events 254
Button Events 256

Chapter 18 File 263
Available Storage Types 263
Accessing the Device's File System 264
Reading Directory Entries 267
Accessing FileEntry and DirectoryEntry Properties 269
Writing Files 272
Reading Files 274
Deleting Files or Directories 275
Copying Files or Directories 276
Moving Files or Directories 276
Uploading Files to a Server 277

Chapter 19 Geolocation 279
Getting a Device's Current Location 280
Watching a Device's Location 284
Setting a Watch 285
Canceling a Watch 289

Chapter 20 Media 293
The Media Object 293
Creating a Media Object 294
Current Position 297
Duration 297
Releasing the Media Object 298
Playing Audio Files 298
Play 298
Pause 299
Stop 299
Seek 299
Recording Audio Files 299
Start Recording 300
Stop Recording 300
Seeing Media in Action 300

Chapter 21 Notification 307
Visual Alerts (Alert and Confirm) 307
Beep 310
Vibrate 310
Notification in Action 310

Chapter 22 Storage 315
Local Storage 316
SQL Database 317

Appendix A Installing the PhoneGap Files 327
Preparing for Samsung bada Development 329
Preparing for iOS Development 329
Preparing for Windows Phone Development 330

Appendix B Installing the Oracle Java Developer Kit 333
Downloading the JDK 333
Installing the JDK 335
Configuring the Windows Path 335
Confirming Installation Success 336

Appendix C Installing Apache Ant 337
Macintosh Installation 337
Windows Installation 338

Index *341*

Foreword

by Bryce A. Curtis

Everywhere you go, people are using mobile devices to keep in touch with family and friends, to find a nearby restaurant, or to check the latest news headlines. Their phones have become an indispensable part of their lives with applications that bind them closer to each other and the world around them. It's these applications that make their phones truly useful. Most users aren't aware of the underlying technology used to develop their favorite app or how much time it took to write. Instead, they view an application in terms of the benefit it provides them. Therefore, as developers, we are free to select technologies that deliver this benefit in the most efficient manner.

One technology decision that must be made early on when developing an application is whether it is to be written using native or web APIs. Depending upon the application, native APIs may be required to meet the user's expectations. However, for most applications, web technologies consisting of HTML 5, JavaScript, and CSS provide equal user experiences. The advantage of using web APIs is that they are written using web technologies familiar to many developers, thus providing an easier and quicker development process. In addition, since web technologies are standardized, they exhibit fairly consistent behavior across the many different mobile platforms available today, such as Android and iOS phones and tablets.

One significant difference between native and web applications is that the native applications provide extensive access to device features such as the camera and accelerometer, while the web applications are limited to what the device's web browser supports. To bridge this gap between native and web, a new type of application called the **hybrid application** was created. A hybrid application is written using the same web technologies—HTML 5, JavaScript, and CSS—but includes additional code that enables native APIs to be called from JavaScript. It works by wrapping your web code with a web browser and packaging both together to create a native application.

This book focuses on how to develop mobile applications using PhoneGap, which is a popular open source toolkit for building hybrid applications. You investigate the extensive PhoneGap API and learn how to include many of the device features in your applications. It will become apparent that PhoneGap delivers on the promise of a simplified, cross-platform mobile development by enabling you to write your application using web technologies and then packaging it up so that it can be distributed throughout the various app stores and markets. With any luck, your application may even become someone's favorite app.

Bryce A. Curtis, Ph.D.
Mobile & Emerging Technologies
IBM Master Inventor
IBM Software Group

Foreword

by Jim Huempfner

There is no doubt that everything is going mobile—not just because everything can but because it is having a transformational impact on how we live, work, and communicate. Mobile applications have become critical solutions for both businesses and consumers.

As a result, many companies are gravitating toward mobile web as their primary mobile app development technology. If not done correctly, defining, designing, building, and maintaining mobile applications for both evolving multiple OS platforms and the ever-changing device landscape can be difficult, time-consuming, and expensive. Numerous commercial and open source products and frameworks that can potentially simplify mobile application creation and development are coming to the marketplace.

PhoneGap is proving to be one of the most popular solutions in this space, allowing users to quickly and easily build applications that will run on multiple platforms, leveraging your existing web development skill sets (tweaked for mobile development, of course). Because of the emergence of this solution as a frontrunner and the challenges customers face in implementing the technology, John Wargo has written this book to aid developers in the process.

After a decades-long career in various computing technologies, John started to focus on mobile development platforms in 2006 when he began working for RIM, the makers of the BlackBerry handheld devices. When I first met him, he was teaching a group of colleagues and me the ins and outs of BlackBerry development. John has a passion for teaching that is surpassed only by his passion for mobile development, which was demonstrated both loud and clear during the class. You'll see that passion and depth of understanding clearly demonstrated in this book as well.

We were fortunate to hire John to work in AT&T's Mobility Group in 2009. He quickly became my team's go-to expert on mobile development, constantly evaluating technologies and learning new options in this rapidly changing mobile environment. John is a particularly valuable resource in helping our customers define their mobile application strategy and understand their options for mobile development, whether they are using the mobile web, native, hybrid frameworks (such as PhoneGap), or mobile application platforms such as MEAP or MCAP.

Mobile technology professionals will benefit from this book because it provides experienced mobile web developers with everything they need to know to transition their mobile web applications into native mobile applications using PhoneGap. This book walks you through configuring and using the development environments you need to work with PhoneGap plus shows you how to use each of the APIs provided by the framework; it's everything you need to get started developing with PhoneGap.

Success in the rapidly evolving and ever-changing mobility space should not cause fear and frustration of inaction. Rather, we should embrace technology enablers like PhoneGap and resources like this book to bring truly winning solutions to reality.

Jim Huempfner
Vice President
Industry Solutions Practice
AT&T

Preface

This book is about PhoneGap—a really cool technology that allows you to build native mobile applications for multiple mobile device platforms using standard web technologies such as HTML, CSS, and JavaScript. I'd been looking at PhoneGap for several years, and when I finally got a chance to start working with it, I quickly found it to be a really simple and compelling way to build a single application that can run across multiple device platforms.

I knew Java from my work at RIM and from building Android applications. I'd poked around at Objective-C for iOS development and even did some work for Windows Mobile using Visual Basic. The world, however, is no longer focusing on applications for single mobile platforms but instead expects that mobile applications are available simultaneously for all popular mobile device platforms. PhoneGap helps solve that particular problem.

This book is for web developers who want to learn how to leverage the capabilities of the PhoneGap framework. It assumes you already know how to build web applications and are looking to understand the additional capabilities PhoneGap provides. The book highlights the PhoneGap API capabilities and how to use the tools provided with PhoneGap.

To understand the topics covered in this book, you will need to have some experience with one or more of the most popular smartphones. Some experience with smartphone SDKs is a benefit, but I'll show you how to install and use the native SDKs as I discuss each supported platform.

The book is organized into three parts:

- **Part I, PhoneGap**: Includes a very thorough introduction to PhoneGap: what it is, how it works, and more

- **Part II, PhoneGap Developer Tools**: Includes instructions on how to install and use the SDKs and PhoneGap tools for each of the supported smartphone platforms
- **Part III, PhoneGap APIs**: Includes a detailed description of each PhoneGap API plus sample code that illustrates how to exercise the API

Additional information, downloadable code projects, and errata can be found on the book's web site at www.phonegapessentials.com.

When I first proposed this book to my publisher, it had a completely different structure than the book you're reading now. As I started writing, I realized that the structure I'd picked didn't work for people learning PhoneGap. So, I quickly reordered it and broke it into the parts listed earlier. I've tried to take you step-by-step through the things that matter for PhoneGap development. I also tried to make it as complete as possible—and not skip anything related to the topic at hand. This means, for example, that when you get to the chapters on configuring development environments for PhoneGap, you'll see that I cover each supported platform in detail (with the exception of webOS since at the time HP indicated it was going to kill the platform). If you need to write PhoneGap applications for any of those platforms, you'll find the information you need here. If you are focusing on a subset of the supported platforms, you'll find that you will need to skip some chapters, but they'll be there later if you expand the scope of your development efforts. The other PhoneGap books that preceded this one focused primarily on Android and iOS, and that didn't seem right to me.

If you're looking for a no-nonsense, complete guide to PhoneGap, this is it.

Inside the Book

As I worked through the manuscript, I deliberately assessed each topic against the book's title and my goals for the publication. I kept my focus on PhoneGap and eliminated any topic that didn't directly relate.

What you'll find in the book:

- Lots of detailed information about PhoneGap and how PhoneGap works
- Lots of code examples

What you won't find in this book:

- Mobile web development topics (this is a book about PhoneGap, not mobile web development)
- Complete listing of the `phonegap.js` source file

- Expressions or phrases in languages other than English
- Obscure references to pop-culture topics (although there is an obscure reference to Douglas Adams' *Hitchhiker's Guide to the Galaxy* and one blatant reference to Monty Python)
- Pictures of my cats (I have no cats, but a picture of one of my dogs did make it into the book)

As you look through the example code provided herein, it's important to keep in mind that the code was deliberately written to clearly illustrate a particular PhoneGap-related topic or concept. While there are many things a developer can do to write compact and/or efficient code, it's distracting to readers when they have to analyze every line in order to be able to tell what's really going on therein. In this book, the code is expanded to make it as readable as possible. There are, for example, very few instances where JavaScript anonymous functions are used in the sample applications. Although using them would have made the code samples smaller, I deliberately eliminated them (in all but one chapter) for readability purposes.

No effort whatsoever has been made to optimize these examples for speed or compactness. They've been created to teach you the nuances of the PhoneGap APIs, not best practices for web development.

The Challenges in Writing a PhoneGap Book

Writing a book about PhoneGap (and many other mobile technologies) is hard. The writing isn't hard, but keeping up with the changes that occur as you write is hard. For this book, a lot of important and interesting things happened during the writing process, and I found myself regularly rewriting chapters to accommodate recent changes. The good news is that most of the PhoneGap-specific content in here will remain valid for a very long time. It was industry changes and developer tool changes that gave me the most trouble.

Let me give you some examples:

- **Six (or more) versions of PhoneGap**: When I started the book, version 1.0 of PhoneGap had just been released. It seemed that I'd picked the perfect time to start work on a PhoneGap book. It took me just about four-and-a-half months to write the manuscript, and during that time three additional versions of PhoneGap (1.1, 1.2, and 1.3) were released. During editing and all of the post-production work that needed to be done on the manuscript, versions 1.4 and 1.5 were released. I expect that by the time this book makes it onto paper, yet another version of PhoneGap, version 1.6, will have been released.

- **HP killing and then resurrecting webOS**: As I started the manuscript, HP announced it was discontinuing its webOS devices and would be seeking someone to acquire the technology. For that reason, I decided to omit any webOS-related topics from the book. Of course, HP then changed its mind and announced it would be releasing webOS as an open source project. Unfortunately, the announcement was made after I'd finished the manuscript, so you will not find much information about webOS development for PhoneGap in this book. After the book is published, I will try to publish an update that includes information on webOS support.
- **Nokia changed the way it supported web development**: Immediately after I completed the chapter on Symbian development, Nokia released a new version of its Symbian SDK that removed support for testing web applications on the simulator. Readers of this book will need to make sure they deploy an older version of the SDK in order to build and test PhoneGap applications for Symbian.
- **Adding Windows Phone support to PhoneGap**: With the release of PhoneGap 1.2, the development team added partial support for Windows Phone development. This was fortunate since it allowed me to replace the webOS chapter with one on Windows Phone. With PhoneGap release 1.3, the team added full API support for Windows Phone development.
- **Adding BlackBerry PlayBook support to PhoneGap**: In PhoneGap 1.3, the development team added support for the BlackBerry PlayBook. This, of course, completely changed the way the Ant scripts used to build BlackBerry applications worked, and the chapter had to be completely rewritten. The BlackBerry stuff stayed basically the same, but the command-line options changed, and a whole new suite of tools was added to support the PlayBook.
- **Deprecating support for the Symbian OS**: Beginning with version 1.5, the PhoneGap project has removed support for Symbian from the PhoneGap download. You will have to download the code from a separate location if you want to continue to work with Symbian applications.
- **PhoneGap donated to the Apache project**: One of the biggest changes that occurred during this process was Nitobi's announcement that the project was being donated to the Apache Foundation. While not a huge change for the development community, what was difficult was that the project was supposed to get a name change. It was first donated to Apache as DeviceReady, but then because of a conflict with a company with the same name, it was quickly changed to Callback, which was for some bizarre reason later changed to Apache Cordova (named after the street

where Nitobi's offices were located). We've been told that the commercial product will keep the PhoneGap name while the open source project will have a different name, but I'm really not sure how that's going to work out.

- **Nitobi Acquired by Adobe**: Immediately following the previous announcement (actually the next day), Adobe Systems Incorporated (www.adobe.com) announced it was acquiring Nitobi, the company responsible for the PhoneGap project. That meant big changes for PhoneGap since the folks at Nitobi could now focus entirely on the PhoneGap project instead of working on it in their spare time. A while later, Adobe announced it was ceasing development of its Flash product for mobile devices. This was huge news and clearly indicated that PhoneGap now had a very important place in Adobe's mobile strategy.

One of the biggest problems I faced was getting the help I needed when things didn't work or didn't make sense. As an open source project run by volunteers, many of my forum questions went unanswered (and to this day are still unanswered). You can try to get help there, but usually you're on your own (all the more reason to pick up this book).

Code Conventions

I put a few notes and sidebars in the manuscript, but for the most part I kept the manuscript as clean and simple as I could. I did, however, illustrate sample code in two ways.

A code snippet, a section of a complete application, will be represented in the manuscript in the following manner:

```
var d = new Date(heading.timestamp);
hc.innerHTML = "Timestamp: " + d.toLocaleString();
```

The code could stand alone, like a complete function that you could use in your application, but in many cases this type of listing illustrates a piece of code that simply affects one small part of an application.

On the other hand, complete code listings will look like this:

HelloWorld Example

```
<!DOCTYPE HTML>
<html>
<head>
  <title>HelloWorld</title>
</head>
<body>
```

```
  <h1>Hello World!</h1>
  <p>This is a very simple web page.</p>
</body>
</html>
```

In this example, the code shown is a complete, functional application that you can copy into your IDE and use.

Web Resources

I've created a web site for the book: www.phonegapessentials.com (see Figure P-1). The site contains information about the book's chapters but will also contain any errata (ideally none!), reader comments, and more. I will also make the book's source code available so you can test the applications yourself and use the code from this book in your own projects.

I also regularly publish mobile development–related articles to my personal web site at www.johnwargo.com. Check out the site when you get a chance.

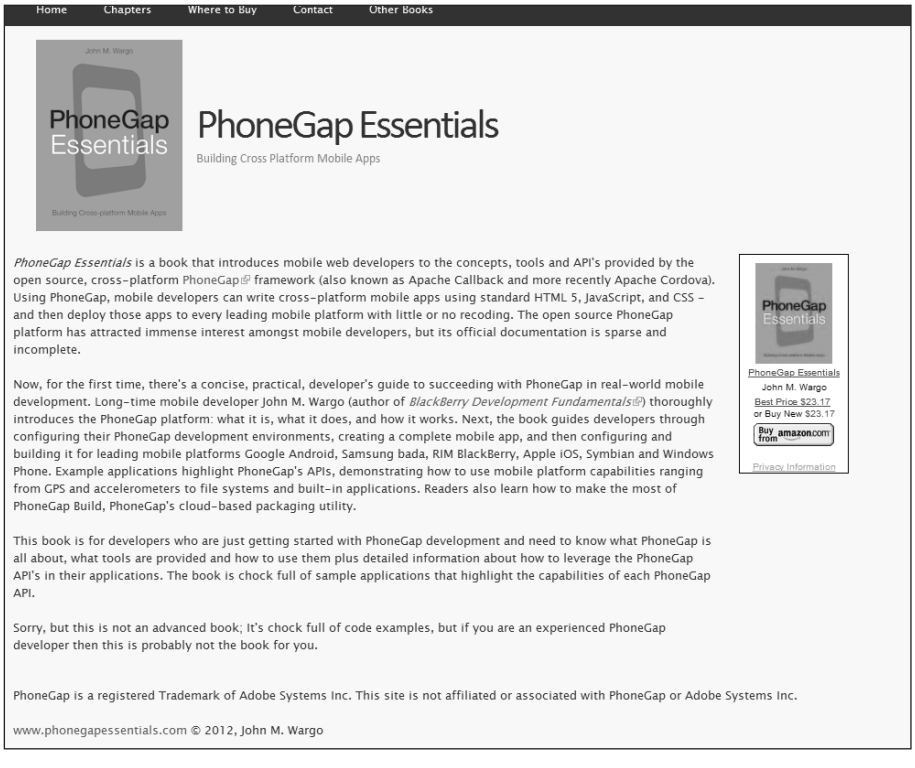

Figure P-1 PhoneGap Essentials web site

Acknowledgments

I want to thank Bryce Curtis for his excellent technical review of the manuscript and his help clarifying some of the issues that cropped up as I worked through the manuscript. There were quite a few places where Bryce added important clarifications that ultimately made this a better book.

Thanks also to the folks at Nitobi (now Adobe) for their help with this book.

Thanks to my managers at AT&T: Abhi Ingle, Jim Huempfner, Shiraz Hasan, and Vishy Gopalakrishnan for supporting me in this endeavor.

Finally, thanks to Greg Doench and the rest of the editorial staff at Pearson Education for their continued support and for letting me do another mobile development book.

About the Author

John M. Wargo has been a professional software developer for most of his career. He spent many years as a consultant and has created award-winning enterprise and commercial software products.

His involvement with mobile development began with a stint at Research In Motion (RIM) as a developer supporting a large U.S.-based carrier and its customers. After leaving RIM, he wrote one of the first books dedicated to BlackBerry development called *BlackBerry® Development Fundamentals* (Addison-Wesley, 2010; www.bbdevfundamentals.com).

He is a technical advisor for *The View*, a magazine for IBM Lotus Domino developers and administrators, and has penned a series of articles on mobile development for that publication.

Until recently, he worked for AT&T as a practice manager in AT&T's Advanced Mobile Applications Practice, specializing in cross-platform development tools and working with customers designing and building both enterprise and consumer mobile applications. He is now part of SAP's Mobile Solution Management team, focusing on the developer experience for SAP's mobile development tools.

2

PhoneGap Development, Testing, and Debugging

As mentioned in the previous chapter, a PhoneGap application can do anything that can be coded in standard, everyday HTML, CSS, and JavaScript. There are web applications and PhoneGap applications, and the distinction between them can be minor or can be considerable. In this chapter, we'll analyze the anatomy of a PhoneGap application, identifying what makes an application a PhoneGap application and then highlighting ways to make a PhoneGap application…better.

The following sections will highlight different versions of the requisite Hello-World application found in every developer book, article, and 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. Refer to the chapters that follow for specific information on how to create and test PhoneGap projects for each of the supported mobile platforms.

Hello, World!

As with any developer book, we're going to start with the default HelloWorld application and then build upon it to highlight different aspects of a PhoneGap application. The following HTML content describes a simple web page that displays some text on a page.

HelloWorld1 Application

```
<!DOCTYPE HTML>
<html>
<head>
  <title>HelloWorld1</title>
</head>
<body>
  <h1>Hello World</h1>
  <p>This is a sample PhoneGap application</p>
</body>
</html>
```

If you package that page into a PhoneGap application and run it on a smartphone or device emulator (in this case an Android emulator), you will see something similar to what is shown in Figure 2-1.

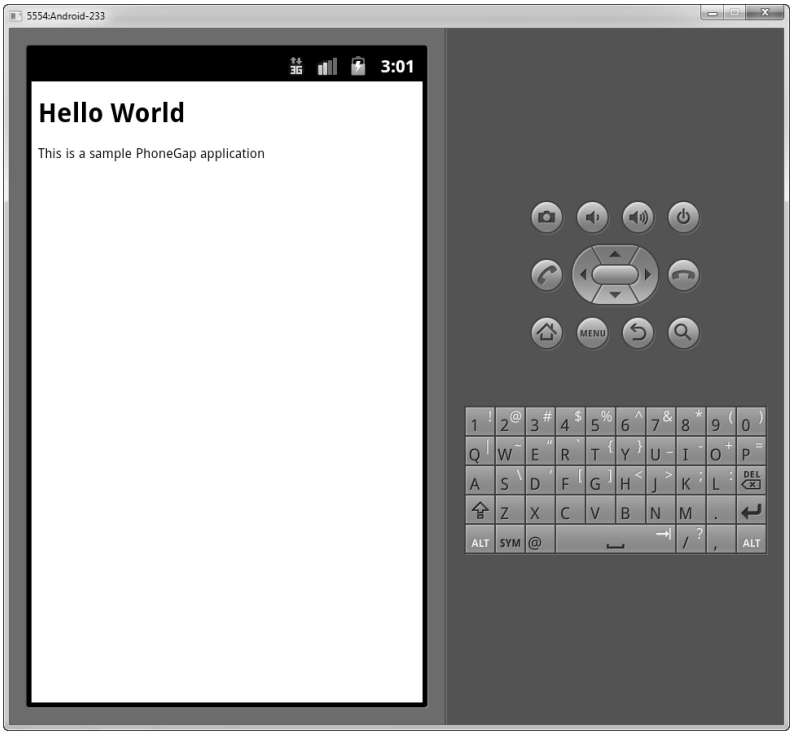

Figure 2-1 HelloWorld1 application running on an Android emulator

This is technically a PhoneGap application because it's a web application running within the PhoneGap native application container. There is, however, nothing

PhoneGap-ish about this application. It's running in the PhoneGap native container, but it isn't leveraging any of the APIs provided by the PhoneGap framework.

Therefore, any web application can be built into a PhoneGap application; there's nothing forcing you to use the PhoneGap APIs. If you have a simple web application that simply needs a way to be deployed through a smartphone app store, then this is one way to accomplish that goal.

PhoneGap Initialization

Now let's take the previous example application and add some PhoneGap-specific stuff to it. The HelloWorld2 application listed next has been updated to include code that recognizes when the PhoneGap application has completed initialization and displays an alert dialog letting you know PhoneGap is ready.

HelloWorld2 Application

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html;
      charset=utf-8">
    <meta name="viewport" id="viewport"
      content="width=device-width, height=device-height,
      initial-scale=1.0, maximum-scale=1.0,
      user-scalable=no;" />
    <script type="text/javascript" charset="utf-8"
      src="phonegap.js"></script>

    <script type="text/javascript" charset="utf-8">

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

      function onDeviceReady() {
        navigator.notification.alert("PhoneGap is ready!");
      }
    </script>

  </head>
  <body onload="onBodyLoad()">
    <h1>HelloWorld2</h1>
    <p>This is a sample PhoneGap application.</p>
  </body>
</html>
```

On the iPhone simulator, the application will display the screen shown in Figure 2-2.

Figure 2-2 HelloWorld2 application running on an iOS simulator

Within the `<Head>` section of the web page are two new entries: meta tags that describe the content type for the application and viewport settings.

The `content-type` setting is a standard HTML setting and should look the same as it would for any other HTML 5 application.

The `viewport` settings tell the 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. 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=1.0`, `maximum-scale=1.0`, 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 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 HelloWorld2 application, it could, for example, consume only the upper half of the screen on some devices.

You may find that on some platforms the settings have no effect. On the BlackBerry Torch simulator, the height and width attributes are respected; on the BlackBerry Storm simulator, the application doesn't consume the entire height of the screen no matter how the attributes are set.

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

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

This loads the PhoneGap API library and makes the PhoneGap APIs available to the program. In this example, and all of the examples throughout the rest of the book, I'll load the PhoneGap JavaScript library using this standard snippet of code. In reality, the PhoneGap file being loaded by your application will include version information in the file name. As I wrote the chapter, PhoneGap 1.0 had just been released, so the code in reality looked like this when I wrote the application:

```
<script type="text/javascript" charset="utf-8"
  src="phonegap-1.0.0.js"></script>
```

As I wrote subsequent chapters, the PhoneGap team released three additional versions of the framework. Rather than have inconsistent PhoneGap JavaScript file names in the book, I chose to just show `phonegap.js` as illustrated in the first example. In reality, many of the example applications used throughout the book were actually built using PhoneGap Build, which requires only the simple `phonegap.js` reference (or no reference at all), which is then replaced with the appropriate JavaScript file version PhoneGap Build is currently using.

Beginning with PhoneGap 1.5, the project team changed the name for the open source project to Cordova and changed the JavaScript file (for most but not all of the supported platforms) from `phonegap.js` to `cordova.js`. So, even though you're working with PhoneGap, the JavaScript file name no longer matches the commercial name for the project.

JavaScript code in a PhoneGap application does not have immediate access to the PhoneGap APIs after the web application has loaded. The native PhoneGap application container must complete its initialization process before it can respond to calls JavaScript made using the PhoneGap APIs. To accommodate this delay in

API availability, a web developer building PhoneGap applications must instruct the container to notify the web application when the PhoneGap APIs are available. Any application processing that requires the use of the APIs should be executed by the application only after it has received its notification that the APIs are available.

In the HelloWorld2 application, this notification is accomplished through the addition of an `onload` event defined in the page's body section, as shown here:

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

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

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

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

```
navigator.notification.alert("PhoneGap is ready!")
```

In production applications, this function could update the user interface (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 the next sample application.

The PhoneGap Navigator

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

Leveraging PhoneGap APIs

Now that we know how to configure an application to wait until the PhoneGap APIs are available, let's build an application that actually uses the PhoneGap APIs as shown in the following HelloWorld3 application.

HelloWorld3 Application

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html;
      charset=utf-8">
    <meta name="viewport" id="viewport"
      content="width=device-width, height=device-height,
      initial-scale=1.0, maximum-scale=1.0,
      user-scalable=no;" />
    <script type="text/javascript" charset="utf-8"
      src="phonegap.js"></script>

    <script type="text/javascript" charset="utf-8">

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

      function onDeviceReady() {
        //Get the appInfo DOM element
        var element = document.getElementById('appInfo');
        //replace it with specific information about the device
        //running the application
        element.innerHTML = 'PhoneGap (version ' +
          device.phonegap + ')<br />' + device.platform + ' ' +
          device.name + ' (version ' + device.version + ').';
       }
    </script>

  </head>
  <body onload="onBodyLoad()">
    <h1>HelloWorld3</h1>
    <p>This is a PhoneGap application that makes calls to the
    PhoneGap APIs.</p>
    <p id="appInfo">Waiting for PhoneGap Initialization to
    complete</p>
  </body>
</html>
```

Figure 2-3 shows a portion of the application's screen when running on the BlackBerry Torch 9800 simulator.

Figure 2-3 HelloWorld3 application running on a BlackBerry 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 PhoneGap used to build the application. Device-specific information is available via the PhoneGap device API (http://docs.phonegap.com/phonegap_device_device.md.html), and this sample application uses only a subset of the available methods in this API.

Figure 2-3 highlights one of the problems with the PhoneGap APIs: inconsistent implementation of an API across different mobile device platforms. The call to the `device.platform` API is supposed to return the name of the mobile device platform the application is running on. In this case, the call should return "BlackBerry," but instead it returns "3.0.0.100" for some bizarre reason. For iOS devices, the call returns "iPhone" when in reality it should be returning "iOS." It's important to keep in mind that any function call might not return what you expect depending on the mobile platform the application is running on. Be sure to test your application on each platform you plan on supporting and make adjustments to your code as needed to deal with inconsistencies with the PhoneGap APIs. Expect the values returned by this property to change over time as well.

Enhancing the User Interface of a PhoneGap Application

As you can see from the application examples highlighted so far, the PhoneGap framework doesn't do anything to enhance the UI of a PhoneGap application. The framework provides access to device-specific features and applications and

leaves it up to developers to theme their applications however they see fit. Web developers should use the capabilities provided by HTML, CSS, and even JavaScript to enhance the UI of their PhoneGap applications as needed; we're not going to cover mobile web UI design here.

As Android and iOS-based smartphones became more popular, web developers found themselves needing to be able to build web applications that mimic the look and feel of native applications on these mobile platforms. To accommodate this need, many open source and commercial JavaScript mobile frameworks were created to simplify this task such as jQuery Mobile (www.jquerymobile.com), Dojo Mobile (www.dojotoolkit.org/features/mobile), and Sencha Touch (www.sencha.com/products/touch).

Although not directly related to PhoneGap development, the use of these frameworks is very common for PhoneGap applications, so it's useful to highlight them here. In this section, we'll discuss how to enhance the UI of a PhoneGap application using jQuery Mobile (jQM), an offshoot of the popular jQuery project. The jQuery and jQM libraries work together to provide some pretty useful UI elements and theming for any mobile web application.

Note: jQM currently supports most of the mobile platforms supported by PhoneGap. As of this writing, the Samsung bada OS has not been tested but is expected to work, and support has not yet been added for the Windows Phone OS.

In the following HelloWorld4 application, we'll take the HelloWorld3 application and apply an enhanced UI using the jQuery Mobile framework.

HelloWorld4 Application

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html;
      charset=utf-8">
    <meta name="viewport" id="viewport"
      content="width=device-width, height=device-height,
      initial-scale=1.0, maximum-scale=1.0,
      user-scalable=no;" />
    <link rel="stylesheet" href="jquery.mobile-1.0b3.css" />
    <script type="text/javascript" charset="utf-8"
      src="jquery-1.6.4.js"></script>
    <script type="text/javascript" charset="utf-8"
      src="jquery.mobile-1.0b3.js"></script>
    <script type="text/javascript" charset="utf-8"
```

```
        src="phonegap.js"></script>

      <script type="text/javascript" charset="utf-8">

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

        function onDeviceReady() {
          //Get the appInfo DOM element
          var element = document.getElementById('appInfo');
          //replace it with specific information about the device
          //running the application
          element.innerHTML = 'PhoneGap (version ' +
            device.phonegap + ')<br />' + device.platform + ' ' +
            device.name + ' (version ' + device.version + ').';
        }
      </script>

    </head>
    <body onload="onBodyLoad()">
      <div data-role="page">
        <div data-role="header" data-position="fixed">
          <h1>HelloWorld4</h1>
        </div>
        <div data-role="content">
          <p>This is a PhoneGap application that makes calls to
          the PhoneGap APIs and uses the jQuery Mobile
          framework.</p>
          <p id="appInfo">Waiting for PhoneGap Initialization to
          complete</p>
        </div>
        <div data-role="footer" data-position="fixed">
          <h1>PhoneGap Essentials</h1>
        </div>
      </div>
    </body>
  </html>
```

Figure 2-4 shows the application running on the Android simulator.

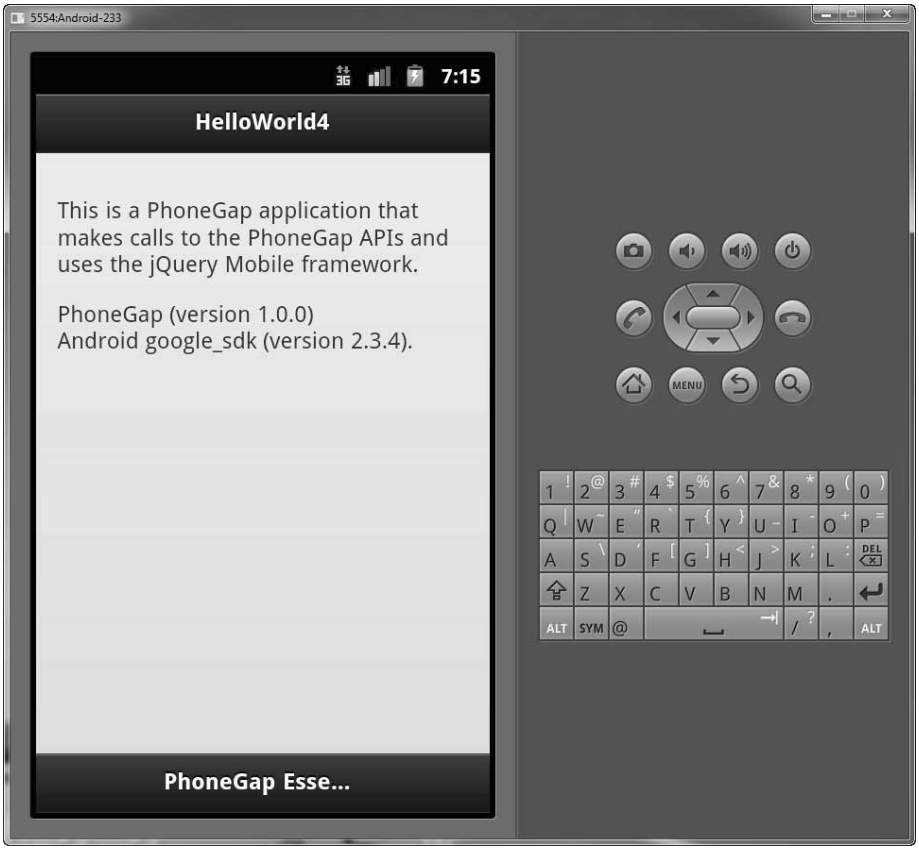

Figure 2-4 HelloWorld4 application running on an Android emulator

Notice that the `device.platform` call is working correctly on the Android emulator; in Figure 2-4, it lists "google_sdk" as the platform for the emulator.

Notice how jQM has a problem rendering the "PhoneGap Essentials" text in the footer. Just so you can see how this looks on a different mobile platform, Figure 2-5 shows the exact same web content running within a PhoneGap application on the BlackBerry Torch simulator. This isn't an issue with PhoneGap but instead is an issue related to available screen width and how jQM renders content leaving space on the left and right for buttons (which aren't used in this example).

Figure 2-5 HelloWorld4 application running on a BlackBerry simulator

In this version of the application, some additional resources have been added to the page's header:

```
<link rel="stylesheet" href="jquery.mobile-1.0b3.css" />
<script type="text/javascript" charset="utf-8"
  src="jquery-1.6.4.js"></script>
<script type="text/javascript" charset="utf-8"
  src="jquery.mobile-1.0b3.js"></script>
```

The first line points to a CSS file provided by the jQM framework. It contains the style information used to render the iPhone-ish UI shown in the figure. Next come references to the jQuery and jQuery Mobile JavaScript libraries that are used to provide the customized UI plus add additional capabilities to the application. The files referenced in the example application are the full versions of the CSS and JavaScript files. These files are used during testing of the application and should be replaced with the min versions of the files, as shown in the following code snippet, before rolling the application into production.

```
<link rel="stylesheet" href="jquery.mobile-1.0b3.min.css" />
<script type="text/javascript" charset="utf-8"
  src="jquery-1.6.4.min.js"></script>
<script type="text/javascript" charset="utf-8"
  src="jquery.mobile-1.0b3.min.js"></script>
```

The min versions are compressed so comments, white space, line breaks, and so on, are removed from the files. This allows the files to take up less space within the packaged application, helping reduce the overall file size for the application, and allows these resources to load more quickly when the user launches the application.

The body of the HTML page has been updated to include several HTML `<div>` tags wrapped around the content for the application. These `<div>`s include a `data-role` attribute that is used by jQM to define specific areas of the content page that are then styled appropriately depending on which role is assigned.

In Figure 2-5, the content in the section of the page given the `header` data-role is styled with a gradient background and forced to the top of the page by the `data-position="fixed"` attribute. Similarly, the content in the section of the page given the `footer` data-role is styled with a gradient background and forced to the bottom of the page by the `data-position="fixed"` attribute. The page content defined within the `data-role="content"` `<div>` will be rendered between the header and footer, with the middle section scrollable as needed to display all of the content within the section.

These examples only lightly cover the capabilities of jQM; there's so much more you can do with this framework to enhance the user experience within your PhoneGap applications. Refer to the jQM online documentation or several of the new books on jQM for additional information about the capabilities provided by the framework.

Testing and Debugging PhoneGap Applications

Each of the mobile platforms supported by PhoneGap has a mechanism a developer can use to test and, in the unlikely event your code has bugs, debug PhoneGap applications. In general, you can load a PhoneGap application into a device simulator or emulator, provided as part of the mobile platform's SDK, or you can load an application onto a physical device. There are also third-party solutions you can use to test your PhoneGap applications within a desktop browser interface.

Running a PhoneGap Application on a Device Simulator or Emulator

Each smartphone operating system supported by PhoneGap has a device emulator or simulator (E/S) provided by the originator of the OS. In some cases, what's provided is a generic emulator that simply mimics the capabilities of the specific OS version, while for other mobile platforms there are simulators available that mimic specific devices. Either way, there's a software-only solution available

that developers can use to test PhoneGap applications in an almost real-world scenario (I'll explain "almost real-world" in the following section).

An E/S is typically included with the native development tools for each mobile platform, but in some cases there are options that can be downloaded individually. Research In Motion, for example, includes a set of simulators with each BlackBerry Device Software version SDK but also provides individual downloads for specific BlackBerry Device Software versions or for older devices that have software updates available for them. Either way, there are likely options available for each and every device or device OS you want to test your application on. The chapters that follow provide detailed information on how to configure a development environment for each of the mobile devices platforms supported by PhoneGap. That's where you will find instructions on how to test PhoneGap applications on the appropriate E/S for the target platform.

Running a PhoneGap Application on a Physical Device

While the device E/S is a great option for developer and system testing of a PhoneGap application, final testing should always be performed on a physical device. As good as these options are, there is always something that doesn't work quite right on a simulator or emulator.

To run a PhoneGap application on a physical device, you will create the PhoneGap application first using the native SDK or package the application for platforms that use a widget approach. Once you have a deployable application, you will connect the device to your development computer via a Universal Serial Bus (USB) cable and transfer the application to the mobile device using some component of the native SDKs. The process varies depending on the mobile platform you are working with.

For iOS applications, for example, Apple requires a special provisioning process for every iOS device on which you want to install your application. The process requires membership in Apple's developer program and involves the Xcode development environment, Apple's developer portal, a provisioning profile, and a physical device.

For Android and BlackBerry devices, the native SDK includes command-line utilities you can use to copy an application to a target device. There's no special provisioning process; you simply connect the device to the developer computer, issue the command, and test away. In some cases, you can deploy to devices directly from the Eclipse IDE. For Android devices, there are steps you must complete to configure the device for testing applications. On BlackBerry, you'll need to secure a set of signing keys (they're free at https://bdsc.webapps.blackberry.com/java/

documentation/ww_java_getting_started/Code_signing_1977871_11.html) and sign the application before it will run on a physical device.

Regardless of the platform you use, digging into the details of on-device testing is beyond the scope of this book. Please refer to the documentation for the appropriate native SDK for additional information about how to test applications on physical devices.

Leveraging PhoneGap Debugging Capabilities

As mentioned earlier, there are two types of PhoneGap applications: PhoneGap applications that consist of a web application packaged inside of a native application container (for Android, BlackBerry, iOS, and Windows Phone) and PhoneGap applications deployed simply as packaged web applications (on bada, Symbian, and webOS).

For the mobile platforms where PhoneGap applications are simply packaged web applications, the freely available native SDK typically includes support for debugging web content running in a device emulator or simulator. In the chapters that follow, you will find instructions on how to leverage native debugging tools for these platforms when testing PhoneGap applications. You will, however, need to refer to the native SDK documentation for detailed information on how to use these tools.

The problem with the native application options for PhoneGap is that the native tools designed to help developers debug applications for each platform are designed to debug native applications; they have none or limited capabilities for debugging web content that is packaged within native applications. The BlackBerry WebWorks development tools originally supported the ability to debug web content packaged within a BlackBerry WebWorks application (which is essentially what a PhoneGap application is on BlackBerry). In 2011, RIM abandoned the Eclipse and Visual Studio IDEs and switched to an entirely command-line-driven approach.

To help debug your PhoneGap applications, you can fill your code with calls to the `alert()` function. This is what I have always called a poor man's debugger, but it works quite well for certain types of application debugging tasks. If you see an event that's not firing within your application or some variable that's not being set or read correctly, you can simply insert an alert that displays a relevant message and use that to see what's going on. There's an example of this approach in the HelloWorld2 application shown earlier with the use of PhoneGap's `navigator.notification.alert("")` function. In this case, I used the alert to help debug what was happening in the `onDeviceReady()` function. It seemed to

be working on Android, but not BlackBerry, so I used the alert to help confirm my suspicion and to help test different approaches as I attempted to fix the problem.

The problem with this approach is that when you fill your buggy code with alerts, you're constantly interrupting the application flow to dismiss the alerts as they come up. For a simple problem, this approach works pretty well, but when debugging more troublesome errors, you will need an approach that allows you to let the application run and then analyze what is happening in real time or after the application or a process within the application has completed, without interrupting the application. PhoneGap applications can do this through the JavaScript `console` object implemented by the WebKit browser rendering engine.

Using the `console` object, developers can write messages to the browser's console, which can be viewed outside of the running program through capabilities provided by the native SDKs or device simulators or emulators. The `console` object has scope at the window level, so it's essentially a global object accessible by any JavaScript code within the application. WebKit supports several options; the most common ones used are as follows:

- `console.log("message");`
- `console.warn("message");`
- `console.error("message");`

Example 2-1 shows a sample application that illustrates the use of this feature.

Example 2-1

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,
      height=device-height, initial-scale=1.0,
      maximum-scale=1.0, user-scalable=no;" />
    <meta http-equiv="Content-type" content="text/html;
      charset=utf-8">
    <script type="text/javascript" charset="utf-8"
      src="phonegap.js"></script>

    <script type="text/javascript" charset="utf-8">

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

```
      function onDeviceReady() {
        //Just writing some console messages
        console.warn("This is a warning message!");
        console.log("This is a log message!");
        console.error("And this is an error message!");
      }

    </script>
  </head>
  <body onload="onBodyLoad()">
    <h1>Debug Example</h1>
    <p>Look at the console to see the messages the application
    has outputted</p>
  </body>
</html>
```

As you can see from the code, all the application has to do is call the appropriate method and pass in the text of the message that is supposed to be written to the console.

In some cases, the browser component executing your application's web content won't throw an error if you try to do something that's not supported in your JavaScript code (calling a PhoneGap API function that doesn't exist, for example, because you've misspelled it). In this scenario, simply wrap the errant call in a `try/catch` block so your application will have a chance to write its error to the console, as shown in the following example:

```
try {
  console.log("Validating the meaning of life");
  somefunctioncall("42");
} catch (e) {
  console.error("Hmmm, not sure why this happened here: " +
    e.message);
}
```

Figure 2-6 shows the messages from Example 2-1 highlighted in the Xcode console window. This window is accessible while the program is running on an iOS simulator, so you can debug applications in real time.

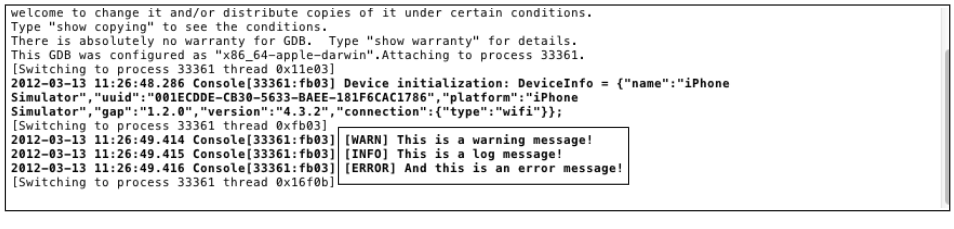

Figure 2-6 Viewing console messages in Xcode

When working with the BlackBerry simulator, you can access the logs by holding down the simulator's Alt key and typing **lglg**. The simulator will display the Event Log, as shown in Figure 2-7.

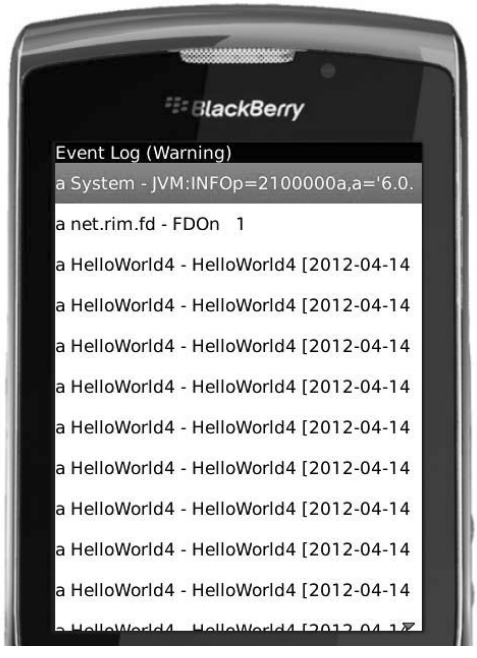

Figure 2-7 BlackBerry Event Log application

When you open an Event Log entry, you can see the details behind the entry, as shown in Figure 2-8. Press the keyboard's n and p keys to navigate to the next and previous entries in the log.

Figure 2-8 Viewing console messages on BlackBerry

Note: In my testing with the BlackBerry simulator, only the `console.log` messages appear within the Event Log application; the BlackBerry implementation of the WebKit engine doesn't seem to respond to `console.warn` and `console.error` messages.

Within the BlackBerry Event Log application, you have the ability to clear the log, filter what's displayed in the log, and copy the contents of the log to the clipboard so you can use them in another application or send them to yourself via email. Additionally, when working with a physical device, you can connect the device to your development system and use the BlackBerry Java Loader application (`javaloader.exe`) to copy the logs from the device. Many of these options are described in detail in my other mobile development book, *BlackBerry® Development Fundamentals* (www.bbdevfundamentals.com).

The Android SDK includes utilities that allow a developer to monitor log activity, while an application runs within an Android emulator. This functionality is provided by the LogCat utility, which is an integral part of the Eclipse plug-in but also available through the command line or a stand-alone utility.

To open the LogCat window in Eclipse, open the Window menu, select Show View, and then select Other. In the dialog that appears, expand the Android category and select LogCat, as shown in Figure 2-9, and then click OK. Eclipse will open a new pane in the messages area of the IDE, as shown in Figure 2-10.

Figure 2-9 Eclipse Show window dialog

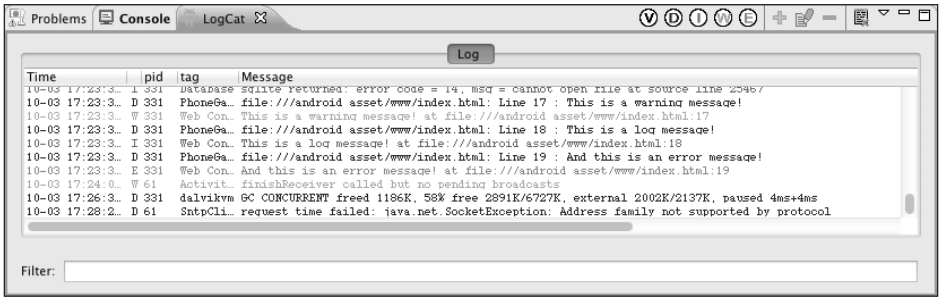

Figure 2-10 Eclipse messages area

This pane will display all messages generated by the Android device emulator as well as console messages written by your PhoneGap application; you can see the three messages written by the sample application. Use the V (verbose), D (debug), I (info), W (warning), and E (error) buttons at the top of the pane to filter the contents of the pane as needed to allow you to more quickly locate the entries you are looking for while debugging an application.

Google also offers a stand-alone utility called the Dalvik Debug Monitor Server (DDMS) that you can use to monitor the Android emulator console when testing PhoneGap applications outside of the Eclipse IDE. To launch the DDMS utility, you must first launch an Android emulator. Once the emulator is running, open a file explorer (Finder on Macintosh or Windows Explorer on Windows), navigate to the Android SDK tools folder, and execute the DDMS utility located therein. The file is called `ddms.bat` on Microsoft Windows and `ddms` on Macintosh.

When the utility launches, it will display a window similar to the one shown in Figure 2-11. At the top of the utility are windows that show the different processes running in the emulator on the left and a list of additional options on the right. The lower half of the application's window displays the same LogCat pane from the Eclipse plug-in.

To access the LogCat content from the command line on Windows, open a command prompt, navigate to the Android SDK `platform-tools` folder, and issue the following command:

```
adb logcat
```

On Macintosh, open a terminal window, navigate to the Android SDK `platform-tools` folder, and issue the following command:

```
./adb logcat
```

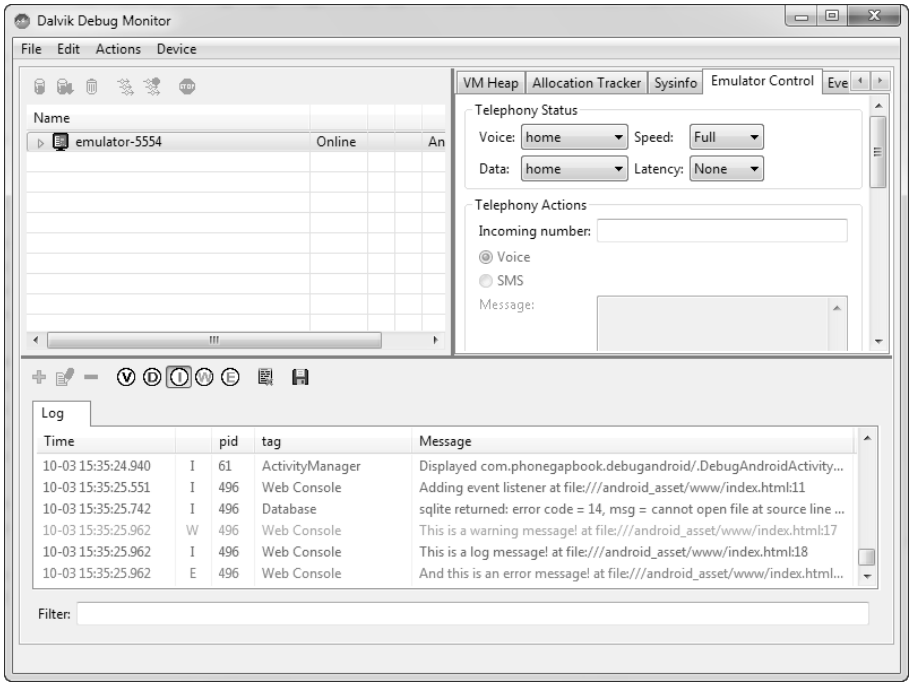

Figure 2-11 Android DDMS application window

The `adb` utility will connect to the emulator and retrieve and display in real time the contents of the logcat from the Android emulator, as shown in Figure 2-12. In the figure, the three console messages generated by the application are highlighted.

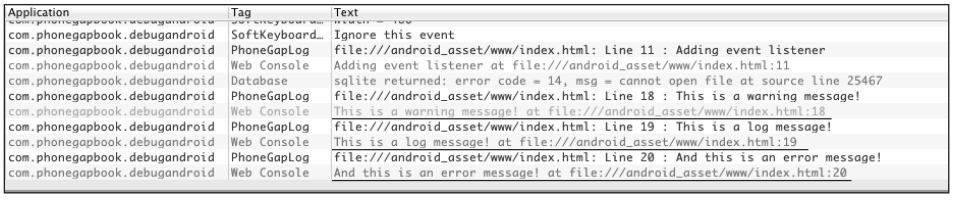

Figure 2-12 Viewing console messages on Android

Third-Party PhoneGap Debugging Tools

There's a very active partner community supporting PhoneGap with additional tools for PhoneGap developers. In this section, I'll introduce several of the available tools that help developers test and debug PhoneGap applications. This is by

no means a complete list of options; refer to the PhoneGap wiki (http://wiki.phonegap.com) for information on additional tools that might be available.

Ripple Mobile Environment Emulator

When developing a PhoneGap application, it's quite time-consuming to build the application and load it into a simulator or emulator for testing. The Ripple Mobile Environment Emulator (RMEE) is a freely available tool that helps alleviate this problem by providing a desktop browser–based interface you can use to test your PhoneGap applications. The RMEE emulates the execution of the PhoneGap APIs within the browser container. You should use the emulator for quick testing of PhoneGap application features during development and then switch to packaging/building PhoneGap applications and testing them on actual devices or device emulators or simulators for more thorough testing. The RMEE is not designed to replace testing on real devices, device simulators, or device emulators.

RIM and the Ripple Emulator

Tiny Hippos, the company that produced of the Ripple Mobile Environment Emulator, was recently purchased by RIM and is expected to become the default way to test BlackBerry WebWorks applications. The emulator has supported PhoneGap for quite a while and is expected to continue to support the project under RIM's ownership.

The RMEE is implemented as an extension to the Google Chrome browser, so before you can start using the emulator, you must first install the latest version of Chrome from www.google.com/chrome. Once you have Chrome up and running, launch the browser and navigate to http://ripple.tinyhippos.com. From the Tiny Hippos home page, click the Get Ripple link, and follow the prompts to install the latest version of the emulator.

Before you can start emulating PhoneGap within the RMEE, you must first configure the browser to allow the emulator access to files on the local file system. Open the Chrome browser, right-click the Ripple icon to the right of the browser's address bar, and select Manage extensions. The browser will display a page similar to the one shown in Figure 2-13. Enable the "Allow access to file URLs" option for the RMEE as shown in the figure and then close the page by clicking the X to the right of the Extensions tab.

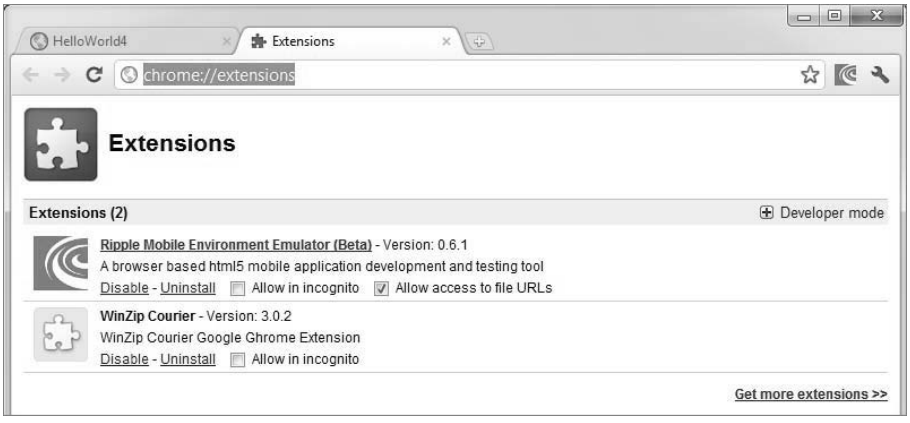

Figure 2-13 Configuring the Ripple Emulator in Google Chrome

Once the browser has been configured, open your application's `index.html` file in the browser. You can press Ctrl+O on Windows or Command+O on Macintosh to open the File Open dialog. Once the page has loaded, you need to enable Ripple for the selected page. To do this, complete one of the following options:

- Click the Ripple icon to the right of the browser's address bar to open a window, allowing you to enable Ripple for the loaded page.
- Right-click the page, open the Emulator menu, and then select Enable.
- Append `?enableripple=true` to the file URL to enable Ripple directly within the address bar when loading an application's `index.html` file.

Once the RMEE is enabled for the loaded page, the browser will display a page, shown in Figure 2-14, that prompts you to identify which type of emulation you want to enable for this page. As you can see, the RMEE supports PhoneGap plus several other platforms and frameworks. Click the PhoneGap button to continue.

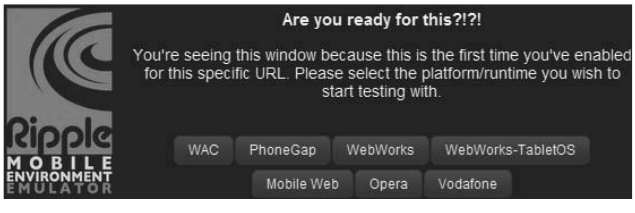

Figure 2-14 Enabling PhoneGap emulation in the Ripple emulator

At this point, the RMEE will display a page with the content from the `index.html` file rendered within the boundaries of a simulated smartphone screen, as shown in Figure 2-15. Wrapped around the simulated smartphone are properties panes that can be used to configure options and status for the simulated smartphone such as simulated device screen resolution, accelerometer, network, geolocation, and more.

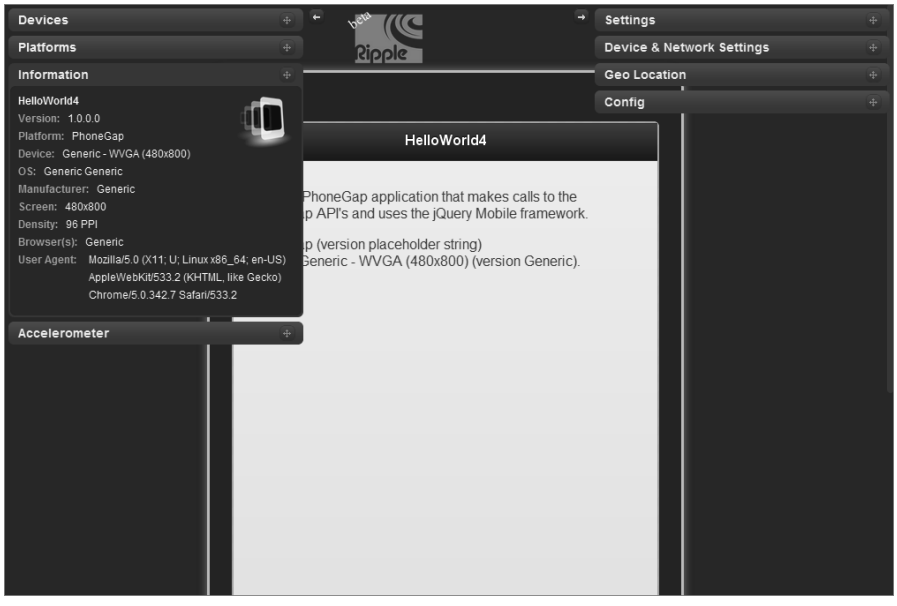

Figure 2-15 PhoneGap application running in the Ripple emulator

You can click each of the tabs to expand the options for the tab and make changes to the simulated device's configuration. At this point, you would simply click around within the simulated smartphone screen and interact with the options presented within your application. When you find a problem or a change you want to make within the PhoneGap application, simply return to your HTML editor, make the necessary changes, write the changes to disk, and then reload the page in the Chrome browser to continue with testing.

Weinre

Web Inspector Remote (Weinre) is a community-built remote debugger for web pages. It has been donated to the PhoneGap project and is currently implemented as part of the PhoneGap Build service. You can find the download files and instructions at http://phonegap.github.com/weinre. Weinre consists of a debug server, debug client, and debug target. The debug server runs on Macintosh or Windows, and the debug client runs in any compatible desktop browser.

For PhoneGap development, it allows a developer to debug a web application on physical device or a device emulator or simulator. To configure Weinre, perform the following steps:

1. Install a debug server on a desktop computer.
2. Launch the debug server.
3. Windows only: Point a compatible desktop browser at the debug server.
4. Connect the remote web application to the server.

The server component of Weinre consists of a stand-alone Macintosh executable or a Java JAR file for Windows. On Macintosh, load the debug server by double-clicking the application's executable in Finder. The debug server and debug client are packaged together in the same application, so there are no additional steps needed to launch the debug client.

On Windows, the debug server consists of a single JAR file that, assuming Java is on the Windows Path, can be loaded using the following command:

```
java -jar path/to/weinre.jar
```

There are additional command-line options that can be passed to the JAR file while it's loading to allow you to configure the host address the server will respond to, the server port number, and more. Refer to the Weinre documentation for additional information about the available command-line options. When the server starts, it will display a message indicating the URL to use to start the debug client; by default it should be http://localhost:8080. Point a compatible WebKit-based browser at the server to open the debug client.

To connect the PhoneGap application to the debug server, add the following `<script>` tag to the `<body>` section of the application's `index.html` file,

```
<script src="http://debug_server:8080/target/
  target-script-min.js"></script>
```

replacing the `debug_server` portion of the URL with the correct host name or IP address for the debug server. This provides the code needed for the PhoneGap application to upload information to the debug server. The Android emulator does not have the ability to connect to host-side resources using an IP address, so for the Android emulator, you must use the host address http://10.0.2.2, as shown in the following example:

```
<script src="http://10.0.2.2:8080/target/
  target-script-min.js"></script>
```

Note: Be sure to remove the Weinre `<script>` tag from your PhoneGap application before releasing it into production. The application will likely hang if attempting to connect to debug server that isn't available.

Figure 2-16 shows the debug server running on a Macintosh. On the bottom of the window are tabs that control the server while the toolbar on the top of the window contain options for the remote debugger client.

Figure 2-16 Weinre server/debug client on a Macintosh

The debug client provides the means to view and optionally manipulate many of the page elements and other aspects of your application's web content. You can view the browser console, as shown in Figure 2-17, to see console messages written by the PhoneGap application, or you can change application values or properties to tweak the application while it's running.

The available documentation for Weinre is pretty light, but since the project's capabilities are based upon the Google Chrome Developer Tools, you can find additional information on the Google Code web site at http://code.google.com/chrome/devtools/docs/overview.html.

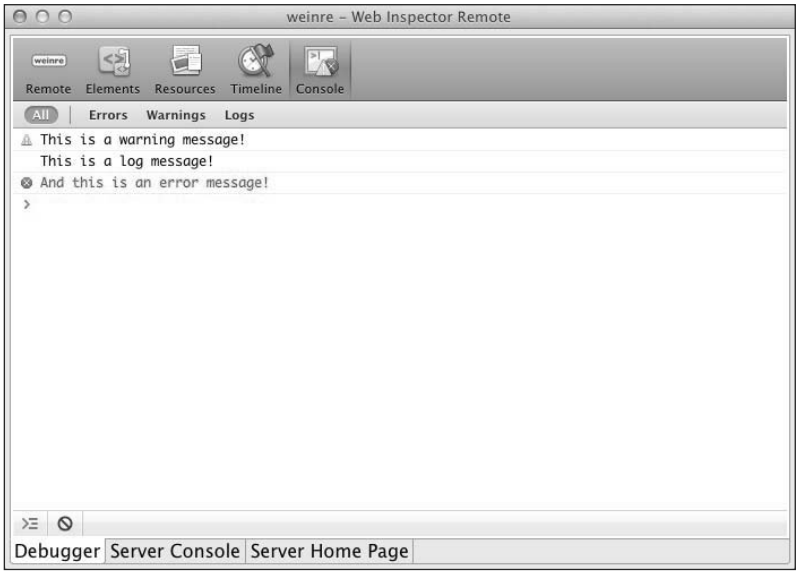

Figure 2-17 Weinre debug client console

Dealing with Cross-Platform Development Issues

As interesting as all of these PhoneGap capabilities are, there are a lot of issues that make cross-platform development tasks difficult. The PhoneGap project is supported by developers from all over the world, including developers who may have experience with only one or a small number of mobile platforms and developers who have a strong opinion about how something should be done. The problem with this is that when you take development projects written by different people and try to collect them into a single framework, you can bump up against inconsistencies. Add to this that every mobile platform supported by PhoneGap is different and has different ways of doing things, and you have a difficult task to make everything work cleanly and seamlessly.

Note: To the PhoneGap project's credit, things move pretty quickly, and the issues I'm complaining about here could very well be fixed in any subsequent release of the framework. Be sure to check the latest documentation before working around any of the issues listed in the sections that follow.

Let's look at some examples.

API Consistency

Figure 2-18 shows the supported feature matrix from the PhoneGap web site; you can find the page at www.phonegap.com/about/features. As you can see, the table is pretty complete; there are some gaps, but it's more full than empty. On the other hand, since PhoneGap is supposed to be a cross-platform framework, the gaps in this table make it very hard to truly create a cross-platform application using those APIs. If a particular feature you want to use in your application is supported on only some mobile platforms, then you'll have to make special accommodation within your application for platforms that do not support the particular API.

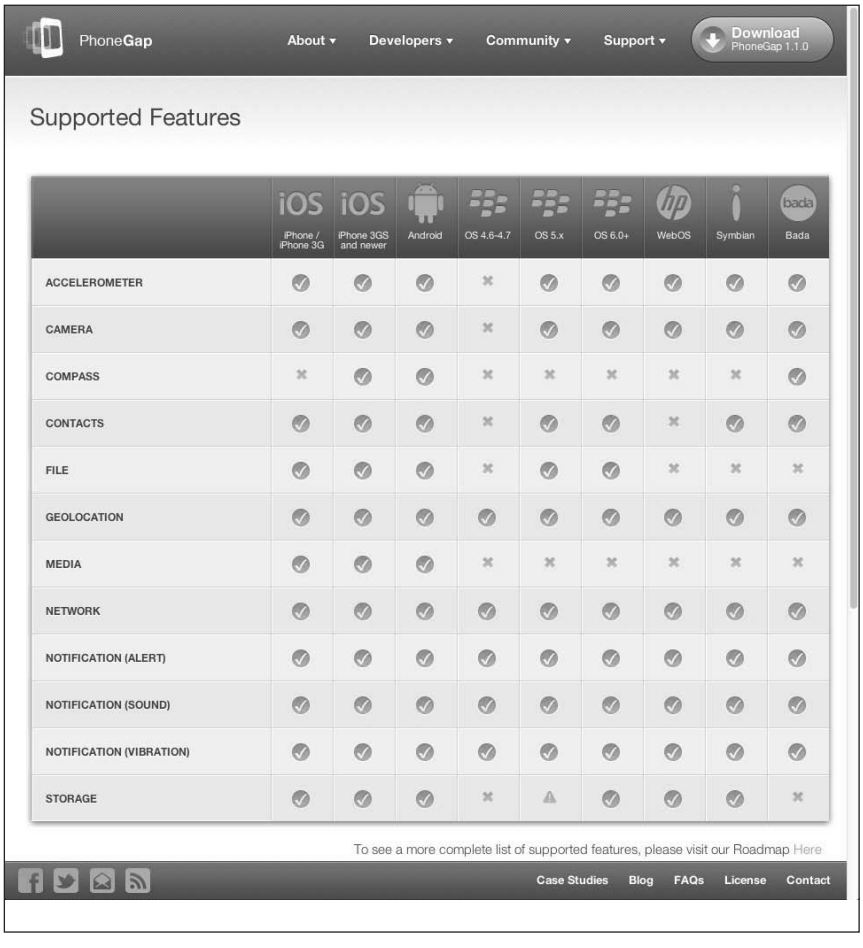

Figure 2-18 PhoneGap-supported feature matrix

Another problem arises when you look through the API documentation found at http://docs.phonegap.com/. For most of the PhoneGap APIs, the documentation lists that the APIs are supported only on Android, BlackBerry, and iOS devices. It's likely the issue here is that the PhoneGap developers are like most developers and don't like to write (or update) documentation; the impact on you is huge. Do you rely upon the API documentation? Do you instead ignore the documentation and use feature matrix as the correct reference? Or do you cover your bases and assume it is all wrong and test everything?

No matter what, this can be quite a challenge; ideally the PhoneGap project team will get more organized and make sure all of the documentation is up-to-date as each new version is released.

Multiple PhoneGap JavaScript Files

One of the first issues I bumped up against when learning to do cross-platform PhoneGap development was that the PhoneGap JavaScript library is different between mobile platforms. So, the JavaScript code within the PhoneGap JavaScript file for BlackBerry projects is different from what is found in the PhoneGap JavaScript file for Android projects. My original thought when I started was that I could just copy the web content folder between projects, build the application, and be done, but since each platform's JavaScript file is different, I would have to copy over the web content and then also make sure the correct PhoneGap JavaScript file was in the folder as well.

To make things work, with earlier versions of the PhoneGap framework, the BlackBerry and bada PhoneGap JavaScript libraries had different file names than on other platforms. This has supposedly been fixed, but you better check to make sure when building applications.

Web Content Folder Structure

As you will see in the chapters that follow, in some cases, some PhoneGap project developers have created nonstandard project folder structures for PhoneGap projects. For example, for a typical Symbian project (described in Chapter 7), the application's web content files would normally be placed in the root of the project's folder structure. The HTML, JavaScript, and CSS files should be placed right at the top of the folder hierarchy, so they can be easily accessed when working with the project. For some bizarre reason, the PhoneGap project places the files in a `framework/www` folder, complicating the project's folder structure and making it more difficult to get to the application's content files.

Application Requirements

One of the things you might bump into as you build cross-platform PhoneGap applications is the need to supply additional files in your application to accommodate the requirements for a particular OS version. For example, in the default PhoneGap project for iOS, you will find the following note:

```
<!-- If your application is targeting iOS BEFORE 4.0 you MUST
put json2.js from http://www.JSON.org/json2.js into your www
directory and include it here -->
```

Apparently a feature was added in PhoneGap 0.9 that requires the use of the `JSON.stringify()` function, so you will have to make sure you include the appropriate JSON library in your application. This further complicates a developer's ability to use an application's web content across multiple device platforms since an iOS application in this example might have additional lines of code needed to support this iOS-specific feature.

Application Navigation and UI

Mobile device platforms typically share some common elements but at the same time implement unique features or capabilities that help set them apart from competitors. The Android and iOS operating systems support many of the same features but sometimes implement them in a different way. Because of this, any mobile application designed to run on different mobile operating systems must take into consideration the differences between mobile platforms.

As you build PhoneGap applications for multiple mobile device platforms, you will need to implement different UI capabilities on different operating systems. On the Android and BlackBerry platforms, there's a physical Escape button that can be pressed to return to a previous screen; on iOS, there will need to be a back button added to the bar at the top of the application screen.

Because of this, a PhoneGap application will need to either contain additional code that checks to see what platform it's running on and update the UI accordingly or it will need to have different versions of the application's web content depending on which OS the application is running on. Neither approach is easy. There are several books on mobile web development available that deal directly with these types of issues.

Application Icons

Each mobile platform and often different versions of a particular device OS have different requirements for application icons. A developer building PhoneGap applications for multiple device platforms will need to be prepared to create a suite of icons for their application that addresses the specific requirements for each target device platform and/or device OS. The PhoneGap project maintains a wiki page listing the icon requirements for the different supported operating systems at http://wiki.phonegap.com/w/page/36905973/Icons%20and%20Splash%20Screens.

Additionally, for some devices on some carriers (older BlackBerry devices, for example), the mobile carrier applies a specific theming to the OS in order to help distinguish themselves in the market. Any application icon designed for one of these devices will need to accommodate, as best as possible, rendering pleasantly within different themes.

This page intentionally left blank

Index

A

Accelerator API
  overview of, 157–158
  querying device orientation, 158–161
  watching device orientation, 161–164
`addresses` array, specifying contact properties, 225–226
Adobe, in history of PhoneGap, 5
Alerts
  debugging PhoneGap applications and, 37–38
  Notification API and, 307–310
`allowEdit` property, Camera API, 180
Android
  accelerator determining device orientation, 157–158, 164
  Apache Ant and, 337
  application status events, 253–254
  building PhoneGap applications, 13–14
  button events, 257–258, 261
  camera simulators, 170
  Capture API example on, 200–204
  Compass API example on, 209
  configuring PhoneGap Build for mobile platforms, 143–145
  contact information, 231
  debugging PhoneGap applications, 41–43
  device object running on, 245
  Eclipse plug-in and, 19
  errors related to contact information, 229–230
  geolocation support, 279
  JDK (Java Developer Kit) and, 333–334
  Media API support, 293
  media files, 295
  network status events, 256
  operating systems supported by PhoneGap, 3
  PhoneGap API support, 9
  PhoneGap Build support, 141
  picture capture process, 168–169, 173–175
  releasing `Media` objects, 298
  searching for contact information, 235
  testing applications created with PhoneGap Build, 152
  testing applications on physical devices, 36–37
  `watchHeading` function on, 213
Android development tools
  AVD (Android Virtual Device) for testing PhoneGap applications, 60–64
  configuring Eclipse development environment, 64–66
  creating PhoneGap project, 67–69
  creating PhoneGap project with Eclipse, 73–74
  installing SDK on Macintosh OSs, 60
  installing SDK on Windows OSs, 58–59
  making changes to Java source files, 70–72
  managing PhoneGap projects from command-line, 74–77
  options for creating PhoneGap projects, 66–67

Android development tools (*cont.*)
steps in installation of, 57–58
testing PhoneGap applications, 77–79
Android Virtual Device (AVD), testing PhoneGap applications with, 60–64, 78
Antenna Volt, types of hybrid applications, 21
Apache
Cordova Git repository. See Git repository
history of PhoneGap and, 5
Apache Ant
BlackBerry development environment and, 97
building PhoneGap applications, 76–77
installing on Macintosh OSs, 337
installing on Windows OSs, 338–339
APIs (application programming interfaces)
capturing settings from another application and adding to bada project, 93
consistency as cross-platform issue, 50–51
defining application version in bada, 88–90
PhoneGap APIs. See PhoneGap APIs
PhoneGap supported, 10
running web applications within PhoneGap container, 8
suite in PhoneGap, 3
Appcelerator Titanium, types of hybrid applications, 20
Apple
development environment. See iOS development environment
iOS. See iOS
iPhone. See iPhone
PhoneGap and, 11
registering as Apple developer, 113–114
Application container, designing for, 11–13
Application development
on Android. See Android development tools
on bada. See bada development environment
on BlackBerry. See BlackBerry development environment
on iOS. See iOS development environment
with PhoneGap Build. See PhoneGap Build
with Symbian. See Symbian development environment
Windows OSs. See Windows development environment
Application Manager, bada
creating application ID, 88
creating application profile, 86–88
defining application version, 88–90
defining platform version, 90–93
selecting target devices, 93–94
Application profile, creating for bada development project, 86–88
Application status events, 251–254
Applications, PhoneGap. See also Web applications
architecture of, 6–7
building, 13–16, 27
cross-platform issues, 49–53
debugging. See Debugging PhoneGap applications
Hello, World! example, 23–25
hybrid. See Hybrid applications
initialization, 25–28
leveraging PhoneGap APIs, 28–30
running on physical device, 36–37
running on simulators, 29–30, 33–34, 35–36

testing. See Testing PhoneGap applications
user interface enhancements, 30–35
Web 1.0 approach to building, 11
Web 2.0 approach to building, 11–12
Arrays, specifying contact properties, 225–226
AT&T WorkBench, 21
Audio
callback functions, 295–297
capture on Android devices, 202
capture with Capture API, 186, 198–199
creating `Media` objects, 294
determining current position while playing, 297
determining duration of playback, 297–298
example of use of Media API, 300–305
`mediaFileURI`, 294–295
playing clips, 298–299
recording, 299–300
AVD (Android Virtual Device), testing PhoneGap applications with, 60–64, 78

B

bada development environment
adding manifest file to PhoneGap project, 94–95
capturing API settings from another application, 93
configuring application security, 90, 92
creating application ID, 88
creating application profile, 86–87
creating PhoneGap project, 82–86
defining application version, 88–89
defining platform version, 90–92
defining unique name for application, 87–88
downloading/installing PhoneGap files, 80–82
overview of, 79–80
preparing PhoneGap for, 329
selecting target devices, 93–94
testing PhoneGap applications, 95–96
bada (Samsung), PhoneGap supported operating systems, 4
Beep, in Notification API, 310
BES (BlackBerry Enterprise Server), 109
BlackBerry
accelerator determining device orientation, 157–158
accelerator support and, 161
adding/saving contacts, 232–233
Apache Ant and, 337
application status events and, 253
build issues, 151
building PhoneGap applications, 14–15
button events, 257–258, 261
Capture API on, 196, 204
configuring camera options, 178, 180
configuring PhoneGap Build for mobile platforms, 143–145
debugging PhoneGap applications, 40–41
device object running on simulator, 245–246
E/S (emulator/simulator) and, 35–36
errors related to contact information, 229–230
`FileWriter` object and, 274
geolocation support, 279
getting current location of device, 284
HelloWorld application on, 29–30, 34
JDK (Java Developer Kit) for building applications, 333
Media API support, 293
mileage tracker example, 322
PhoneGap API documentation, 51

BlackBerry (*cont.*)
PhoneGap API support, 8–9
PhoneGap Build support, 141
PhoneGap supported operating systems, 4
picture capture process, 168, 170–172
reading directory entries, 269–272
running contacts example on, 231
searching for contact information, 234
signing keys, 99
storing contact information, 228
testing applications created with PhoneGap Build, 152
testing applications on physical device, 36–37
watching location of device, 286, 288
WebWorks. See WebWorks
BlackBerry development environment
build process, 104–107
building PhoneGap applications, 107–109
`config.xml` file, 100–103
creating PhoneGap project, 99–100
installing WebWorks SDK, 98–99
overview of, 97
testing PhoneGap applications on device, 111–112
testing PhoneGap applications on simulator, 109–111
BlackBerry® Development Fundamentals (Wargo), xxiv, 21, 40, 97, 105, 107
BlackBerry Enterprise Server (BES), 109
BlackBerry Mobile Data System (MDS)
overview of, 106–107
testing PhoneGap applications on, 109–111
BlackBerry WebWorks. See WebWorks
Build process. See also PhoneGap Build
accessing contact information and, 230
in BlackBerry development environment, 104–107
building applications for BlackBerry, 107–109
PhoneGap applications, 13–16
in PhoneGap Build, 148
`build.xml file`, 109
Button events
event listener for, 258–262
list of button types, 257
overriding button behavior, 257–258
overview of, 256–257
running on Android, 261

C

Callback functions
Capture API, 187–188
Contacts API, 236, 242
`DirectoryReader` object and, 267
File API, 270–271, 273–274, 277–278
Geolocation API, 280–281
how PhoneGap works and, 9
Media API, 295–297
Notification API, 308–309
SQL database, 321–322
Storage API, 319–320, 326
Camera API
accessing pictures on devices, 165–166
`allowEdit`, 180
Android example, 173–175
BlackBerry example, 171–172
Capture API compared with, 185
configuring camera options, 176
dealing with issues related to, 182–184
default options, 166–167
`destinationType`, 178–179
`encodingType`, 181

inconsistencies between device platforms, 168–170
iOS example, 169–170
iPhone example, 167–168
`mediaType`, 181–182
optic quality and, 177–178
overview of, 165
`sourceType`, 179–180
`targetHeight` and `targetWidth`, 181
Cameras, testing PhoneGap applications via, 152–153
Capture API
audio and video capture, 198–199
Camera API compared with, 185
configuring capture options, 189–191
image preview on iOS, 197–198
inconsistencies between device platforms, 195–196
Media API compared with, 293
overview of, 185
running on Android device, 200–204
running on BlackBerry device, 204
running on iPhone, 191–195
using, 186–189
Chrome (Google), 44–45
`clone` method, contacts and, 242
Cloud
building PhoneGap applications in, 141
packaging PhoneGap applications, 14
Command-line tools
development on BlackBerry and, 98
managing projects with, 74–77
testing applications, 77–79
Compass API
overview of, 205
querying device orientation, 205–206
running on iPhone, 206–208
`watchHeading` function, 210–213
`watchHeadingFilter` function, 213–215
watching device orientation, 209
Compression, JPEG format, 177
`config.xml` file
BlackBerry projects, 100–103
PhoneGap Build and, 16
PhoneGap Build projects, 145–146, 150
`confirm` method, in Notification API, 307–310
Connection object
example, 219–220
overview of, 217–219
running on Android device, 220
`console` object, JavaScript, 38–39
Contacts API
adding/saving contact on BlackBerry, 232–233
cloning contacts, 242
creating contacts, 224
example, 226–230
overview of, 223
removing contacts, 242
running on Android device, 235
running on BlackBerry device, 231
running on iOS device, 235
searching for contact information on BlackBerry, 234
searching for contact information on iPhone, 237
searching for contacts, 235–241
specifying contact properties, 224–226
Contacts API, W3C, 223
Copying files or directories, 276
Cordova Git repository. See Git repository
`createTable` function, SQL database, 319–320

Cross-platform applications
building native applications, 3
development issues, 49–53
CSS (Cascading Style Sheets)
building cross-platform native application, 3
running web applications within PhoneGap container, 7
Cygwin, building Symbian applications on Windows OS, 126–128

D

Dalvik Debug Monitor Server (DDMS), 42–43
`database` object, `transaction` method of, 319
DDMS (Dalvik Debug Monitor Server), 42–43
Debug mode, in PhoneGap Build, 153–154
Debugging camera problems, 183–184
Debugging PhoneGap applications
leveraging debugging capabilities, 37–43
overview of, 35
in PhoneGap Build, 153–154
RMEE (Ripple Mobile Environment Emulator) for debugging, 44–46
on Symbian, 134
third-party tools, 43–44
Weinre (Web Inspector Remote) for debugging, 46–49
on Windows Phone, 139–140
`destinationType` property, Camera API settings, 178–179
Developers
adding developer tools to Eclipse, 65–66
registering as Apple developer, 113–114
tools for, 55
Development environments
Android. See Android development tools
bada. See bada development environment
BlackBerry. See BlackBerry development environment
iOS. See iOS development environment
PhoneGap Build. See PhoneGap Build
Symbian. See Symbian development environment
Device APIs and Policy (DAP) Working Group, W3C (Worldwide Web Consortium), 10
Device location
canceling a watch, 289–291
getting current location, 280–284
setting a watch, 285–288
watching location of, 284
Device object
`device` properties, 244
overview of, 243
running on Android, 245
running on BlackBerry, 245–246
running on iPad, 246–248
running on iPhone, 246
Device orientation, in Accelerator API
overview of, 157–158
querying device orientation, 158–161
watching device orientation, 161–164
Device orientation, in Compass API
querying device orientation, 205–206
`watchHeading` function, 210–213
`watchHeadingFilter` function, 213–215
watching device orientation, 209
`device` properties, device object, 244
`deviceready` events, 250–251
Devices, physical. See Physical devices
Digital signing, configuring PhoneGap Build for mobile platforms, 143–145
Directories
accessing, 264

copying, 276
deleting, 275–276
errors accessing, 265
moving, 276–277
properties, 269–272
reading directory entries, 267–269
`DirectoryEntry` object
copying directories, 276
deleting directories, 275–276
moving directories, 276–277
properties, 269–272
`DirectoryReader` object, 267–269
Documentation, PhoneGap API, 17–18, 51
Dojo Mobile, 31
Downloads
bada SDK, 80–82
installing PhoneGap and, 327
JDK (Java Developer Kit), 333–334
Droid (Motorola), Capture API example on, 201
Drupal, PhoneGap plug-ins, 19
Duration, audio playback, 297–298
`duration` property, Capture API, 190

E

E/S (emulator/simulator)
camera simulators, 170
contacts example on BlackBerry simulator, 231
device object running on BlackBerry simulator, 245–246
device object running on iPad simulator, 246–248
launching PhoneGap project in iPhone simulator, 120
`onCameraError` on iOS simulator, 183
running PhoneGap applications, 35–36, 78
testing BlackBerry applications, 109–111
testing PhoneGap application in bada emulator, 95–96
testing PhoneGap application in iPhone simulator, 123
testing PhoneGap application with AVD, 60–64
testing PhoneGap Build applications, 152
Windows Phone Emulator, 136
Eclipse
configuring development environment for, 64–66
creating PhoneGap project with, 67–74
LogCat window, 41–42
Package Explorer, 70–71
PhoneGap plug-ins, 19
testing PhoneGap applications, 36–37, 77
Workbench, 65
Emulator Web Application, testing application in bada emulator, 95–96
`encodingType` property, Camera API settings, 181
Enterprises, iOS development and, 114
Errors
build issues, 150–151
camera problems, 182–183
Capture API, 188–189
Compass API, 206
Contacts API, 228–230
database transactions, 319–321
directory access, 265
file and directory access, 265–266
geolocation, 281
Media API, 295–296
Event listeners
for application status events, 251–253
creating, 249–250
for `deviceready` events, 250–251
Event Log application, BlackBerry, 40–41

Events API
application status events, 251–254
button events, 256–262
creating event listeners, 249–250
deviceready events, 250–251
network status events, 254–256
types of events supported by PhoneGap, 249
ExternalHosts, configuring in Xcode, 305

F

Facebook, PhoneGap plug-ins, 18
File API
accessing file system, 264–267
copying files or directories, 276
deleting files or directories, 275–276
FileEntry and DirectoryEntry properties, 269–272
moving files or directories, 276–277
overview of, 263
reading content from files, 274–275
reading directory entries, 267–269
storage types, 263–264
uploading files to servers, 277–278
writing data to files, 272–274
File API:Directories and System specification, W3C (Worldwide Web Consortium), 263
File system, accessing, 264–267
FileEntry object
copying files, 276
deleting files, 275
moving files, 276–277
properties, 269–272
FileReader object, 274–275
FileTransfer object, 277–278
FileURI, for Media object, 294–295
FileWriter object, 272–274
Folders
installing PhoneGap and, 328
location for iOS project, 118
4G networks, connection object and, 218

G

Geolocation API
canceling a watch, 289–291
getting current location of device, 280–284
overview of, 279–280
setting a watch, 285–288
watching device location, 284
Geolocation API specification, W3C (Worldwide Web Consortium), 279
Git repository
delivering application files to build server, 147
downloading/installing files for bada development project, 80–81
Google Android. See Android
Google Chrome, 44–45
Google Groups, 19
GPS capabilities. See Geolocation API
Graphics. See Images

H

HP/Palm webOS. See webOS
HTML (Hypertext Markup Language)
building cross-platform native applications, 3
HTML5 approach to building PhoneGap applications, 11–13
HTML5 support for geolocation, 279
HTML5 support for storage, 315
running web applications within PhoneGap container, 7

Web 1.0 (traditional) approach to building applications, 11
Web 2.0 approach to building applications, 11–12
Hybrid applications
defined, 3
frameworks of, 19–20
Hypertext Markup Language. See HTML (Hypertext Markup Language)

I

IBM, in history of PhoneGap, 4–5
IBM Worklight, 22
Icons
creating iOS projects, 119
creating PhoneGap Build projects, 145–146
as cross-platform issue, 53
IDEs (integrated development environments)
bada as, 82–86
Eclipse as, 64
Image capture. See Camera API; Capture API
Images
accessing on mobile devices, 165–166
accessing pictures on devices, 165–166
displaying image file URI, 169–170
`mediaType` property, 182
rotating graphics with jQuery Rotate, 212
`index.html`
creating PhoneGap project with Eclipse, 73–74
delivering application files to build server, 146
Infuse 4G device (Samsung), 204
Initialization, of PhoneGap applications, 25–28
`INSERT` statement, SQL database, 323–324
Installing PhoneGap. See PhoneGap installation
Integrated development environments (IDEs)
bada as, 82–86
Eclipse as, 64
iOS
accessing media files, 295
application status events, 251, 253
building PhoneGap applications, 15
button events, 256–257
camera simulators, 170
Capture API example on, 197
configuring camera options, 180
configuring PhoneGap Build for mobile platforms, 143–145
device object running on iPad simulator, 246–248
displaying image file URI, 169–170
Hello, World! application on, 26
image preview on, 197–198
Media API support, 293
`onCameraError` in iOS simulator, 183
PhoneGap API documentation, 51
PhoneGap API support, 9
PhoneGap Build support, 141
PhoneGap plug-in for Drupal, 19
PhoneGap supported operating systems, 3
picture capture process, 168–169
searching for contact information, 235
testing applications on physical device, 36
uploading files to server and, 278
iOS development environment
accessing web content for project, 119–122
creating PhoneGap project, 116–117
folder location for projects, 118
installing Xcode, 114–116

iOS development environment (*cont.*)
naming projects and defining project locations, 117–118
overview of, 113
preparing PhoneGap for, 329–330
registering as Apple developer, 113–114
testing PhoneGap applications, 122–123
versioning, 118–119
iPad
device object and, 246–248
PhoneGap support, 3–4
iPhone
accelerator support and, 159
Camera API example, 167–168, 191–195
configuring camera options, 180
device object, 246
inconsistent implementation of PhoneGap APIs, 30
launching PhoneGap project in, 120
PhoneGap support, 3
running HelloWorld application on iPhone simulator, 26
searching for contact information, 237
testing PhoneGap application in, 123
iPhoneDevCamp, 4

J

Java API, RIM (Research In Motion), 246
Java Developer Kit. See JDK (Java Developer Kit)
Java, making changes to source file using Eclipse, 70–71
JavaScript
`alert` method, 307
bada source code files, 329
build cross-platform native applications, 3
building PhoneGap applications, 13–14
`console` object, 38–39
cross-platform issue, 51
loading JavaScript library, 27
running web applications within PhoneGap container, 7–8
Web 2.0 approach to building applications, 11–12
WebWorks providing JavaScript methods, 246
JDK (Java Developer Kit)
Android development and, 57
bada and, 80
BlackBerry development and, 97
configuring Windows Path environment, 335–336
confirming installation of, 336
downloading, 333–334
installing, 334
JRE (Java Runtime Environment) included in, 337
JPEG format
compression, 177
images, 181
`mode` property of Capture API, 190
jQuery
`$()` function, 212
reasons for using, 268
rotating graphics with, 212
jQuery Mobile (jQM)
as application interface, 192, 200
creating interface for directory reader, 268
creating interface for media application, 300–301
creating interface for notification application, 313
searching for contact information, 237
use in application development, 30–35
JRE (Java Runtime Environment), 333–334, 337

K

Key/value pairs, local storage and, 316

L

Launch screens, creating iOS PhoneGap project, 119
LG Thrill device
  device object on, 245
  video capture on, 202–203
`limit` property, Capture API, 190
Linux OSs
  building Symbian PhoneGap applications on, 125
  configuring Eclipse development environment, 64
  launching Unix applications from command line, 75
  options for PhoneGap development on Android, 57
Local storage, Storage API, 316–317
LogCat window, Eclipse, 41–43

M

Macintosh OSs
  bada development tools and, 79
  building Symbian PhoneGap applications on, 125–127
  configuring Eclipse development environment, 64
  development environment. See iOS development environment
  installing Android SDK on, 60
  installing Apache Ant on, 337
  Installing BlackBerry WebWorks SDK, 98–99
  JDK (Java Developer Kit) and, 333
  launching Unix applications from command line, 75
  options for PhoneGap development on Android, 57
  packaging Symbian PhoneGap projects, 131
  testing Symbian PhoneGap projects, 132
  Windows Phone development and, 135
Magnetic poles, device orientation and, 206
Make utility
  installing, 126–127
  packaging PhoneGap projects, 131–132
Makefiles, 126, 131–132
Manifest file, adding to PhoneGap project in bada, 94–96
Media API
  callback functions, 295–297
  creating `Media` objects, 294
  determining current position while playing media files, 297
  determining duration of playback, 297–298
  example of use of, 300–305
  `FileURI`, 294–295
  overview of, 293
  playing audio clips, 298–299
  recording audio, 299–300
  releasing `Media` objects, 298
Media Capture API, W3C (Worldwide Web Consortium), 185
Media files, using Capture API, 186–187
`Media` objects
  creating, 294
  releasing, 298
`mediaType` property, Camera API settings, 181–182
Memory cards, 61
Microsoft Windows. See Windows OSs
Mobile browsers. See also Web browsers, 279

Mobile Data System (MDS)
BlackBerry and, 106–107
testing PhoneGap BlackBerry applications on, 109–111
`mode` property, Capture API, 190–191
Motorola Droid, Capture API example on, 201
Moving files or directories, 276–277

N

Names
defining unique name for application, 87–88
iOS PhoneGap project, 117
PhoneGap Build project, 145
Navigation, as cross-platform issue, 52
Navigator object, instantiating APIs from, 28
Network status events, 254–256
Networks, connection object and, 217–218
New project dialog, PhoneGap Build, 147–148
Nitobi
history of PhoneGap and, 4–5
support offered by, 19
Nokia
Symbian. See Symbian
Web Tools, 125–126
Notification API
beep, 310
example application of, 310–313
overview of, 307
vibrate, 310
visual alerts, 307–310

O

`offline` events, network status events, 254
`online` events, network status events, 254
`onStatus` function, media playback and, 296–297
Open source frameworks
PhoneGap as, 3
support and, 19
Optic quality, cameras and, 177
Oracle JDK. See JDK (Java Developer Kit)
`organizations array`, specifying contact properties, 226
OSs (operating systems)
application requirements as cross-platform issue, 52
bada. See bada (Samsung)
configuring Eclipse development environment, 64
emulator/simulators and, 35–36
installing Apache Ant on, 337–339
JDK (Java Developer Kit) and, 334
Linux OSs. See Linux OSs
Macintosh OSs. See Macintosh OSs
PhoneGap supported, 3–4
Windows OSs. See Windows OSs
OTA (over the air)
deploying applications to BlackBerry smartphones, 108–109
testing PhoneGap applications, 152

P

Packaging PhoneGap projects
cloud-based service, 14
with Symbian, 131–132
Palm webOS (HP). See webOS
`pause` events, application status events, 251–254
`pause` method, `Media` objects, 298–299
Persistent storage, file storage options, 263–264
PhoneGap APIs
accelerometer. See Accelerator API

camera. See Camera API
capture. See Capture API
capturing API settings from another application and adding to bada project, 93
compass. See Compass API
connection. See Connection object
contacts. See Contacts API
defining application version in bada, 88–90
devices. See Device object
events. See Events API
files. See File API
geolocation capabilities. See Geolocation API
leveraging, 28–30
media. See Media API
notifications. See Notification API
responding to JavaScript calls, 27–28
storage. See Storage API
supporting multiple mobile platforms, 8–9
PhoneGap Build
build process, 148
building applications with, 27
cloud-based packaging service, 14
configuring, 143–145
configuring projects, 148–150
`config.xml` file, 16
creating accounts, 142–143
creating application for, 145–146
creating projects, 146
dealing with build issues, 150–151
debugging applications, 153–154
delivering application files to build server, 146–147
development environments compared with, 142
need for, 328
new project dialog, 147–148
overview of, 141
testing applications, 152–153
PhoneGap installation
overview of, 327–328
preparing for bada development, 329
preparing for iOS development, 329–330
preparing for Windows Phone development, 330–331
PhoneGap, introduction to
building applications, 13–16
designing for application container, 11–13
history of, 4–5
how it works, 6–10
hybrid application frameworks, 19–22
limitations of, 17–18
overview of, 3–4
plug-ins, 18–19
reasons for using, 5–6
support options and resources, 19
Photos. See Images; Pictures
Physical devices
testing accelerator on, 158
testing applications created with PhoneGap Build, 152
testing BlackBerry applications on, 111–112
testing Eclipse applications on, 78–79
testing PhoneGap applications on, 36–37
Pictures. See also Images
accessing on devices, 165–166
`mediaType` property, 182
`play` method, `Media` object, 298–299
Playback, of media files
determining current position, 297
determining duration of, 297–298
playing audio clips, 298–299

Plug-ins
Eclipse, 64
jQuery Rotate, 212
for use with PhoneGap, 18–19
PNG format, 181, 190
Properties
connection object, 217–218
contact, 224–226
device object, 244–245
`FileEntry` and `DirectoryEntry`, 269–272
geolocation, 280–281

Q

`quality` property, Camera API settings, 177–178
Queries, SQL databases, 324–325

R

Raw images, 178–179
Reading
content from files, 274–275
directory entries, 267–272
Recording audio, 299–300
`remove` method, contacts, 242
Research In Motion. See RIM (Research In Motion)
`resume` events, application status, 251–254
RIM (Research In Motion)
BlackBerry. See BlackBerry
emulator/simulators and, 35–36
Java API, 246
PhoneGap supported operating systems, 4
RMEE (Ripple Mobile Environment Emulator), 44–46
RMEE (Ripple Mobile Environment Emulator), 44–46

S

Samsung
bada development environment. See bada development environment
Infuse 4G device, 204
PhoneGap supported operating systems, 4
SDKs (software development kits)
downloading/installing bada SDK, 80–82
installing Android SDK on Macintosh OSs, 60
installing Android SDK on Windows OSs, 58–59
installing BlackBerry WebWorks SDK, 98–99
Nokia, 125–126
PhoneGap Build compared with, 142
testing PhoneGap applications, 78–79
Windows Phone 7.1, 135–136
Searches, for contacts, 235–241
Security
configuring in bada development environment, 90, 92
PhoneGap Build projects and, 145
`seekTo` method, `Media` objects, 298–299
Sencha Touch, use in application development, 31
Servers
BES (BlackBerry Enterprise Server), 109
DDMS (Dalvik Debug Monitor Server), 42–43
delivering application files to build server, 146–147
uploading files to, 277–278
Session storage, local storage, 316
Signing keys
BlackBerry applications, 99
configuring PhoneGap Build for mobile platforms, 143–145

Simulators. See E/S (emulator/simulator)
Smartphones
application status events, 251
button events, 256
Capture API example on, 201
connection object, 217
current location of, 280
deploying applications to, 108–109
device object example, 245
emulator/simulators and, 35–36
file storage options, 263–264
geolocation capabilities, 279
how PhoneGap works, 6
memory cards, 61
mimicking native applications, 31
Ripple emulator and, 46
running HelloWorld application on, 24–25
specifying contact properties and, 226
SQLLite and, 315, 326
testing PhoneGap applications, 152–153
Web 1.0 and Web 2.0 technologies, 12
`sourceType` property, Camera API settings, 179–180
Splash screens, PhoneGap Build projects, 145–146
SQL databases
creating transactions, 317–318
example of mileage tracker application, 322
executing SQL statements, 320–324
opening, 317
passing functions to transactions, 319–320
querying SQL statements, 324–325
SQL (Structured Query Language), 317
SQLLite database engine, 315, 326
`stop` method, `Media` object, 298–300
Storage API
creating database transactions, 317–318
executing SQL statements, 320–324
local storage, 316–317
mileage tracker example on BlackBerry Torch 9800, 322
opening SQL database, 317
overview of, 315
passing functions to transactions, 319–320
querying SQL statements, 324–325
SQLLite and, 326
Storage types, files, 263–264
Strings, specifying contact properties, 224–225
Strobe, types of hybrid applications, 22
Structured Query Language (SQL), 317
Symbian
building PhoneGap applications, 15
cross-platform issues, 51
PhoneGap Build support, 141
supported operating systems, 4
Symbian development environment
configuring application settings, 129–130
creating PhoneGap project, 128–129
installing Make utility, 126–127
installing Nokia Web Tools, 125–126
modifying HelloWorld application for, 130–131
overview of, 125
packaging PhoneGap projects, 131–132
testing applications created with PhoneGap Build, 152
testing PhoneGap applications, 132–134

T

Tables, SQL database, 319–320
Tablets, support for WebWorks tablet applications, 103

targetHeight/targetWidth properties, Camera API settings, 181
Temporary storage
accessing temporary sandbox storage, 264–265
file storage options, 263–264
Testing PhoneGap applications
on Android emulator, 78
on AVD (Android Virtual Device), 60–64
in bada development environment, 95–96
in BlackBerry development environment, 109–112
in iOS development environment, 122–123
overview of, 35
in PhoneGap Build, 152–153
on physical device, 36–37, 78–79
on simulator, 35–36
in Symbian development environment, 132–134
in Windows development environment, 139–140
Tiggr, types of hybrid applications, 22
Titanium Appcelerator, 20
Torch simulators. See also BlackBerry
contacts example running on, 231
getting current location of device, 284
HelloWorld application running on, 29–30
mileage tracker example, 322
reading directory entries, 269–272
watching location of device, 286, 288
transaction method, of database object, 319
type property, connection object, 217–218

U

UIs (user interfaces). See also jQuery Mobile (jQM)
cross-platform issues related to, 52
enhancements, 30–35
Universities, iOS development and, 114
Uploading files
to build server, 146–147
to servers, 277–278
URI
Camera API and, 165
camera destinationType properties, 177–178
camera quality properties, 177–178
capture process and, 173–175
configuring camera options, 176
FileURI for media objects, 294–295
iOS example displaying image file URI, 169–170
USB
running PhoneGap applications on physical device, 78
testing PhoneGap BlackBerry applications, 111
User interfaces (UIs). See also jQuery Mobile (jQM)
cross-platform issues related to, 52
enhancements, 30–35

V

Versions, in bada development project
defining application version, 88–89
defining platform version, 90–92
Versions, in iOS development project, 118–119
Vibrate, in Notification API, 310

Video capture
on Android devices, 202–204
with Capture API, 186, 198–199
Video, `mediaType` property, 182
Virtual machines (VMs)
developing applications for Windows Phone, 135
running Windows OS on Macintosh, 79
Visual alerts, in Notification API, 307–310
Visual Studio 2010 Express
creating Windows Phone project, 136–139, 330
testing Windows Phone project, 139–140
VMs (virtual machines)
developing applications for Windows Phone, 135
running Windows OS on Macintosh, 79
Voice recorders, audio capture on Android devices, 202–203

W

W3C (Worldwide Web Consortium)
Contacts API, 223
Device APIs and Policy (DAP) Working Group, 10
File API:Directories and System specification, 263
Geolocation API specification, 279
Media API and, 293
Media Capture API, 185
Web SQL Database Specification, 315
Web Storage API Specification, 315
Widget Packaging and XML Configuration specification, 145
widget specification, 125
`watchHeading` function, Compass API, 210–213
`watchHeadingFilter` function, Compass API, 213–215
`watchID` variable, geolocation API
canceling a watch, 289–291
overview of, 284
setting a watch, 285–288
Web 1.0, 11
Web 2.0, 11–12
Web App Simulator
Nokia Web Tools for Windows OSs, 126
packaging Symbian PhoneGap projects, 126
testing Symbian PhoneGap projects, 132–134
Web applications
building into PhoneGap application, 25
building PhoneGap applications, 13–16
PhoneGap application types and, 37
running within PhoneGap container, 7–8
Web browsers
geolocation support, 279
running web applications within PhoneGap container, 7–8
storage and, 315, 317
Web content
accessing for iOS PhoneGap project, 119–122
adding to Windows Phone PhoneGap project, 138
creating PhoneGap project using PhoneGap Build, 145–146
folder structure as cross-platform issue, 51
Web Inspector, debugging Symbian PhoneGap projects, 134
Web Inspector Remote (Weinre)
debugging applications created with PhoneGap Build, 154
debugging PhoneGap applications, 46–49

Web Runtime (WRT) widgets
configuring application settings for PhoneGap project on Symbian, 129–130
running PhoneGap applications on Symbian as, 125–126
Web sites, PhoneGap resources, 19
Web SQL Database Specification, W3C (Worldwide Web Consortium), 315
Web Storage API Specification, W3C (Worldwide Web Consortium), 315–316
Web Tools, Nokia, 125–126
Web views, rendering, 7
webOS
building PhoneGap applications, 14–15
PhoneGap Build support, 141
PhoneGap supported operating systems, 3
testing applications created with PhoneGap Build, 152
WebSDKSimulator
Nokia Web Tools for Mac OSs, 126
testing Symbian PhoneGap projects, 132
WebWorks. See also BlackBerry
accessing contact information and, 230
build process, 104–107
creating PhoneGap project, 99–100
debugging web content, 37
installing SDK, 98–99
JavaScript methods provided by, 246
overview of, 21
Weinre (Web Inspector Remote)
debugging applications created with PhoneGap Build, 154
debugging PhoneGap applications, 46–49
`.wgz` files, packaging PhoneGap projects, 131
Wi-Fi networks, connection object and, 217–218
Widget Packaging and XML Configuration specification, W3C (Worldwide Web Consortium), 145
Widgets
configuring application settings for PhoneGap project on Symbian, 129–130
running PhoneGap applications on Symbian as, 125
Wikis, PhoneGap resources, 19
Windows development environment
creating PhoneGap project, 136–139
installing Windows Phone development tools, 135–136
overview of, 135
testing PhoneGap applications, 139–140
Windows OSs
bada development tools and, 79
building Symbian PhoneGap applications on, 125–126
configuring Eclipse development environment, 64–65
installing Android SDK on, 58–59
installing Apache Ant on, 338–339
Installing BlackBerry WebWorks SDK, 98–99
options for PhoneGap development on Android, 57
packaging Symbian PhoneGap projects, 131
testing Symbian PhoneGap projects, 132–134
Windows Path environment
configuring for use with JDK, 335–336
installing Apache Ant on Windows OSs, 338–339
Windows Phone
building PhoneGap applications, 15
creating PhoneGap project for, 136–139

installing development tools, 135–136
PhoneGap supported operating systems, 3
preparing PhoneGap for, 330–331
support for, 135
Windows Phone Emulator, 136, 140
Workbench. See AT&T WorkBench
Worklight, types of hybrid applications, 22
Worldwide Web Consortium. See W3C (Worldwide Web Consortium)
Writing data, to files, 272–274
WRT (Web Runtime) widgets
configuring application settings for PhoneGap project on Symbian, 129–130
running PhoneGap applications on Symbian as, 125–126

X

X coordinates. See Device orientation, in Accelerator API
Xcode
accessing web content for iOS project, 119–122
configuring ExternalHosts, 305
creating iOS project, 116–122
installing, 114–116
naming projects and defining project locations, 117–118
new project window, 117
preparing PhoneGap for iOS development, 329
welcome screen, 116
XHTML, 11

Y

Y coordinates. See Device orientation, in Accelerator API

Z

Z coordinates. See Device orientation, in Accelerator API
Zip archives
options for delivering application files to build server, 146
packaging PhoneGap projects, 131
PhoneGap files distributed via, 328