

Adobe

Dreamweaver CS5.5

Designing and Developing for Mobile with jQuery, HTML5 and CSS3

STUDIO TECHNIQUES

David Powers



Adobe

Dreamweaver CS5.5

Designing and Developing for Mobile with jQuery, HTML5 and CSS3

STUDIO TECHNIQUES

David Powers



Adobe Dreamweaver CS5.5 Studio Techniques: Designing and Developing for Mobile with jQuery, HTML5, and CSS3

David Powers

This Adobe Press book is published by Peachpit.
For information on Adobe Press books, contact:

Peachpit

1249 Eighth Street
Berkeley, CA 94710
510/524-2178
510/524-2221 (fax)

For the latest on Adobe Press books, go to www.adobepress.com
To report errors, please send a note to errata@peachpit.com
Peachpit is a division of Pearson Education.

Copyright © 2011 by David Powers

Associate Editor: Valerie Witte
Production Editor: Cory Borman
Developmental Editor: Anne Marie Walker
Copyeditor: Anne Marie Walker
Proofreader: Patricia Pane
Composition: WolfsonDesign
Indexer: Joy Dean Lee
Cover Image: Alicia Buelow
Cover Design: Charlene Charles-Will

Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.
For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Dreamweaver and Photoshop are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

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 Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN-13: 978-0-321-77325-8
ISBN-10: 0-321-77325-X

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

Contents

	About the Author	iv
	Acknowledgments	v
	Introduction	vi
Section I	Dreamweaver CS5.5	1
<i>Chapter 1</i>	Dreamweaver Goes Mobile	3
	Assessing HTML5 and CSS3	6
	Using HTML5 and CSS3 with Dreamweaver CS5.5	14
	Developing for Multiple Devices	27
Section II	HTML5 and CSS3	69
<i>Chapter 2</i>	Progressive Enhancement with HTML5 and CSS3	29
	Improving an Existing Site	31
	Sacrificing a Uniform Look	68
<i>Chapter 3</i>	Adapting Pages for Mobile with Media Queries	7
	Understanding Media Queries	73
	Adapting the Tozai Hotel Site	82
	Assessing Media Queries	115
<i>Chapter 4</i>	Making Your Site Available Offline	117
	How Offline Sites Work	118
	Making the Tozai Hotel Site Available Offline	124
	Going Offline	138
Section III	jQuery Mobile and PhoneGap	139
<i>Chapter 5</i>	Introducing jQuery Mobile	141
	Creating a Basic Site with jQuery Mobile	143
	Building on a Solid Foundation	173
<i>Chapter 6</i>	Diving Deeper into jQuery Mobile	175
	A Guide to jQuery Mobile Custom Data Attributes	177
	Rapid Deployment with jQuery Mobile Widgets	188
	Case Study: Creating a Reservation Form	207
	Submitting a Form and Displaying the Response	216
	Getting Your Hands Dirty with Code	218
<i>Chapter 7</i>	Building a Native App with PhoneGap	219
	Setting Up PhoneGap in Dreamweaver	221
	Case Study: A Travel Notes App	230
	Going Further	270
	Index	271

Bonus material mentioned in this eBook is available after the index.

About the Author



David Powers started developing websites in 1994 while at the BBC (British Broadcasting Corporation). He'd just taken on the role of Editor, BBC Japanese TV, and needed a way of advertising the fledgling channel in Japan. The problem was that he had no advertising budget. So, he begged the IT department for a corner of server space and singlehandedly developed an 80-page bilingual website, which he regularly maintained for the next five years—on top of all his other duties.

After three decades as a radio and TV journalist, David left the BBC in 1999 to work independently. He created multilingual websites for several leading clients, including the Embassy of Japan in London and Oxford Analytica. In 2003, he decided to combine his professional writing and editing expertise with his passion for the web, and began writing books on web development. This is his fourteenth so far. Readers frequently comment on David's ability to explain complex technical subjects in a jargon-free style that's easy to understand. At the same time, he doesn't talk down to readers, thereby appealing equally to more experienced web developers.

David is an Adobe Community Professional and Adobe Certified Instructor for Dreamweaver. You'll often find him giving help and advice in the Dreamweaver forums and Adobe Developer Center—to which he has contributed many popular tutorials and training videos. He greatly enjoys traveling and taking photos—all the photos used in this book were taken by him.

David has also translated a number of musical plays from Japanese into English, and he likes nothing better than sushi with a glass or two of cold sake.

Acknowledgments

Writing a book about new software is a solitary activity, grappling with a constantly moving target and pounding the keyboard to deliver the chapters on time. But none of it would be possible without an army of helpers. First, there's Scott Fegette, Senior Product Manager for Dreamweaver, who kept me informed of the engineering team's plans. Then there's Kin Blas, a Dreamweaver engineer actively involved in developing jQuery Mobile, who clarified points I found difficult to understand. My thanks go to them and to the rest of the Dreamweaver team for their help both directly and indirectly.

I've also had a strong backup team at Peachpit: Victor Gavenda, who accepted the concept of this book and liked it so much that he persuaded Adobe Press that it was high time one of my books was printed in color; Valerie Witte, my editor, who calmly accepted my frequent changes of mind about the structure of the book; Anne Marie Walker, my development editor, who picked up inconsistencies and helped me (mis)spell the American way; Tom Muck, my technical editor, who spotted problems with code and made suggestions to improve it; and Cory Borman, who oversaw the production process.

Many others have helped indirectly. At times, the Twitter stream felt like an annoying distraction, but it provided some invaluable leads, alerting me to changes in this fast-moving industry. It also provided some essential light relief, although I'm not sure I'm ready to watch another cat video just yet.

Introduction

Don't be fooled. Although the .5 might give the impression that Dreamweaver CS5.5 is a point release, it's anything but. Dreamweaver engineers have packed a stunning amount of new features into this version. To mention just a few, there's code hinting for the popular jQuery JavaScript library, the ability to see what pages will look like at different screen resolutions without leaving the Document window, support for jQuery Mobile widgets, and integration of PhoneGap to build native apps for Android or iOS (the operating system used in the iPhone, iPad, and iPod touch).

The emphasis in Dreamweaver CS5.5 is firmly on mobile development and designing for multiple screens, but that's not all. There's improved support for HTML5 and CSS3, including tools to simplify the creation of rounded corners and drop shadows without images. Previous versions of Dreamweaver supported only a limited range of CSS selectors. Live view now supports them all. Oh yes, Dreamweaver CS5.5 supports web fonts, too.

There's a lot to absorb, and this book aims to guide you through all the new features with the help of three case studies. The first one centers on redesigning a website for display on desktops, tablets, and smartphones using HTML5, CSS3, and media queries. The second takes a cut-down version of the same site and builds a dedicated mobile version using jQuery Mobile, a sophisticated JavaScript and CSS framework designed to work consistently on all major mobile platforms. The final case study develops a simple app that stores information in a database, accesses a mobile phone's GPS sensor, and displays a map.

Is This the Right Book for You?

The new features in Dreamweaver CS5.5 are aimed at web designers and developers who are already comfortable with HTML and CSS. It also helps to have at least a basic understanding of JavaScript and some jQuery experience. If the

thought of diving into code sends shivers up your spine, this might not be the most appropriate book for you. Web development is becoming increasingly sophisticated, and the days of just copying and pasting snippets of code are rapidly drawing to a close.

Having said that, you don't need to be an expert. I firmly believe that if you understand why you're being told to do something a particular way, you're more likely to remember and be able to adapt it for your own projects. Each step is explained, as are new concepts, but I don't go back to basics, such as describing what a function or event handler is.

Mac or Windows?

The differences between the Mac and Windows versions of Dreamweaver are so few as to be negligible. In the rare cases where there is a difference, I point it out and show a screen shot if necessary. The most important difference, as far as this book is concerned, lies in PhoneGap integration. Both Windows and Mac support Android, but the software necessary to build apps for iOS runs only on a Mac. The other difference, as always, lies in keyboard shortcuts. I provide both versions, Windows first, followed by Mac.

Using a multibutton mouse is now so common among Mac users that I refer only to right-click instead of giving Control-click as the alternative. On most Macs, the F keys now control hardware features, such as sound level and brightness. When I refer to F keys, you need to hold down the Fn key at the same time. Alternatively, open Keyboard in System Preferences and select the "Use all F1, F2, etc. keys as standard function key" check box.

Although I test on both operating systems, I had to choose one for taking screen shots. Most of them have been taken on Windows 7, but some have been taken on Mac OS X 10.6 where appropriate. However, this is a book about mobile development. So, many screen shots have also been taken on Android (HTC Desire and Samsung Galaxy Tab) and iOS (iPad and iPod touch). I also tested on a BlackBerry Torch and Windows Phone 7.

Downloading the Case Study Files

This book doesn't come with a CD. However, you can download the files used in the case studies from my website at <http://www.dreamweaver.com>. In most cases, all the necessary files are supplied. However, for licensing reasons, you need to obtain the Calluna Regular web font directly (the details are in Chapter 2). Also, the download files don't include the jQuery Mobile or PhoneGap libraries. Dreamweaver copies them directly to your site when you create a jQuery Mobile page (see Chapter 5) or define the Native Application Settings (see Chapter 7).

Keeping Up to Date

The jQuery Mobile framework was feature complete at the time Adobe locked down the code for the release of Dreamweaver CS5.5. However, work continued on stabilizing and optimizing performance. Consequently, newer versions of the jQuery Mobile style sheet, external JavaScript files, and images are likely to be available by the time you read this. Adobe plans to release extensions to update the files in Dreamweaver. Chapter 5 also describes how to change the source folder for the files so that you can use your own customized versions.

Because jQuery Mobile is a new framework, it's likely to continue to develop. I'll try to keep abreast of its progress and will post updates that affect this book on my website at <http://www.dreamweaver.com>.

Adobe is a jQuery Mobile project sponsor, and Dreamweaver engineers are playing an active role in its development. That holds the promise of even greater things to come.

CHAPTER

4

Making Your Site Available Offline



*You can't always get what you want,
But if you try sometimes,
You might get what you need.*

—The Rolling Stones

Making Your Site Available Offline

Loss of signal is probably one of the most frustrating aspects of surfing the web with a mobile device. You've just clicked a link and the page is beginning to load when your train enters a tunnel. Your connection disappears. Even when the train emerges from the tunnel, your mobile has to hunt for a signal and you often need to start all over again.

HTML5 can't improve mobile connectivity, but it does make it possible to continue interacting with websites, even when no network connection is available. The secret lies in caching the necessary files. Although browsers automatically cache recently downloaded files, what's different about HTML5 is that you can instruct the browser to download files in advance of their being needed. You can also specify alternative files to be displayed if the user is offline.

In this chapter, you'll learn how to make a site available offline by creating a file that not only tells the browser which files to cache, but also specifies substitute files for offline use. To speed up this process, the download files for this chapter contain a Dreamweaver extension that I created to generate a list of all files used in a site or folder.

How Offline Sites Work

To make a site available without a network connection—an *offline web application*, as the HTML5 specification calls it—you need to create a *manifest*. This is a list of files that the browser needs to download and store in an application cache. The first time someone visits your site, the browser

checks the manifest and downloads the listed files ready for use offline. The next time the same user visits your site, the browser checks the manifest. If it detects a change, all the files are downloaded again, updating the application cache.

Figure 4.1 shows which browsers support offline applications as reported by caniuse.com. Light green shows full support; darker green shows partial support; and pink indicates no support. Internet Explorer (IE) is the only mainstream browser with no support. Crucially, though, iOS Safari, Android, and Opera Mobile all support offline access, making it ideal for websites that you expect to be accessed on mobile devices.



Figure 4.1 Most modern browsers apart from IE support offline access.

Creating a Manifest

The manifest is a plain text file that must be saved with a `.manifest` filename extension. It's not important where you locate the manifest, but the most logical place is in the site root. However, if you want to make only part of a site available offline, the manifest should be located in the relevant folder and cover the files in all subfolders. The first line inside the manifest file should look like this:

```
CACHE MANIFEST
```

There should be only a single space between `CACHE` and `MANIFEST`, both of which should be in uppercase.

Following this is a list of files grouped according to how you want them to be treated when the user is offline:

- ▶ **Explicit section.** All files in this section are downloaded automatically, even if they're not required for the current page.
- ▶ **Online whitelist section.** Files in this section are never cached. The browser always tries to access the online version.
- ▶ **Fallback section.** This is where you specify substitute files that the browser should use when the user is offline.

The following basic rules apply to all sections:

- ▶ Each file must be listed on a separate line, except in the fallback section where the original and substitute files are listed on the same line with a space between them.
- ▶ Document-relative paths should be relative to the manifest.
- ▶ Paths relative to the site root (in other words, those that begin with a leading slash) or fully qualified URLs are also acceptable.
- ▶ The list should include not only web pages, but other assets, such as images, style sheets, and JavaScript files.
- ▶ Blank lines are permitted.
- ▶ Comments can be included, but they must be on a separate line beginning with a hash or pound sign (#) optionally preceded by spaces or tab characters.

Sections can be listed in any order and don't need to be a single block. For example, you might want to make some files available offline only for a limited period. So, it makes sense to list them separately from the core files that don't normally change.

You create sections by placing a section header on a separate line.

Specifying files that should be cached

The explicit section is the default, so files listed immediately after `CACHE MANIFEST` are automatically downloaded and cached. To switch back to the explicit section after the



Section headers must be written in uppercase and are followed by a colon. Headers can be preceded by spaces, but there should be nothing else on the same line.

online whitelist or fallback section, place the following section header on a separate line:

CACHE:

Specifying files that must always be accessed online

Server-side scripts and other files that you don't want to be cached locally should be listed in the online whitelist section. You create this by adding the following header on a separate line:

NETWORK:

Then list the path or URL of each file on a separate line in the same way as for files that you want to be downloaded.

If your site accesses resources on other domains or subdomains, you should add an asterisk (*) on a line of its own in the online whitelist section like this:

NETWORK:

*

This indicates that access to resources on other domains is not blocked.

Specifying alternative files to use offline

To specify alternatives for files that can't be accessed offline, create a fallback section by placing the following section header on a separate line:

FALLBACK:

Each entry in the fallback section lists a file in the online site followed by the location of a substitute file to be used when offline. Both files are listed on the same line and separated by one or more spaces.

To represent any file, use a single forward slash (/) as the first part of the entry. For example:

FALLBACK:

/ offline.html

This substitutes offline.html for any file not listed elsewhere.



Technically speaking, you can use the CACHE: section header immediately after CACHE MANIFEST, but it's unnecessary.



Browser Caches

Application caches are designed to make the website—or parts of it—available offline. They're separate from the normal browser cache, which speeds up the rendering of pages by avoiding the need to download files that haven't changed. When the normal cache reaches capacity, older files are deleted to make way for newer ones. The location of both types of cache is dependent on the browser.

The HTML5 specification doesn't prescribe any limit for the amount of disk space used by an application cache. The specification is equally vague about allowing users to delete specific application caches. Web developers should exercise their judgment about which files to make available offline and not fill up users' disk space unnecessarily.

Keeping the cache up to date

More often than not, updates to a site involve changing the contents of a file without changing its name. This presents a problem for the application cache. The browser checks only the filenames in the manifest. If they're the same, it assumes the cache doesn't need updating.

To force the browser to update the cache, you need to change the contents of the manifest. The simplest way to do this is to add a comment with a version number like this:

```
CACHE MANIFEST
# version 4
```

Increment the version number each time you make changes to the site, and upload the revised manifest after all the changes have been uploaded. You don't need to use a version number. Any unique value—such as a time-stamp—in a comment will do.

Serving the Manifest

You attach a manifest to a web page with the HTML5 `manifest` attribute in the opening `<html>` tag like this:

```
<html manifest="mysite.manifest">
```

The value of the `manifest` attribute should be a document-relative or site-root-relative path to the manifest file.

You should do this in every page in a site that you want to make available offline.

It's important to serve the manifest with the correct MIME type: `text/cache-manifest`.

Because this is a new MIME type, it might not be supported by all servers.

Setting the correct MIME type on Apache

If your web server runs on Apache, you should be able to configure it using an `.htaccess` file in your site root. If you already have an `.htaccess` file, add the following line to it:

```
AddType text/cache-manifest .manifest
```

If you don't have an `.htaccess` file, you can create one in Dreamweaver:

1. Choose File > New.
2. In the New Document dialog box, select Other from the list on the left, and set Page Type to Text. Click Create.
3. Type the following line of code into the new document, paying careful attention to spelling (Apache directives are case-sensitive):

```
AddType text/cache-manifest .manifest
```

4. Save the file in your site root with the name **.htaccess**. The name begins with a dot. Although it's a text file, make sure it's *not* saved with a `.txt` filename extension.

On Windows, the file will be saved as normal.

On a Mac, you'll see a warning that files with names that begin with a dot are reserved for the system and will be hidden (**Figure 4.2**). Click Use `."`. The file will be listed as normal in the Dreamweaver Files panel. However, you won't be able to see it in the Finder or any other Mac program unless it supports hidden files.



5. Upload the `.htaccess` file to your website.

Setting the MIME type on other web servers

If your website is on a server other than Apache, you need to ask the server administrator to enable the `text/cache-manifest` MIME type.



.htaccess

An `.htaccess` file is a mini-configuration file for the Apache web server. It has the advantage that all the settings are applied immediately without the need to restart the server. Normally, an `.htaccess` file is located in the site root and applies to the whole site. However, you can apply different settings to individual folders (directories) by placing an `.htaccess` file in the folder you want to control (the same settings apply to all subfolders unless overridden by another `.htaccess` file).

Most hosting companies configure their servers to allow site owners to fine-tune their settings with `.htaccess`. However, if you don't have permission to use `.htaccess`, you need to ask the server administrator to enable the `text/cache-manifest` MIME type.

Figure 4.2 On a Mac, Dreamweaver warns you that names beginning with a dot have special status.



As long as they're attached to a manifest, visited pages are stored in the application cache because a page that links to the manifest is automatically included in the explicit section. However, it's generally recommended that you list files individually rather than relying on this default behavior.



Web pages that use a server-side technology, such as PHP, ColdFusion, or ASP.NET, can also be linked to a manifest. However, the offline version stored by the application cache contains only the HTML output. For example, if the dynamic code outputs the current date, the version stored in the application cache displays the date when the online version was most recently accessed. As soon as you go back online, the stored date is updated.

Creating a "Lazy" Manifest

The HTML5 specification includes among its examples the following extremely simple manifest:

```
CACHE MANIFEST
FALLBACK:
/ /offline.html
NETWORK:
*
```

Instead of downloading all pages immediately, the browser stores only the fallback page (`offline.html`) and pages that are visited while the user is online. When the user goes offline, cached pages are retrieved from the user's application cache. But if the user clicks a link to a page that hasn't previously been visited, `offline.html` is displayed instead.

This lazy way of caching can be very useful on a large site. However, you still need to update the manifest with a version number or other unique value each time a page is edited. Otherwise, the old version of the page remains in the application cache.

Only HTML pages can be linked to a manifest. So, other resources—such as style sheets and images—are not stored in the application cache unless they're listed in the explicit section of the manifest.

Making the Tozai Hotel Site Available Offline

As you just learned, making a website available offline is a simple matter of generating the manifest, uploading it to your website, and making sure that it's served with the correct MIME type. The user's browser takes care of the rest. If the browser doesn't support offline web applications, it simply ignores the manifest.

The Tozai Hotel website consists of only 28 files, so typing out the manifest manually isn't a major chore, although it's important to get the spelling and path names right. However, life would be a lot easier if you could generate a file list automatically. So, I created a Dreamweaver extension to do it for you.

Installing the Generate Site Manifest Extension

The Generate Site Manifest extension is included in the download files for this book, and it takes only a minute or so to install.

1. Launch Adobe Extension Manager CS5.5 from within Dreamweaver or directly using one of the following methods:
 - ▶ Choose **Commands > Manage Extensions**.
 - ▶ Choose **Help > Manage Extensions**.
 - ▶ Launch the Extension Manager from the **Start** menu in Windows or from the **Finder** in Mac OS X.
2. Click the **Install** button in the Extension Manager title bar, and navigate to the **ch04/extension** folder in the download files.
3. Select **GenerateSiteManifest_1_0.mxp**, and click **Open** (Select on a Mac).
4. Read the **Extension Disclaimer** and choose to accept the terms. The extension should install immediately and display a brief description in the Extension Manager (**Figure 4.3**).

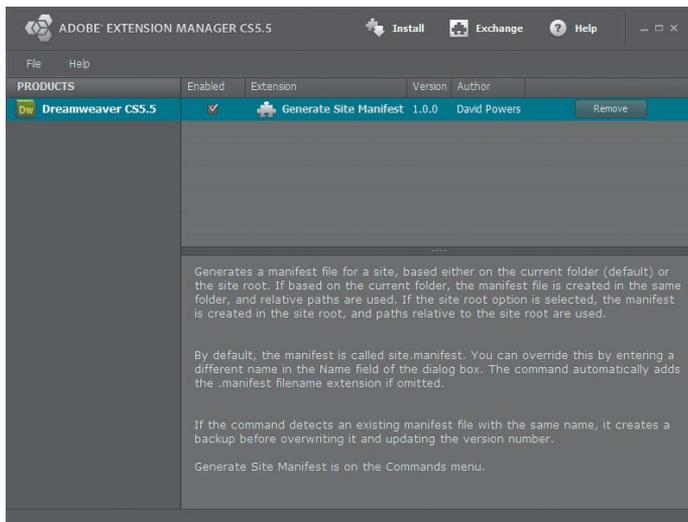


Figure 4.3 The Generate Site Manifest extension has been successfully installed.

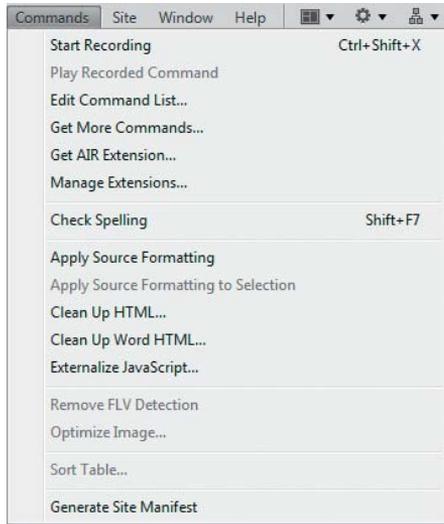


Figure 4.4 The extension adds a new item at the bottom of the Commands menu.



The extension should be enabled immediately in Dreamweaver. However, if it fails to appear at the bottom of the Commands menu, close and relaunch Dreamweaver.

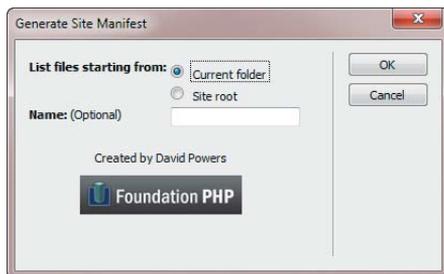


Figure 4.5 The Generate Site Manifest dialog box lets you choose the scope and name of the manifest.

5. The Generate Site Manifest extension should now be listed at the bottom of the Commands menu in Dreamweaver (**Figure 4.4**).
6. Close the Extension Manager.

Using the Generate Site Manifest Command

The Generate Site Manifest command installed by the extension inspects the site's folder structure and builds a list of all files (except manifests and their backups, and .htaccess files), which it stores in a manifest file ready for you to edit. The command's dialog box (**Figure 4.5**) has the following options:

- ▶ The radio buttons let you choose whether to list files starting from the current folder or the site root.
- ▶ If you choose the “Current folder,” all paths are relative to the folder, and the manifest is created in the same folder.
- ▶ If you choose “Site root,” the paths are relative to the site root and the manifest is created in the root folder.
- ▶ By default, the manifest is saved as site.manifest. However, you can change this by entering your own value in the Name text field. The command automatically adds the .manifest filename extension to the name.

When you run the command the first time, it sets the manifest's version number to 1. If the command detects an existing manifest with the same name, it saves a backup with a .manifest.bak filename extension before generating a new manifest with an updated version number. This avoids the need to build the online whitelist and fallback sections from scratch each time you generate a new manifest file. You can copy and paste them from the backup when editing the new file.

Try out the command with the Tozai Hotel files.

1. Open one of the HTML files in your working copy of the Tozai Hotel site. Alternatively, open one of the HTML files in ch03/complete.

2. Choose **Commands > Generate Site Manifest**.
3. Leave the options in the **Generate Site Manifest** dialog box at their default settings, and click **OK**.
4. If `site.manifest` doesn't immediately appear in the **Files** panel, click the  icon at the top of the panel to refresh its contents. You should now see `site.manifest` listed in the same folder as the file you opened (**Figure 4.6**).
5. Before you can edit the manifest file in Dreamweaver, you need to make a small adjustment to the program's preferences. Choose **Edit > Preferences (Dreamweaver > Preferences on a Mac)**, and select the **File Types / Editors** category from the list on the left.
6. In the "Open in code view" field, insert a space at the end of the existing list of filename extensions, and type **.manifest** (**Figure 4.7**).

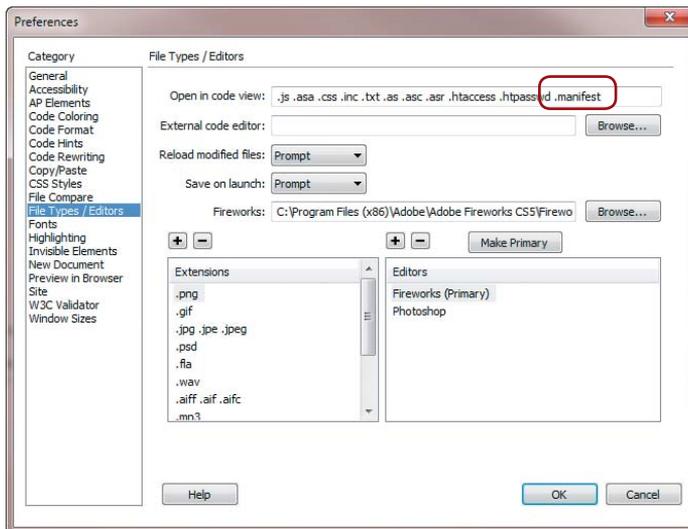


Figure 4.7 You need to add the `.manifest` filename extension to the list of files that Dreamweaver can edit.

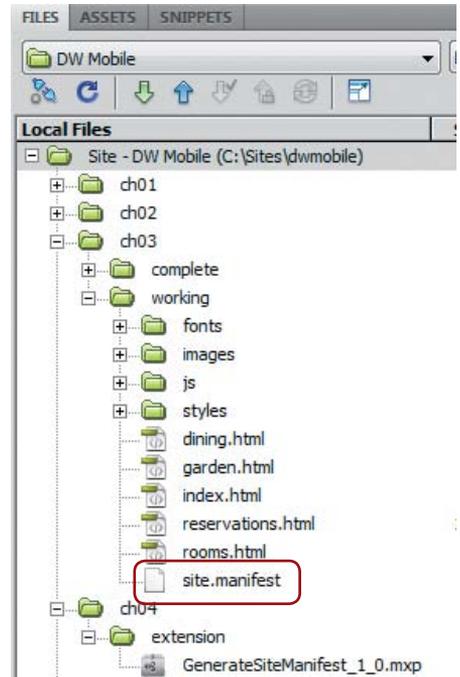


Figure 4.6 The manifest file has been created in the same folder.



Don't forget the period at the beginning of `.manifest`.

7. Click OK to close the Preferences dialog box.
8. In the Files panel, double-click `site.manifest` to open it in the Document window. You should see the following code:

```
CACHE MANIFEST
```

```
# version 1
```

```
dining.html  
garden.html  
index.html  
reservations.html  
rooms.html
```

```
fonts/Calluna-Regular-webfont.eot  
fonts/Calluna-Regular-webfont.svg  
fonts/Calluna-Regular-webfont.ttf  
fonts/Calluna-Regular-webfont.woff
```

```
images/basin_bg.jpg  
images/basin_bg_phone.jpg  
images/basin_bg_tab.jpg  
images/chef.jpg  
images/cherry_blossom.png  
images/exterior.jpg  
images/exterior_tab.jpg  
images/hotel-room.jpg  
images/sake.jpg  
images/sake_tab.jpg  
images/sashimi.jpg  
images/stone-lantern.jpg  
images/sushi.jpg
```

```
js/jquery-1.5.min.js
```

```
styles/desktop.css  
styles/phone.css  
styles/tablet.css  
styles/tozai.css  
styles/tozai_mq.css
```

You now have a complete list of files ready to divide into the explicit, online whitelist, and fallback sections.

9. Edit the code by adding an online whitelist section header before the list of font files like this:

```
CACHE MANIFEST

# version 1

dining.html
garden.html
index.html
reservations.html
rooms.html

NETWORK:
fonts/Calluna-Regular-webfont.eot
```

10. Save site.manifest and close it.
11. Run the Generate Site Manifest command again and refresh the Files panel if necessary. You should now have both site.manifest and site.manifest.bak in the same folder as the HTML file you opened.
12. Double-click site.manifest to open it. The first few lines should look like this:

```
CACHE MANIFEST

# version 2

dining.html
garden.html
index.html
reservations.html
rooms.html

fonts/Calluna-Regular-webfont.eot
```

The version number has changed, and the list has been generated anew, so the online whitelist section header has disappeared.



If you delete the existing manifest files, the version number reverts to 1. This is fine when experimenting before deploying a manifest file, but it could cause problems with a live site. If users have an earlier copy of the manifest with the same number, the updated files won't be downloaded.

13. Right-click `site.manifest.bak` and choose `Open with > Dreamweaver` from the context menu. The file contains the edit you made in step 9.

You can continue experimenting with the `Generate Site Manifest` command, selecting the option to list files starting from the site root, and changing the name.

Editing the Manifest File

When deciding how to organize your manifest file, it's a good idea to look at the size of the files in your site. Unlike media queries, you can't restrict what is cached by each type of device. It's an all-or-nothing decision. Unless you're careful, you could undo all the good work of your media queries by forcing mobile phones to download files they'll never use.

Overall, the Tozai Hotel site weighs in at 696 KB, broken down as follows:

- ▶ **Fonts.** 212 KB
- ▶ **Images.** 370 KB
- ▶ **JavaScript (external).** 83 KB
- ▶ **Style sheets.** 9 KB
- ▶ **HTML files.** 22 KB

Quite clearly, the bulk of the weight lies in the first three categories. The fonts are used purely for aesthetic reasons, so they can easily be sacrificed offline. The styles specify alternative fonts anyway. Many of the images are decorative, but the site would be less attractive and meaningful if you got rid of all of them. However, the external JavaScript file is used only by `reservations.html`, which is meaningless offline. Although the form isn't connected to a processing script in the example files, in a real website users would need to be online to submit a request about the availability of rooms. So, the external JavaScript can be dispensed with; and `reservations.html` needs to have a fallback page for offline use.

Losing the fonts, external JavaScript, and some of the images reduces the overall download by approximately half. You can't avoid serving all the style sheets to every device, but the size is trivial and could be reduced by eliminating comments and unnecessary whitespace.

Here's my suggested version of site.manifest for the Tozai Hotel site:

CACHE MANIFEST

version 1

dining.html
garden.html
index.html
rooms.html

images/basin_bg.jpg
images/chef.jpg
images/cherry_blossom.png
images/hotel-room.jpg
images/sashimi.jpg
images/stone-lantern.jpg
images/sushi.jpg

styles/desktop.css
styles/phone.css
styles/tablet.css
styles/tozai.css
styles/tozai_mq.css

FALLBACK:

images/basin_bg_phone.jpg images/basin_bg.jpg
images/basin_bg_tab.jpg images/basin_bg.jpg
reservations.html reservations_off.html

NETWORK:

fonts/Calluna-Regular-webfont.eot
fonts/Calluna-Regular-webfont.svg
fonts/Calluna-Regular-webfont.ttf
fonts/Calluna-Regular-webfont.woff

images/exterior.jpg
images/exterior_tab.jpg
images/sake.jpg
images/sake_tab.jpg



In a real-world situation, it would make more sense to use the same background image for all devices rather than serving smaller ones through media queries. Alternatively, you could add the background images to the online whitelist section to prevent them from being cached and display the site offline without the background image.

The following points should be noted:

- ▶ Only one version of the background image at the top of the page, `basin_bg.jpg`, is in the explicit section. It's 37 KB but is required for the desktop layout.
- ▶ The fallback section instructs browsers to replace `basin_bg_phone.jpg` and `basic_bg_tab.jpg` with the larger image, `basin_bg.jpg`, when offline. The styles for tablets and phones use the CSS3 `background-size` property to scale the image, so it looks the same in all devices.
- ▶ The fallback section tells browsers to substitute `reservations_off.html` for `reservations.html` when offline. This tells users to go online to check the availability of rooms (**Figure 4.8**).

Figure 4.8 When accessed offline, the reservations page displays a different message.



- ▶ In addition to the fonts, four images that are 183 KB in total have been added to the online whitelist section. This prevents them from being downloaded to the application cache. It means these particular images won't be available offline (**Figure 4.9**), but they're mainly decorative. However, they need to be listed explicitly here. Otherwise, they aren't displayed even when the user is online.
- ▶ The manifest results in browsers caching 177 KB, just 25 percent of the total size of the site.

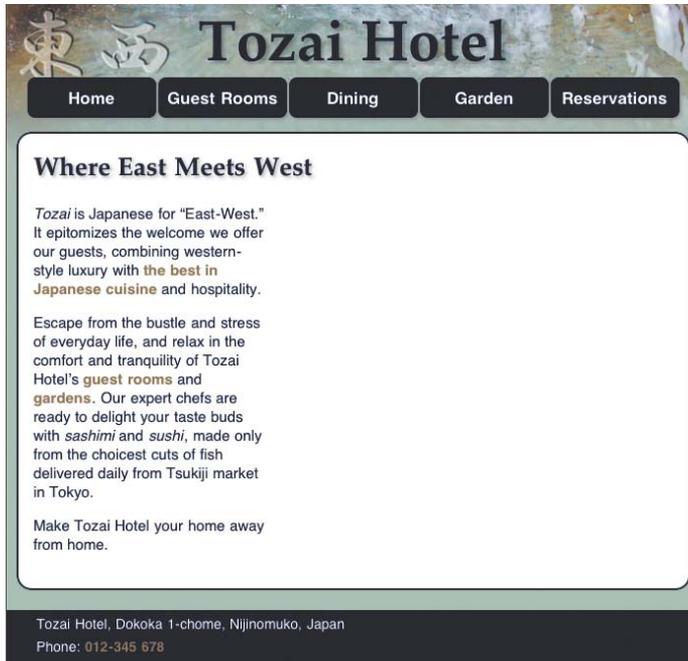
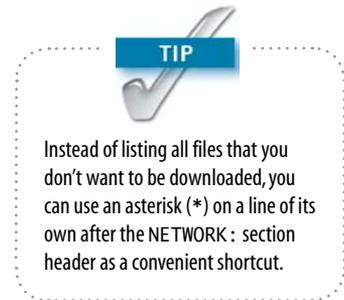


Figure 4.9 The exterior image isn't shown when the index page is viewed offline on a tablet.



Attaching the Manifest File

The manifest file needs to be attached to all web pages listed in the explicit section. However, it should not be attached to any pages that you don't want to be cached, because attaching a manifest automatically adds the file to the explicit section, even if it isn't listed there.

There are two ways to attach a manifest file in Dreamweaver:

- ▶ Manually in Code view
- ▶ With the Find and Replace dialog box

To attach a manifest file in Code view:

1. Position the insertion point just before the closing angle bracket of the opening `<html>` tag at the top of the page.
2. Insert a space to bring up code hints. Use your keyboard down arrow key or mouse to select `manifest` (**Figure 4.10**), and press Enter/Return or double-click. This inserts `manifest=""` and moves the insertion point to between the quotes.



Figure 4.10 Dreamweaver displays a code hint for `manifest` in the `<html>` tag.

3. Type **site.manifest** (or the name of your manifest file) between the quotes.

Alternatively, right-click and choose Code Hint Tools > URL Browser from the context menu. Click Browse, and navigate to the manifest file. Click OK (Choose on a Mac) to insert the filename and path.

In a small site like Tozai Hotel, attaching a manifest file manually to each HTML file takes only a couple of minutes, but you need a more efficient approach on a larger site. Dreamweaver doesn't have a dedicated dialog box to handle this, but the Find and Replace dialog box does the job quickly and easily.

This is how you do it:

1. In the Files panel, Ctrl-click/Command-click to select the files you want to attach the manifest file to (**Figure 4.11**).
2. Choose Edit > Find and Replace or press Ctrl+F/Command+F to open the Find and Replace dialog box.
3. Set "Find in" to **Selected Files in Site**.
4. Set Search to **Specific Tag**, and select **html** from the adjacent list.
5. If necessary, click the icon to remove further search option menus.
6. Set Action to **Set Attribute**, and select **manifest** from the adjacent list.
7. In the To field, type the name (and path, if necessary) of the manifest file. The settings in the Find and Replace dialog box should now look like **Figure 4.12**.

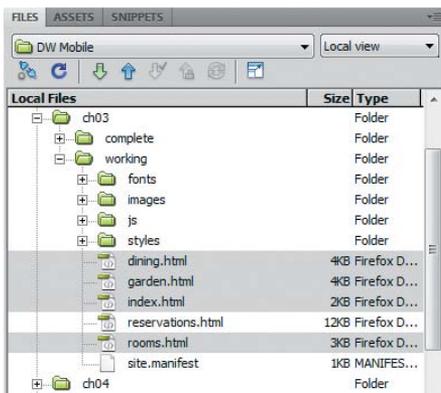
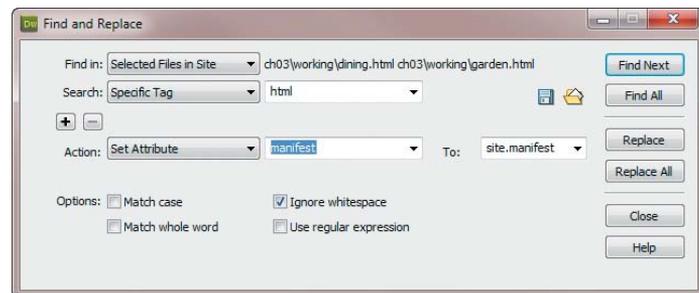


Figure 4.11 Select only the files that you want to be cached by the manifest.

Figure 4.12 Find and Replace offers a quick way to attach a manifest to multiple pages.



8. Click Replace All.
9. Dreamweaver warns you that the operation cannot be undone in files that are not currently open and asks you to confirm. Click Yes.
10. The Search tab of the Reports panel opens to display the changes (**Figure 4.13**).

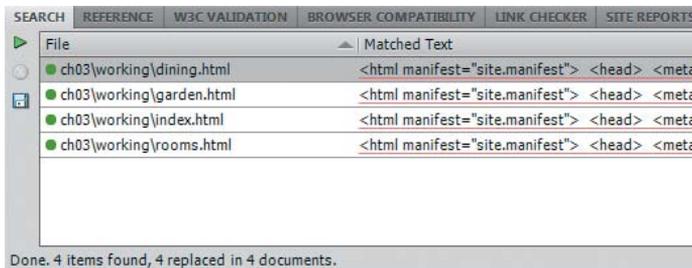


Figure 4.13 The Reports panel confirms that the `manifest` attribute has been added to the selected pages.



If you attach the wrong file or make a mistake in the path name, you can use the Find and Replace dialog box to change the value of the `manifest` attribute. You can also remove the `manifest` attribute by setting Action to Remove Attribute.

Right-click the gray area to the right of the tabs, and choose Close Tab Group to close the Reports panel.

Testing a Site Offline

As soon as you add a manifest file to the pages in a site, browsers that support offline web applications start caching the files. Once they're stored in the application cache, the browser relies on the manifest file to inform it of any changes. It's worth repeating that the manifest file needs to be updated not only when you add or remove files from the site, but also if existing pages are edited. Consequently, you should attach the manifest file only in the final stages of testing a site. Otherwise, you need to update the manifest's version number every time you make an adjustment to the site.

When you have decided the site's ready, create the manifest file, and attach it to the pages you want to make available offline. Then upload the manifest and web pages to your web server.

In theory, the application cache should be created and populated by visiting just one page. However, the time it takes for all files to be downloaded depends entirely on the browser and network conditions.

To test the application cache on a mobile device, disable all wireless connections:

- ▶ On iOS, choose Settings, and turn on Airplane Mode.
- ▶ On Android devices, choose Settings > Wireless and network(s), and tap Airplane mode or Flight mode to select it.
- ▶ On BlackBerry, choose Manage Connections, and tap Turn All Connections Off.

It might take a short while for the mobile device to disconnect from Wi-Fi and other networks.



Figure 4.14 In Flight mode, the Samsung Galaxy Tab warns about the lack of a network connection.



Figure 4.15 iOS tells you to turn off Airplane Mode and offers a shortcut to Settings.

Once disconnected, open the browser and navigate to the site. Usually, the browser displays a warning telling you there is no network connection (**Figure 4.14**) or telling you to turn off Airplane Mode (**Figure 4.15**).

Click OK to dismiss the alert. You should now be able to continue to the site, which should be loaded from the application cache. If you have specified an alternative page in the fallback section, it should be displayed instead of the normal page, as shown in Figure 4.8 earlier in this chapter.

If the alternative page fails to display or if images are missing, there are two likely explanations:

- ▶ The manifest file is not being served with the correct MIME type.
- ▶ The files are being served from the browser's normal cache rather than from the application cache.

A simple way to check whether the manifest file is being served with the correct MIME type is to try to load it directly in Firefox, Safari, or IE 9. If the browser asks if you want to save the file, the MIME type is probably OK. The Firefox dialog box actually confirms it as a manifest file (**Figure 4.16**). If the manifest opens in the browser as plain text, you need to check the `.htaccess` file or ask the server administrator to verify the MIME type.

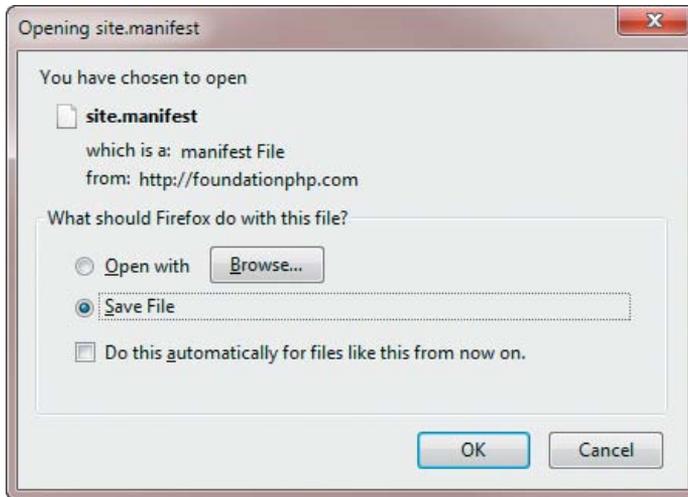


Figure 4.16 Firefox correctly identifies the MIME type.



At the time of this writing, Opera and Chrome open manifest files as plain text, even when they are served with the correct MIME type.

The second issue is not quite as easy to check. In my experiments on a small number of mobile devices, browsers appeared to use the application cache only if a file couldn't be found in the normal cache. For example, my iPad continued to display the online version of `reservations.html`, even offline. However, going back online and visiting several other sites cleared it out of the cache. Only then did the offline version display correctly.

Generally speaking, the fact that browsers store files in their local cache is beneficial. It avoids unnecessary downloads, saving bandwidth and speeding up the user's experience. However, you might want to add the following line to the `<head>` of pages that you don't want to be available offline:

```
<meta http-equiv="expires" content="-1">
```

This doesn't prevent the page from being cached, but it expires the page immediately, so the browser always fetches a new version. The downside of using this technique is that the page will always be downloaded afresh.

Going Offline

It doesn't take a great deal of effort to make a website available offline, although it's important to update the manifest file by adding a version number or another unique identifier each time you make any changes to the site's content. However, just because you can make a site available offline doesn't necessarily mean that you should. Ask yourself whether the site makes sense offline. Remember that a manifest forces the browser to download all files listed in the explicit section, taking up bandwidth and valuable disk space on the user's device. Firefox asks the user's permission to create an application cache, but most other browsers don't.

When creating a manifest, give careful thought to the size and importance of files you add to the explicit section. Are they really vital to the offline version of the site? If not, add them to the online whitelist section or specify substitutes in the fallback section.

All the techniques explored in Chapters 2–4 can be used in websites designed for a wide range of devices from desktops to mobile phones. The rest of the book is devoted to building websites and apps designed specifically for modern smartphones using the jQuery Mobile framework, which has been integrated into Dreamweaver CS5.5.

Index

A

Accelerometer mobile device feature, 230

accessibility. *See* ARIA

Accessible Rich Internet Applications. *See* ARIA

action attribute, 68

a-d values, `data-grid` attribute, 180–181

adjacent sibling selectors, 32

Adobe Dreamweaver CS5 with PHP: Training from the Source, 216

Adobe Extension Manager CS5.5, 125

a-f values, attributes

- `data-counttheme`, 181
- `data-dividertHEME`, 181
- `data-groupingtheme`, 181
- `data-split-theme`, 182
- `data-theme`, 182, 187–188
- `data-track-theme`, 182

::after CSS pseudo-element, 20

Ajax, online forms, 217

`alert()` method, 247

alert value, `data-icon` attribute, 184

and keyword, 75

Android

- Android Developers website, 270
- Android SDK, 221, 270
 - configuring Dreamweaver, 222–224
- display width and orientation, 81
- jQuery Mobile, 24, 147
- packaging apps for deployment, 6
- PhoneGap, 26–27, 221
- support
 - for CSS3 media queries, 72–73
 - for HTML5, 8
 - for last-of-type pseudo-class, 110
 - for offline applications, 119
- testing
 - offline web applications, 136
 - Travel Notes app, 266–269

Animations, Transforms, Transitions category, CSS Properties pane, 20

Apache web server, 122–123

Apple. *See also* iOS; iPad; iPhone; iPod

- Safari Web Inspector, 250, 261

WHATWG involvement, 11

Apple, Safari support

- for columns without values, 250
- for CSS3
 - `background-size` property, 100
 - @import rule, 84
 - media queries, 72–73
 - prefixes, 13
 - selectors, 18
- for drop shadows, 48
- for HTML5, 8
- for jQuery Mobile, 142
- for offline applications, 119
- manifest MIME type, 136
- for OTF, 40
- for TTF, 40
- for WOFF, 39

ARIA (Accessible Rich Internet Applications) roles, 36–39, 150–151

- Find and Replace settings, 39
- for HTML5 semantic elements, 37

`arrow-d` value, `data-icon` attribute, 184

`arrow-l` value, `data-icon` attribute, 184

`arrow-r` value, `data-icon` attribute, 184

`arrow-u` value, `data-icon` attribute, 184

<article> element, 8

- rule to apply font family/size, 8
- WAI-ARIA roles, 37

article role, 37

<article> tag, 236

<aside> element, 8

- WAI-ARIA roles, 37

Aside option, List View widget, 191

aspect ratio

- `aspect-ratio` media feature, 74
- jQuery Mobile, 162

ASP.NET

- jQuery Mobile, 144, 216
- manifests, 124

`attr()` method, 150, 254

attribute selectors, CSS, 17–18

auto property, margins, 109

autofocus attribute, 56

autoform value, `data-role` attribute, 177

B

 property, 34–36

Back and Home buttons, 172–173

back value, attributes

- `data-icon`, 184
- `data-rel`, 169–170, 182, 185–186

background images, hiding, 91–95

`background-image` property, 92, 109

`background-size` property, 100

banner role, 37

::before CSS pseudo-element, 20

`bind()` method, 66

BlackBerry OS

- display width and orientation, 81–82, 115
- jQuery Mobile, 24, 147
- PhoneGap, 220

support

- for CSS3 media queries, 72–73
- for last-of-type pseudo-class, 110
- by PhoneGap, lack of, 6
- testing offline web applications, 136

Blas, Kin, 142

`block` attribute, 109–110

#blossom style rule, 93

`border-radius` property, 13, 14, 16, 49–52

- Live view support, 17

bottom value, `data-iconpos` attribute, 184

`box-shadow` property, 14, 16, 45–49

- lack of Live view support, 17

browsers. *See also* specific browsers

- standard and quirks modes, 7
- support for HTML5, 6
- semantic elements, 7–8

Buivenga, Jos, 40

button value, `data-role` attribute, 165–166, 177

Button widget, 164–166, 189–190, 203–205

C

cache manifest, 5

CACHE MANIFEST, 119, 128–129

CACHE: section, 120–121

- FALLBACK: section, 121, 131
- NETWORK: section, 121, 129, 133
- version numbers, 122, 130
- Camera mobile device feature, 230
- caniuse.com, 7, 119
- <canvas> element, 12
- case sensitivity, HTML5, 7
- CDN (content distribution network), Google, 9
- Çelik, Tantek, 10
- change() method, 65–66
- check value, data-icon attribute, 184
- Checkbox widget, 189, 201–203
- Chrome (Google), support
 - for columns without values, 250
 - for CSS3
 - background-size property, 100
 - @import rule, 84
 - media queries, 72–73
 - prefixes, 13
 - selectors, 18
 - for drop shadows, 48
 - for HTML5, 8
 - for jQuery Mobile, 142
 - for offline applications, 119
 - manifest file MIME type, 137
 - for WOFF, 39
- Chrome Developer Tools, 250, 261
- click event, 251
- click() method, 264
- ColdFusion
 - jQuery Mobile, 144, 216
 - manifests, 124
 - parsing before validator submission, 67
- ColdFusion 9 Web Application Construction Kit*, 216
- collapsed value, data-state attribute, 182
- Collapsible Block widget, 189, 194–195
- collapsible blocks, 178–179
- collapsible value, data-role attribute, 177–178, 194
- collapsible-set value, data-role attribute, 177, 194
- color
 - drop shadows, 43–44, 47
 - values and opacity, 21–22
- color attribute, 47, 53
- color media feature, 74–75
- Color module, CSS3, 21
- color-index media feature, 74
- comma-separated values, box shadows, 46–47
- comparison operators, JavaScript, 214
- Compass mobile device feature, 230
- compatibility charts, CSS (Cascading Style Sheets), 18
- complementary role, 37
- conditional comments, IE, 76, 96–97
- Contacts mobile device feature, 230
- contain keyword, 100
- content distribution network. *See* CDN
- content value, data-role attribute, 149–150, 156–157, 164, 177
- contentinfo role, 37
- context-aware image sizing, 83
- controlgroup value, data-role attribute, 177, 202–203
- convertToMDY() function, 256
- Coordinated Universal Time. *See* UTC
- Copy Dependent Files dialog box problem, 185
- cover keyword, 100
- createElement() method, 9
- CSS (Cascading Style Sheets), 14–15. *See also* CSS3
 - compatibility charts, 18
 - IE (Internet Explorer), 7
 - jQuery Mobile, 142
 - Media Queries module, 72
 - PhoneGap, 220
 - predefined layouts, 9
 - progressive enhancement, 30–31
 - sprites, 147
 - vendor-specific prefixes, 13
- CSS3 (Cascading Style Sheets), 14–15. *See also* CSS
 - attribute selectors, 17–18
 - code hints, properties, 14
 - color values and opacity, 21–22
 - Document window size, 22–23
 - drop shadows, 16
 - embedded fonts, 17
 - Multiscreen Preview panel, 14, 22–23
 - properties, 20–21
 - pseudo-classes, 18–19
 - pseudo-elements, 19–20
 - rounded corners, 16
 - Styles panel Properties pane
 - new categories, 20–21
 - versus* Property inspector, 16
- custom data attributes
 - data-ajax, true/false values, 168, 181
 - data-backbtn, true/false values, 171, 181
 - data-back-btn-text, text values, 181
 - data-collapsed, true/false values, 177, 179, 181, 194
 - data-counttheme, a–f values, 181
 - data-direction, reverse values, 181, 187
 - data-dividertHEME, a–f values, 181
 - data-filter, true/false values, 181, 191
 - data-fullscreen, true/false values, 181
 - data-grid, a–d values, 180–181
 - data-groupingtheme, a–f values, 181
 - data-icon, 172, 181
 - values, 184
 - data-iconpos, 172, 181
 - values, 184
 - data-id, text values, 181, 183
 - data-inline, true/false values, 182
 - data-insert, true/false values, 182
 - data-native-menu, true/false values, 182, 200
 - data-placeholder
 - versus* HTML5 placeholder attribute, 210
 - true/false values, 182, 200
 - data-position, fixed/inline/fullscreen values, 182–183
 - data-rel
 - back/dialog values, 169–170, 182, 185–186
 - external value, 168, 170
 - data-role, 150, 160, 182
 - collapsible value, 177–178, 194
 - collapsible-set value, 177, 194
 - controlgroup value, 202–203
 - fieldcontain value, 195, 202
 - navbar value, 177, 179–180
 - nojs value, 177, 180
 - values, 177–178
 - data-split-icon, 182
 - data-split-theme, a–f values, 182
 - data-state, collapsed/horizontal/vertical values, 182
 - data-theme, a–f values, 182, 187–188
 - data-track-theme, a–f values, 182
 - data-transition, 182
 - fade value, 186
 - flip value, 185–186
 - pop value, 185–187
 - slide value, 187

- slidedown value, 187
 - slideup value, 187
 - data-type, horizontal/vertical values, 182, 203
- D**
- data attributes. *See* custom data attributes
 - data-ajax attribute, true/false values, 168, 181
 - data-backbtn attribute, true/false values, 171, 181
 - data-back-btn-text attribute, text values, 181
 - data-collapsed attribute, true/false values, 177, 179, 181, 194
 - data-counttheme attribute, a-f values, 181
 - data-direction attribute, reverse values, 181, 187
 - data-dividertHEME attribute, a-f values, 181
 - data-filter attribute, true/false values, 181, 191
 - data-fullscreen attribute, true/false values, 181
 - data-grid attribute, a-d values, 180–181
 - data-groupingtheme attribute, a-f values, 181
 - data-icon attribute, 172, 181 values, 184
 - data-iconpos attribute, 172, 181 values, 184
 - data-id attribute, text values, 181, 183
 - data-inline attribute, true/false values, 182
 - data-insert, true/false values, 182
 - <datalist> element, 53–56
 - cross-browser solutions, 54
 - data-native-menu attribute, true/false values, 182, 200
 - data-placeholder attribute
 - versus* HTML5 placeholder attribute, 210
 - true/false values, 182, 200
 - data-position attribute, fixed/inline/fullscreen values, 182–183
 - data-rel attribute
 - back/dialog values, 169–170, 182, 185–186
 - external value, 168, 170
 - data-role attribute, 150, 160, 182
 - collapsible value, 177–178, 194
 - collapsible-set value, 177, 194
 - controlgroup value, 177, 202–203
 - fieldcontain value, 195, 202
 - navbar value, 177, 179–180
 - nojs value, 177, 180
 - values, 177–178
 - data-split-icon attribute, 182
 - data-split-theme attribute, a-f values, 182
 - data-state attribute, collapsed/horizontal/vertical values, 182
 - data-theme attribute, a-f values, 182, 187–188
 - data-track-theme attribute, a-f values, 182
 - data-transition, 182
 - fade value, 186
 - flip value, 185–186
 - pop value, 185–187
 - slide value, 187
 - slidedown value, 187
 - slideup value, 187
 - data-trnote attribute, 259
 - data-type attribute, horizontal/vertical values, 182, 203
 - date() method, 248
 - date pickers, 57–67, 110
 - Datepicker widget, jQuery, 57, 67
 - dates and times
 - attributes
 - date, 13, 53
 - datetime, 53
 - datetime-local, 53
 - date type, 57
 - dateParts object, 59–60
 - <select> menus, 110
 - UI Datepicker widget, jQuery, 57, 67
 - UTC (Coordinated Universal Time), 60
 - default value, data-iconpos attribute, 184
 - delete value, data-icon attribute, 184
 - deleteItem() function, 258–259
 - Desire (HTC)
 - display width and orientation, 81, 107
 - embedded fonts, 98
 - navigation menu, 109
 - desktop computers, style rules, 82–84
 - desktop.css, 95–97
 - device-aspect-ratio media feature, 74
 - device-height media feature, 74, 78–80
 - device-width media feature, 74, 78–79
 - <dfn> tags, 34
 - dialog boxes, 185
 - dialog value, data-rel attribute, 182
 - display property, 83, 109–110
 - displayMap() function, 256–257, 263–265
 - <div> elements
 - nested <div> elements, 24
 - universally supported, 36
 - DOCTYPE and <!DOCTYPE HTML> declarations, case sensitivity, 7
 - Document Object Model. *See* DOM
 - Document window viewport, sizing, 95, 102
 - DOM (Document Object Model), 12
 - jQuery Mobile, 142, 147, 157, 166, 213
 - dot notation *versus* square bracket notation, 61
 - dpc (dots per centimeter), 75
 - dpi (dots per inch), 75
 - Dreamweaver CS5 with PHP: Training from the Source*, 68
 - Dreamweaver CS5.5
 - code hints
 - for CSS properties, 14
 - for HTML5 tags, 14–15
 - for jQuery Core, 14–15, 25–26
 - CSS3, 14–15
 - attribute selectors, 17–18
 - color values and opacity, 21–22
 - Document window size, 22–23
 - drop shadows, 16
 - embedded fonts, 17
 - Multiscreen Preview panel, 14, 22–23
 - properties, 20–21
 - pseudo-classes, 18–19
 - pseudo-elements, 19–20
 - rounded corners, 16
 - Styles panel Properties pane, new categories, 20–21
 - Styles panel Properties pane, *versus* Property inspector, 16
 - development for multiple devices, 27
 - HTML5
 - code hints, 14–15
 - editing tags manually, 15
 - jQuery Mobile, 14–15, 24–26
 - media query handling, 14
 - PhoneGap, 14, 26–27
 - Property inspector, 15
 - versus* Properties pane, CSS Styles panel, 16

- support, lack of, for role attribute, 37
- Tag Inspector, Behaviors tab, 15
- W3C validator, 67
- Windows version, no support for iOS, 26
- drop shadows, 13, 16
 - to page elements, 45–49
 - to text, 43–45

E

- editItem() function, 258–259
- property, 34–36
- em unit of measure, 75
- email attribute, 53
- embedded fonts, 17, 39–42, 91–95
- Embedded Open Type (EOT), 40
- :empty() CSS pseudo-class, 19
- EOT (Embedded Open Type), 40
- executeSql() method, 241, 252
- external value, data-rel attribute, 168, 170

F

- fade value, data-transition attribute, 186
- fieldcontain value, data-role attribute, 177
- <figcaption> element, 8
- <figure> element, 8
- Filament Group, 83
- File mobile device feature, 230
- Find and Replace dialog box, 37–39
- Firefox (Mozilla), support
 - for CSS3
 - background-size property, 100
 - @import rule, 84
 - media queries, 72–73
 - prefixes, 13
 - selectors, 18
 - for HTML5, 8
 - for jQuery Mobile, 142
 - for offline applications, 119
 - manifest MIME type, 136
 - for WOFF, 39
- :first-child() CSS pseudo-class, 19
- ::first-letter CSS pseudo-element, 20
- ::first-line CSS pseudo-element, 20
- :first-of-type() CSS pseudo-class, 19
- fixed value, data-position attribute, 182
- Flip Toggle Switch widget, 190, 207

- flip value, data-transition attribute, 185–186
- @font-face rule, 17, 39–42, 91
 - wrapping in @media rule, 99–100
- fonts
 - embedded fonts, 17, 39–42
 - @font-face declaration, 39–42
 - online font library services, 40
- <footer> element, 7–8, 235–236, 256
 - WAI-ARIA roles, 37
- footer value, data-role attribute, 149, 164, 177
- footers, 155–156, 183
- forms, 52–53
 - client-side validation, 52
 - date pickers, 57–66
 - editable drop-down menus, 53–56
- elements
 - <datalist>, 53–56
 - <form>, 195
 - <input>, 52–53
 - <select>, 53–56
- HTML5 attributes, 56–57
 - autofocus, 56
 - date type, 57
 - max, 56
 - min, 56
 - placeholder, 56
 - required, 56
- jQuery Mobile
 - with Radio Button widget, 211–212
 - with Select Menu widget, 208–209
 - with Select Menu widget, replacing with text input field, 213–216
 - with Slider widget, 211–212
 - submitting and displaying response, 216–218
 - with Text Input widget, 209–211
 - spaces, 196
- Forta, Ben, 216
- forward value, data-icon attribute, 184
- fullscreen value, data-position attribute, 182

G

- Gartner research company, 4
- gear value, data-icon attribute, 184
- Generate Site Manifest extension, 125–130
- Geolocation mobile device feature, 230
- getCurrentPosition() method, 245–247

- getElementByClassName() method, 26
- getElementById() method, 26
- getItem() function, 254–256, 259, 263
- getLocation() function, 251
- getNextDay() function, 59–60, 60–61
- getNumDays() function, 62–63
- getTitles() function, 251, 259–261
- Gillenwater, Zoe Micklely, 19, 40
- Google
 - CDN (content distribution network), 9
 - Chrome Developer Tools, 250, 261
 - Google Maps, 265
 - Google Static Maps APIs, 265
 - Google's Chrome, support
 - for columns without values, 250
 - for CSS3
 - background-size property, 100
 - @import rule, 84
 - media queries, 72–73
 - prefixes, 13
 - selectors, 18
 - for drop shadows, 48
 - for HTML5, 8
 - for jQuery Mobile, 142
 - for offline applications, 119
 - manifest file MIME type, 137
 - for WOFF, 39
 - grid media feature, 74–75
 - grid value, data-icon attribute, 184

H

- <h1> tags, ARIA rules, 37
- handheld value, media attribute, 72
- <header> element, 7–8
 - ARIA restrictions, 37
 - WAI-ARIA roles, 37
 - wrapping in <div> tags, 36
- header value, data-role attribute, 149, 156–158, 164, 170–171, 177
- headers, 157–158, 169–171, 183
- height media feature, 74, 78–79
- height property, 80, 101
 - removing from HTML tags, 83
- #hero style rule, 92, 101, 109
- hexadecimal notation, 21
- Hickson, Ian, 6, 10
- home value, data-icon attribute, 184
- horizontal value, attributes
 - data-state, 182
 - data-type, 182, 203
- HSL (hue, saturation, lightness), CSS3 Color module, 21

- hs1a() method, 21–22
- .htaccess file, 123
- HTC Desire
 - display width and orientation, 81, 107
 - embedded fonts, 98
 - navigation menu, 109
- HTML
 - development history, 10–13
 - “living standard,” 11–12
 - poor markup encouragement, 13–14
 - versus* XML (Extensible Markup Language), 10
- html() method, 253
- HTML5
 - assistive technology for disabled, 10
 - backwards compatibility, 7
 - cache manifest, 5
 - caniuse.com, 7
 - case sensitivity, 7
 - code hints, 14–15
 - converting from XHTML 1.0 Strict, 5
 - custom data attributes, 150
 - editing tags manually, 15
 - HTML 4.01 compatibility, 7
 - JavaScript default, 9
 - jQuery Mobile, 142, 144, 147–151, 154, 165–166
 - logo, 10
 - new elements and attributes, 7
 - PhoneGap, 220
 - Travel Notes app, 231–237
 - progressive enhancement, 30–31
 - semantic elements, 7
 - style sheet rule for partial support browsers, 8
 - W3C specification approval process, 6
- HTML5 Now*, 10
- I**
 - <i> property, 34–36
 - icons, adding to buttons, 184–185
 - IE (Internet Explorer), Microsoft conditional comments, 76
 - and Netscape, 30
 - support or lack of, 136
 - for background-size property, 100
 - for CSS, 7, 9
 - for CSS3, @import rule, 84
 - for CSS3, media queries, 72–73, 76–77
 - for CSS3, selectors, 18–19
 - for drop shadows, 48
 - for EOT, 40
 - for HTML5, 6, 8
 - for HTML5, workarounds, 9–10
 - for jQuery Mobile, 142
 - for offline applications, 119
 - for offline applications, manifest file MIME type, 136
 - for WOFF, 39
 - images. *See also* inline images
 - context-aware image sizing, 83
 - embedded, 83
 - hiding background images, 91–95
 - tags, 83
 - width attribute, 109
 - @import media rule
 - conditions, 77–78
 - style sheets, 84
 - Indexed Database API, 240
 - info value, data-icon attribute, 184
 - initial-scale property, 80
 - inline images, 83. *See also* images
 - floatleft and floatright classes, 109
 - inline value, data-position attribute, 182
 - innerHTML property, 12
 - <input> element, 190, 201, 204–206
 - attributes available, 15
 - insertEntry() method, 247, 249
 - Inset option, List View widget, 191
 - Introducing HTML5*, 9, 56
 - Invisible Elements widget, 154
 - iOS
 - display width and orientation, 81
 - iOS SDK
 - configuring Dreamweaver, 222–224
 - downloading, 221
 - jQuery Mobile, 24, 147
 - packaging apps for deployment, 6
 - PhoneGap, 26–27, 220–221
 - configuring Dreamweaver, 222–228
 - support
 - for CSS3 media queries, 72–73
 - for last-of-type pseudo-class, 110
 - for offline applications, 119
 - testing
 - offline web applications, 136
 - Travel Notes app, 266–269
 - iPad (Apple)
 - device-height media feature, 80
 - display width and orientation, 81, 98
 - mobile Internet access, 4
 - PhoneGap, 26
 - testing, Travel Notes app, 266–269
- iPhone (Apple)
 - display width and orientation, 81
 - media features
 - device-height, 78–80
 - device-width, 78–79
 - max-width, 78–79
 - min-width, 78–79
 - width, 78–79
 - PhoneGap, 26
 - testing, Travel Notes app, 266–269
- iPod (Apple)
 - display width and orientation, 81, 115
 - media features
 - device-height, 78–80
 - device-width, 78–79
 - max-width, 78–79
 - width, 78–79
 - PhoneGap, 26
- Items option, List View widget, 191
- J**
 - Java *versus* JavaScript, 26
 - JavaScript
 - default for HTML5, 9
 - DOM methods, 26
 - versus* Java, 26
 - jQuery Mobile, 5, 142
 - PhoneGap, 26, 220
 - JavaScript-disabled content, 180
 - jQuery Core library, 237–238
 - jQuery html() method, 62
 - jQuery Mobile (Local) Mobile Starter, 144, 163
 - jQuery Mobile (PhoneGap) Mobile Starter, 144, 224
 - jQuery Mobile (CDN-content distribution network) Mobile Starter, 144
 - jQuery/jQuery Mobile, 5, 14–15, 24–26
 - code hints, 14–15, 25–26
 - data attributes (*See* custom data attributes)
 - DOM (Document Object Model), 142, 147, 157, 166, 213
 - forms
 - with Radio Button widget, 211–212
 - with Select Menu widget, 208–209
 - with Select Menu widget, replacing with text input field, 213–216
 - with Slider widget, 211–212

- submitting and displaying response, 216–218
 - with Text Input widget, 209–211
 - HTML5, 142, 144, 147–151, 154, 165–166
 - library, 237–238
 - Library Source field, 163
 - mobile site creation, 143–144
 - Back and Home buttons, 172–173
 - collapsible blocks, 178–179
 - dialog boxes, 185
 - footers, 155–156, 183
 - headers, 157–158, 169–171, 183
 - icons, adding to buttons, 184–185
 - IDs, 164
 - JavaScript-disabled content, 180
 - linking to external pages, 163–169
 - Mobile Starters, 144–147
 - Mobile Starters, adding content, 153–155
 - navigation bars, 179–180
 - page transitions, 186–187
 - static *versus* dynamic pages, 144
 - structure, 147–153
 - text, 156–157
 - themes, 187–188
 - processing data input with server-side technology, 216–218
 - ThemeRoller tool, 188
 - updating files, 143, 163
 - widgets
 - Button, 164–166, 189–190, 203–205
 - Checkbox, 189, 201–203
 - Collapsible Block, 189, 194–195
 - Flip Toggle Switch, 190, 207
 - insertion point importance, 158
 - Invisible Elements, 154
 - Layout Grid, 189, 192–194
 - List View, 158–162, 166–168, 189–192, 231–233, 250, 253–254, 259–261
 - Loading, 165
 - Page, 166, 189
 - Password Input, 189, 197
 - Radio Button, 189, 203, 211–212
 - Select Menu, 189, 198–201, 208–209
 - Select Menu, replacing with text input field, 213–216
 - Slider, 190, 205–206, 211–212
 - Text Area, 189, 198
 - Text Input, 189, 195–197, 209–211, 233
- ## K – L
- Keith, Jeremy, 54–55
 - Koch, Peter-Paul, 18
 - <label> tags, 198, 202
 - <lang> tag, 34, 36
 - :last-child() CSS pseudo-class, 19
 - :last-of-type() CSS pseudo-class, 19, 110
 - Lawson, Bruce, 9, 56
 - Layout Grid widget, 189, 192–194
 - lazy manifests, 124
 - left value, data-iconpos attribute, 184
 - <legend> tags, 202
 - Less Than or Equal to IE & Conditional Comment, 96
 - Line Layout category, CSS Properties pane, 20
 - <link> tag, style sheets, 84, 97
 - List Type option, List View widget, 191
 - List View widget, 158–162, 166–168, 189–192, 231–233, 250, 253–254, 259–261
 - list-divider value, data-role attribute, 160, 162, 167, 177
 - listview() method, 253
 - listview value, data-role attribute, 149, 160, 162, 167, 177
 - LiveScript. *See* JavaScript
 - Loading widget, 165
- ## M
- main role, 37
 - .manifest filename extension, 119, 127
 - manifests, offline websites, 118
 - CACHE MANIFEST listings, 120–121, 129
 - creating, 119–120
 - files, attaching, 133–135
 - files, editing, 130–133
 - lazy manifests, 124
 - manifest attribute, 122–123
 - PHP, ColdFusion, or ASP.NET files, 124
 - serving, 122–123
 - site manifest extensions, 125–130
 - up-to-date caches, 122
 - version numbers, 122, 126, 130
 - max attribute, 15, 56
 - maximum-scale property, 80
 - max-width media feature, 74, 78–79
 - media attribute, 5
 - handheld value, 72
 - Media mobile device feature, 230
 - media queries, 5, 14. *See also* Media Queries module, CSS3
 - assessing, 115–116
 - CSS comments, 89
 - inline images, 83
 - site-wide files, 84–90
 - Media Queries dialog box, 85–90
 - Media Queries module, CSS3, 72
 - @media rules, 101, 110
 - conditions, 77–78
 - @font-face rule, wrapping in, 99–100
 - style sheets, 83
 - <meta> tag, 79–80
 - Microsoft category, CSS Properties pane, 20–21
 - Microsoft's IE (Internet Explorer)
 - conditional comments, 76
 - and Netscape, 30
 - support or lack of, 136
 - for background-size property, 100
 - for CSS, 7, 9
 - for CSS3, @import rule, 84
 - for CSS3, media queries, 72–73, 76–77
 - for CSS3, selectors, 18–19
 - for drop shadows, 48
 - for EOT, 40
 - for HTML5, 6, 8
 - for HTML5, workarounds, 9–10
 - for jQuery Mobile, 142
 - for offline applications, 119
 - for offline applications, manifest file MIME type, 136
 - for WOFF, 39
 - MIME type, 122–123, 136–137
 - min attribute, 15, 56
 - min-height media feature, 101
 - minimum-scale property, 80
 - minus value, data-icon attribute, 184
 - min-width media feature, 74, 78–79
 - Mobile Starters, 144–147
 - adding site content, 153–155
 - Mobile Starter page, 24–25
 - updating pages, 143
 - \$.mobile.changePage() method, 249, 259, 265
 - mobile-init event, 238
 - modulo division, 63

- monochrome media feature, 74–75
- month attribute, 53
- moz- prefix, 13
- Mozilla category, CSS Properties pane, 20
- Mozilla's Firefox, support
 - for CSS3
 - background-size property, 100
 - @import rule, 84
 - media queries, 72–73
 - prefixes, 13
 - selectors, 18
 - for HTML5, 8
 - for jQuery Mobile, 142
 - for offline applications, 119
 - manifest MIME type, 136
 - for WOFF, 39
- Mozilla's Firefox, WHATWG, 11
- Multi-column Layout category, CSS Properties pane, 20
- Multiscreen Preview panel, 14

N

- <nav> element, 7–8
 - WAI-ARIA roles, 37
- #nav rule, 101, 107–108
- navbar value, data-role attribute, 177, 179–180
- navigation bars, 179–180
- navigation role, 37
- Netscape
 - CSS3, @import rule, 84
 - IE (Internet Explorer), Microsoft, 30
 - quirks mode, 7
- nojs value, data-role attribute, 177, 180
- non-breaking spaces, 165
- none value, data-role attribute, 177
- note role, 37
- notext value, data-iconos attribute, 184–185
- Notification mobile device feature, 230
- :nth-child() CSS pseudo-class, 18–19, 109
- :nth-last-child() CSS pseudo-class, 19
- :nth-last-of-type() CSS pseudo-class, 19
- :nth-of-type() CSS pseudo-class, 18–19
- number attribute, 13, 53

O

- offline websites
 - manifests, 118

- CACHE MANIFEST listings, 120–121, 129
 - creating, 119–120
 - files, attaching, 133–135
 - files, editing, 130–133
 - lazy manifests, 124
 - manifest attribute, 122–123
 - PHP, ColdFusion, or ASP. NET files, 124
 - serving, 122–123
 - site manifest extensions, 125–130
 - up-to-date caches, 122
 - version numbers, 122, 126, 130
- offline web applications, 118–119
 - going offline, 138
 - online access only files, 121
 - testing offline, 135–137
- only keyword, 75, 78
- :only-child() CSS pseudo-class, 19
- :only-of-type() CSS pseudo-class, 19
- opacity property, 21–22, 44
- Open Type (OTF), 40
- openDatabase() method, 240–242
- Opera
 - support
 - for CSS3, background-size property, 100
 - for CSS3, @import rule, 84
 - for CSS3, media queries, 72–73
 - for CSS3, selectors, 18
 - for drop shadows, 48
 - for HTML5, 8
 - for jQuery Mobile, 142
 - for offline applications, 119
 - for offline applications, manifest file MIME type, 137
 - for OTF, 40
 - for TTF, 40
 - WHATWG involvement in W3C, 11
- Opera category, CSS Properties pane, 20–21
- <option> elements, 198
- orientation media feature, 74
- OTF (Open Type), 40
- overflow property, 157

P

- </p> tag, omitted in HTML5, 14
- <p> tag, rule to apply font family/size, 8
- packaging apps for deployment, 6
- page transitions, 186–187

- page value, data-role attribute, 149–150, 157, 164, 171, 177
- Page widget, 166, 189
- pagebefoeshow event, 250–251
- Palm WebOS, jQuery Mobile, 147
- Password Input widget, 189, 197
- percentages, units of measure, 100
- Perl, jQuery Mobile, 216
- PhoneGap, 14, 26–27
 - API access to mobile device features, 230–231
 - jQuery Mobile (PhoneGap) Mobile Starter, 144, 224–228
 - library, 237–238
 - packaging apps for deployment, 6
 - startup screen for Windows Phone OS, 228–230
 - support for Android and iOS, 6
 - Travel Notes app, 230–231
 - building and testing, 266–269
 - database, 242–244
 - database, current location, 244–250
 - database, displaying record details, 254–257
 - database, displaying records, 250–254
 - database, inserting data, 244–250
 - database, updating and deleting items, 257–262
 - HTML structure, 231–237
 - map, displaying, 262–266
 - programming, 237–241
 - removing from simulator, 269–270
 - up-to-date features, 220
- PHP (PHP Hypertext Preprocessor)
 - jQuery Mobile, 144, 216
 - form-processing scripts, 201
 - manifests, 124
 - parsing before validator submission, 67
- placeholder attribute, 15, 56
 - versus jQuery Mobile data-placeholder attribute, 210
- plus value, data-icon attribute, 184
- pop value, data-transition attribute, 185–187
- populateDate() function, 61–62, 65
- populateYear() function, 61–62
- preferences
 - bold and italics, 35
 - File Types / Editors, filename extensions, 127
 - window sizes, 111
- preventDefault() method, 249
- Property inspector, 15

versus Properties pane, CSS Styles panel, 16
 pseudo-classes, CSS3, 18–19
 pseudo-elements, CSS3, 19–20
 px unit of measure, 75, 100

Q – R

quirks mode, browsers, 7, 18
 quotation marks, caution, 253
 Radio Button widget, 189, 203, 211–212
 range attribute, 53
 refresh value, *data-icon* attribute, 184
 region role, 37
 Related Files toolbar, 83, 89, 91
 rel="external" data attribute, 168, 170
 required attribute, 15, 56
 resetDates() function, 65–66
 resolution media feature, 74–75
 RGB (red, green, blue) values, CSS3 Color module, 21
 rgba(), 21–22
 right value, *data-iconpos* attribute, 184
 role attribute, WAI-ARIA, 36–39
 :root CSS pseudo-class, 19
 rounded corners, 13, 16, 49–52
 rows property, 253

S

Safari (Apple), support
 for columns without values, 250
 for CSS3
 background-size property, 100
 @import rule, 84
 media queries, 72–73
 prefixes, 13
 selectors, 18
 for drop shadows, 48
 for HTML5, 8
 for jQuery Mobile, 142
 for offline applications, 119
 manifest MIME type, 136
 for OTE, 40
 for TTF, 40
 for WOFF, 39
 Safari Web Inspector, 250, 261
 #sake style rule, 92–93, 109
 Samsung Galaxy Tab, display width and orientation, 81–82, 98
 scan media feature, 74
 <script> tags, 59, 239
 search attribute, 53
 search role, 37

search value, *data-icon* attribute, 184
 <section> element, 7–8
 WAI-ARIA roles, 37
 <select> elements, 189, 198–201, 207
 date pickers, 57–65
 text input field disadvantage, 216
 Select Menu widget, 189, 198–201, 208–209
 replacing with text input field, 213–216
 SELECT query, 252–254
 semantic elements, HTML5
 browser support, 7–10
 style sheet rule for partial support browsers, 8
 Code view, adding elements, 15
 new, 7
 WAI-ARIA roles, 36–37
 setValues() function, 60–61
 Sharp, Remy, 9, 56
 show() method, 254
 slide value, *data-transition* attribute, 187
 slidedown value, *data-transition* attribute, 187
 slider value, *data-role* attribute, 177
 Slider widget, 190, 205–206, 211–212
 slideup value, *data-transition* attribute, 187
 Snippets panel, 95–97
 tag, 256
 Specify Site-wide Media Query dialog box, 86–87
 Split Button Icon option, List View widget, 191
 Split Button option, List View widget, 191
 sprites, CSS (Cascading Style Sheets), 147
 SQL (Structured Query Language), 221
 Indexed Database API, 240
 SQL injection, 241
 Web SQL Database API, 240–242
 SQL injection, 241
 SQLite, 221
 square bracket notation *versus* dot notation, 61
 star value, *data-icon* attribute, 184
 Storage mobile device feature, 230
 property, 34–36
 Structured Query Language. *See* SQL
 Stunning CSS3, 19, 40
 <style> block, 236
 @import media rule, 84
 Styles panel Properties pane, CSS3

 new categories, 20–21
 versus Property inspector, 16
 styles/style sheets
 external, attaching, 84–90
 jQuery Mobile, 155
 hiding from earlier browsers, 75–77
 for mobile phones, 112–115
 for multiple devices, 84
 organizing, 82–84
 for tablets, 98–115
 for varying screen widths, 111–112
 swapList() function, 260, 262
 Symbian S60, jQuery Mobile, 24

T

Tag Inspector
 Attributes tab, 160
 Behaviors tab, 15
 Taylor, Jorge, 142
 tel attribute, 53
 tenary operator, 65
 Text Area widget, 189, 198
 Text Bubble option, List View widget, 191
 Text Description option, List View widget, 191
 Text Input widget, 189, 195–197, 209–211, 233
 text values, attributes
 data-back-btn-text, 181
 data-id, 181, 183
 <textarea> element, 189
 <textarea> tag, 198
 text-shadow property, 14, 17, 43–45
 themes, 187–188
 time attribute, 53
 times and dates
 attributes
 date, 13, 53
 datetime, 53
 datetime-local, 53
 date type, 57
 dateParts object, 59–60
 <select> menus, 110
 UI DatePicker widget, jQuery, 57, 67
 UTC (Coordinated Universal Time), 60
 toLowerCase() function, 256–265
 top value, *data-iconpos* attribute, 184
 Travel Notes app, 230–231
 building and testing, 266–269
 database, 242–244
 current location, 244–250
 displaying record details, 254–257

- displaying records, 250–254
 - inserting data, 244–250
 - updating and deleting items, 257–262
 - HTML structure, 231–237
 - map, displaying, 262–266
 - programming, 237–241
 - removing from simulator, 269–270
 - true/false values, attributes
 - data-ajax, 168, 181
 - data-backbtn, 171, 181
 - data-collapsed, 177, 179, 181, 194
 - data-filter, 181, 191
 - data-fullscreen, 181
 - data-inline, 182
 - data-insert, 182
 - data-native-menu, 182, 200
 - data-placeholder, 182, 200
 - TrueType (TTF), 40
 - try/catch blocks, 247
 - TTF (TrueType), 40
 - type attributes, 13
 - Typekit, 40
- U**
- UI Datepicker widget, jQuery, 57, 67
 - ui-btn-active class, 152
 - ui-mobile-viewport-transitioning class, 152
 - elements, 36
 - Unicode (UTF-8), 148
 - updateItem() function, 258–259
 - url attribute, 53
 - User Interface category, CSS Properties pane, 20
 - user-scalable property, 80
 - UTC (Coordinated Universal Time), 60
- V**
- val() method, 61
 - validation of adapted pages, 67–68
 - vertical value, attributes
 - data-state, 182
 - data-type, 182, 203
 - viewport <meta> tag, properties, 79–80
- W**
- W3C (World Wide Web Consortium)
 - ARIA (Accessible Rich Internet Applications) specification, 37
 - DOM (Document Object Model), 12
 - HTML5
 - logo, 10
 - specification approval process, 6
 - HTML5 specification approval process, 6
 - Indexed Database API, 240
 - language tags usage, 36
 - media attribute rules, 72
 - media queries specification, 75
 - Web SQL Database API, 240
 - WOFF (Web Open Font Format), 39–40
 - W3C validator, 67
 - WAI (Web Accessibility Initiative)-ARIA (Accessible Rich Internet Applications) roles, 36–39, 150–151
 - Find and Replace settings, 39
 - for HTML5 semantic elements, 37
 - Web Accessibility Initiative. *See* WAI
 - web applications, offline, 118–119.
 - See also* offline websites
 - going offline, 138
 - online access only files, 121
 - testing offline, 135–137
 - Web Open Font Format (WOFF), 39–40
 - web safe fonts, 17, 40
 - Web SQL Database API, 240–242
 - WebKit browsers, CSS3 prefixes, 13
 - Webkit category, CSS Properties pane, 20–21
 - webkit property, 13, 21
 - webkit-box-shadow property, 17
 - week attribute, 53
 - WHATWG (Web Hypertext Application Technology Working Group)
 - FAQs, 14
 - HTML5 development, 11
 - white-space property, 157
 - widgets, jQuery Mobile
 - Button, 164–166, 189–190, 203–205
 - Checkbox, 189, 201–203
 - Collapsible Block, 189, 194–195
 - Flip Toggle Switch, 190, 207
 - insertion point importance, 158
 - Invisible Elements, 154
 - Layout Grid, 189, 192–194
 - List View, 158–162, 166–168, 189–192, 231–233, 250, 253–254, 259–261
 - Loading, 165
 - Page, 166, 189
 - Password Input, 189, 197
 - Radio Button, 189, 203, 211–212
 - Select Menu, 189, 198–201, 208–209
 - replacing with text input field, 213–216
 - Slider, 190, 205–206, 211–212
 - Text Area, 189, 198
 - Text Input, 189, 195–197, 209–211, 233
 - width attribute, 109
 - width media feature, 74, 78–79
 - width property, 80
 - Windows Phone 7
 - jQuery Mobile, 24, 147
 - PhoneGap, 220
 - configuring Dreamweaver, 228–230
 - Windows version, no support for iOS, 26
 - WOFF (Web Open Font Format), 39–40
 - World Wide Web Consortium. *See* W3C
 - #wrapper style rule, 92, 100
- X – Z**
- Xcode
 - downloading, 221
 - testing Travel Notes app, 266–269
 - XHTML 1.0, 10
 - development of HTML, 10
 - DOCTYPE declaration
 - HTML 1.0 Strict, 31
 - replacing with HTML5 DOCTYPE declaration, 7
 - HTML5
 - compatibility, 7
 - converting to, 5, 33–36
 - XHTML 2.0, 10–11
 - converting to HTML5, 14
 - XML (Extensible Markup Language)
 - versus HTML, 10
 - XMLHttpRequest object, 12