Phil Dutson

COVERS
jQuery Mobile 1.0
with Tips for 1.1

Sams Teach Yourself

jQuery Mobile

in 24 Hours

SAMS

Praise for *Sams Teach Yourself jQuery Mobile in 24 Hours*

"Phil does a great job taking you through the mobile ecosystem and how jQuery Mobile makes it dead simple to break into it. Going from the fundamentals of web and mobile to advanced topics like video and themes, anyone looking to gain greater knowledge in mobile development will profit from this book."

—**Brett Child**, Software Consultant, Software Technology Group

"*Sams Teach Yourself jQuery Mobile in 24 Hours* by Phil Dutson is full of rock-solid real-world examples that can be easily built upon to create a functional, rich, custom, completely usable mobile website.

The book reads incredibly easy; you find that the learning comes almost effortlessly as you read and work through the tutorials. In addition to learning the elements you need to build your own website, you'll also learn how to extend and fill your mobile website with elements such as video and the creation and scanning of QR and Microsoft Tag codes. It even covers the introduction of jQuery Mobile into WordPress and the development of Android-based applications using jQuery Mobile and PhoneGap. I highly recommend a read if you're doing any type of mobile web development."

—**Drew Harvey**, Solution Architect, CrossView, Inc.

"This book is an excellent resource for any developer looking to integrate jQuery mobile into their next project. Phil covers the fundamentals of jQuery mobile while also providing best practices for mobile development."

—**Jim Hathaway**, Web Developer

"This book is an excellent read for beginners and web veterans alike. Phil Dutson does an excellent job of highlighting the jQuery Mobile framework's semantics and syntax while also providing an introduction to mobile web development best practices in general."

—**Greg Lavallee**, Software Engineer, The Washington Post Company

"Well-written, detail-oriented, and documented with plenty of hands-on examples makes *Sams Teach Yourself jQuery Mobile in 24 Hours* flow and easily comprehensible. This book is a must-have library addition for the software developer beginning down the mobile application development path."

—**Tamara Urry**, Sr. Software Engineer & Owner, JET Technical

"In my years of learning, training, and teaching programming, I have rarely come across an individual with Phil Dutson's ability to explain code. Whether you are a beginner, novice, or experienced programmer, *Sams Teach Yourself jQuery Mobile in 24 Hours* is written for you. Developers who want to take advantage of the vast mobile market will want to add this book to their arsenal."

—**Dale Wallentine**, Associate Dean, School of Technology Stevens-Henager College, Logan Utah Campus

Phil Dutson

Sams **Teach Yourself**

jQuery Mobile

in 24 Hours

SAMS 800 East 96th Street, Indianapolis, Indiana, 46240 USA

Sams Teach Yourself jQuery Mobile in 24 Hours

ISBN-13: 978-0-672-33594-5
ISBN-10: 0-672-33594-8

Library of Congress Cataloging-in-Publication Data:

Dutson, Phil, 1981-
Sams teach yourself jQuery mobile in 24 hours / Phil Dutson.
p. cm.
ISBN 978-0-672-33594-5 (pbk. : alk. paper)
1. JavaScript (Computer program language)–Programmed instruction. 2. Web site development--Programmed instruction. I. Title.
QA76.73.J38D88 2013
005.2'762–dc23

2012015341

Printed in the United States of America

First Printing July 2012

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Development Editor
Songlin Qiu

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Geneil Breeze

Indexer
Tim Wright

Proofreader
Kathy Ruiz

Technical Editors
Jim Hathaway
Greg Lavallee

Publishing Coordinator
Olivia Basegio

Multimedia Developer
Dan Scherf

Interior Designer
Gary Adair

Cover Designer
Anne Jones

Compositor
Nonie Ratcliff

Contents at a Glance

Part IV: Extending the Mobile Experience

Table of Contents

About the Author

Phil Dutson is the lead front-end developer for ICON Health and Fitness. He has worked on projects and solutions for NordicTrack, ProForm, Freemotion, Sears, Costco, Sam's Club, and others. He was an original team member of the iFit team that integrated Google Maps into personalized workout n and playback. Phil co-founded and currently manages The E-Com DevBlog, a development blog focused on web development and solutions. To learn more visit http://dev.tonic1394.com.

Dedication

To my patient and exceptionally loving family. Thank you for the support and encouragement.

Acknowledgments

A huge thanks to all of the wonderful people at Sams for working with me on this project. In no particular order I'd like to single out my project editors, Trina MacDonald and Laura Lewin, for making sure I hit or at least came close to my deadlines; Olivia Basegio for helping me get things turned in; my awesome development editor, Songlin Qiu, who not only made sure that I always spelled "lets" as "let's" but also made sure that everything flowed nicely together; and my outstandingly brilliant technical editors, Jim Hathaway and Greg Lavallee, who not only tested every bit of code I threw at them, but pointed out when more explanation was necessary for the reader to really understand the concepts presented. I would also like to thank the production team, who put this book into your hands: Lori Lyons, Geneil Breeze, Nonie Ratcliff, and Kathy Ruiz.

As a personal thanks, I'd like to tell Dave "Davidicus" Brown thank you for being indirectly and yet directly responsible for this endeavor. Another special thanks goes to my designer friends in UltraCube for always telling anyone who would listen (including me) to "believe in your dreams," as well as my current eCommerce crew (Tracy, Casey, Sid, Remo, Brett, Eric, Chris, and Kim) and my previous crew (Berticus, Drewbie, Matt, Branden, and Stretch) for the support, praise, slight criticism, and sanity checks.

We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: consumer@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

Introduction

There is little doubt that the way we currently access, use, and share the things we find online is going to continue to become more and more mobile. Every month thousands of new smartphones are activated and accompany their owners everywhere from trips to the grocery store to mountain hiking. As our thirst for connectivity expands into areas not possible a few years ago we need sites that deliver information quickly and easily, and that will work no matter what device we are u sing for access. This can be done with jQuery Mobile.

Built on the popular and stable jQuery framework, jQuery Mobile can be utilized to transform existing sites into mobile-friendly ones. If you do not have a site yet, don't worry because starting out with a blank slate is a great way to see how simple and easy using jQuery Mobile is. Shortly you will have a site that handles as well on a mobile device as it does on a desktop.

Getting into mobile site development is no longer an option; it's a necessity. Even though the landscape is morphing as much for the mobile web as for the current desktop websites, using a framework like jQuery Mobile helps bridge the gaps between hardware and software platforms. This gives you the peace of mind that users can still use your site, even if they are on a legacy device that may fail to support many other sites today.

Key Features of This Book

You're not only going to learn how to use jQuery Mobile, you're going to put it to work with scannable codes, videos, and CMS integration. Starting out with the basics we build a simple web page to display information about you. You'll then kick it up a notch and learn how to customize your page by including a responsive layout so that small screen devices are shown content made just for them while tablet users get an optimized experience that takes advantage of the increased screen space. You also learn how to detect mobile devices and route them to special directories or sections of your site based on the User Agent.

Mobile users also enjoy the use of rich site content, including videos that improve the user experience by providing a visual guide to a product or service. You learn about embedding videos as well as encoding video so that they play back on mobile devices as well as the pros and cons for each method.

If you are in marketing looking into the viability of a mobile project or endeavor, you'll enjoy learning about the use of scannable codes to help track and direct mobile users to specific sites, videos, or text messages. These codes can be printed in magazines, manuals, and on product packaging to help users learn more about your product or to sign up for updates.

As you progress hour by hour, you get a great foundation for mobile site development as well as gain valuable insight through the tricks, tips, and warnings scattered throughout. Twenty-four hours from now you'll have a greater understanding of mobile site development and what needs to be done to get your next project mobilized and into the hands of millions.

How to Use This Book

This book is straightforward. It starts here in the introduction, and then moves on into the first hour. There are 24 lessons, each of which should take about an hour to complete. I tried to set up the book not only as a lesson guide, but also as a handy reference guide that you can keep around once you are finished. As each hour covers a portion of the jQuery Mobile framework along with examples, this book should prove useful even if you use it only for reference. While it is possible to quaff an energy drink or two and have this entire book finished in one go, I'd honestly recommend you give your subconscious some time to process the concepts you learn in each hour.

How This Book Is Organized

This book has been carved into four parts to help you focus or reference the section that best suits your level of learning or interest. They are

- Part I, "Beginning jQuery Mobile," covers the basics, everything from HTML, CSS, and JavaScript to building a page using jQuery Mobile. If you are entirely new to the game, start right here, and you'll be up to speed in no time.
- Part II, "Creating the User Interface," teaches you about the user interface and how it is styled with jQuery Mobile. This is anything the user is going to see, touch, and use. This part even covers the use of events to help create custom functionality for gestures as well as adjusting the built-in theme manually.
- Part III, "Customizing Your Content," takes you beyond the jQuery Mobile defaults and looks at adding plug-ins, themes, responsive design, and device detection.
- Part IV, "Extending the Mobile Experience," is all about taking it to the next level. Video integration, device emulation, minified code, creating an Android app using PhoneGap, and even adding jQuery Mobile to a WordPress theme are all covered here.

- Q&A, Quiz, and Exercises

  At the end of each hour you find a section that contains some questions and answers for the topic covered during the hour. While some of these questions may offer a deeper explanation or insight into what was covered, others explain the reasons for covering some solutions while overlooking others. The Quiz section can be used to help you test what you learned in the hour while the Exercises help you put it all into practice.

Conventions Used in This Book

This book contains special elements as described here:

TIP

This Is a Tip

These are tips and tricks that you can use to help make your site or experience a little better.

NOTE

This Is Important Information

When you see these you get important information about a topic that was mentioned or covered.

CAUTION

This Is a Warning!

When you see these be careful; the information posted here usually saves you from doing something that might break your code, or lets you know about a result that would otherwise be unexpected.

This book uses a special `monospace` font on code/progamming-related terms/text such as `id="home"`

Code listings contain numbered lines for better understanding.

Sample Code for This Book

Throughout the book various files are referenced to help you learn jQuery Mobile as occasional starting points and for comparison with code you have written yourself. These files can be downloaded in a compressed format by visiting www.informit.com/title/9780672335945. Go to the Downloads tab and click on the "Sample Code" link.

NOTE

The Author's Websites

Anything web related moves fast. Sometimes there are small gaps of time between versions in a framework or browser, but when a community is driving new functionality and more and more devices are released into the market, things are quickly changed. The author of this book, Phil Dutson, maintains two websites that can be useful as you learn about jQuery Mobile and web development. The first is http://www.jquerymobilein24.com/ where you can find blog entries and updates to the world of jQuery Mobile. The second is the eCom DevBlog, which can be found by visiting http://dev.tonic1394.com/.

Each site hosts a blog with a searchable index to help you find what you are looking for, as well as a comments section that you can use to leave feedback or ask questions.

HOUR 4

Introduction to the jQuery Mobile Framework

What You'll Learn in This Hour:

- What files make up the jQuery Mobile framework
- How jQuery Mobile works with data attributes
- How to use jQuery Mobile to create a simple page
- How to use the mobile initialization event

The jQuery Mobile framework is the perfect place to get started with mobile development. Just like the standard jQuery framework, it is built to deliver speed, stability, and an excellent cross-browser experience to your visitors.

During this hour we start with adding jQuery Mobile to a web page and then discuss a little of how jQuery Mobile runs with the use of data roles. We then create a basic page with HTML and make it a mobile page using jQuery Mobile. Finally in this hour we cover the `pageinit` function and the difference between it and the standard jQuery `$(document).ready()` function.

Adding jQuery Mobile to Your Site

Adding the jQuery Mobile framework to your site is almost as easy as adding the standard jQuery framework to your site. In fact, jQuery Mobile requires the standard jQuery framework to function. In this respect, it could be considered part of the jQuery Mobile framework since it will not run without it.

Three files make up the complete jQuery Mobile framework:

- jQuery library JavaScript file
- jQuery Mobile library JavaScript file
- jQuery Mobile CSS style sheet

The two jQuery libraries are included as they contain all the logic that makes the frameworks work. The jQuery Mobile library extends on the base features of the jQuery library. Each of these libraries is available in a production version and a development version. The main difference between the two is that in the production version each file has been minified to remove excess whitespace and leaves out all comments. If you want to dive into what makes each framework tick, grab the development versions as they contain extra lines to help with legibility and comments to help explain some sections of code.

The jQuery Mobile CSS style sheet is included as it contains all the styles, themes, and swatches that are used with jQuery Mobile. These styles include various settings for backgrounds, colors, margin, padding, and sprites. jQuery Mobile does leverage CSS3 styles as this allows a smaller file size, and most modern mobile browsers have fairly good support for them.

When the files listed previously are included together in your HTML file, you are ready to get started using jQuery Mobile.

CAUTION

Including JavaScript in the `head` Element

Last hour we discussed putting all JavaScript includes just before the closing `body` tag and how this helps reduce HTTP request blocking and perceived loading speed. While that is a good practice on a standard HTML page, this will *not* work with all scripts when using jQuery Mobile. Because jQuery Mobile uses AJAX, scripts in the `head` element are considered to be "run once" scripts. If you have scripts that you want to execute only on a single or certain page you should either use the `pageinit` event, or you may include your scripts within the `div` element you are using as a page with the `data-role="page"` attribute.

Listing 4.1 shows the contents of basic_layout.html, which we use to start the layout of a simple mobile page including jQuery Mobile.

LISTING 4.1 Basic Page Layout Including jQuery Mobile

```
 1: <!DOCTYPE html>
 2: <html>
 3:   <head>
 4:     <title>Developing with jQuery Mobile</title>
 5:     <meta name="viewport" content="width=device-width, initial-scale=1">
 6:     <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
 7:     <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 8:     <script
src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"> </script>
 9:   </head>
10:   <body>
```

```
11:     <p>content will go here shortly</p>
12:   </body>
13: </html>
```

Let's take a walk through the basic structure. Beginning with line 1 we set up our `HTML5 DOCTYPE`. This allows most mobile browsers and even most modern desktop browsers to use all the features that jQuery Mobile offers. The `HTML5 DOCTYPE` also allows some backwards compatibility with older browsers.

The next few lines are all standard HTML, but line 5 contains a `meta` element with some attributes that you may not recognize. The `name` attribute tells the browser what type of data this `meta` element contains. In this case it tells the browser that it contains information for the `viewport` or the display size of the page. The `content` attribute is set to `width=device-width, initial-scale=1`. This means that when the page is viewed, it should not be zoomed in to a particular portion of the page or zoomed out and shrink everything down. Instead the page should be rendered with the same dimensions as the device it is being viewed on, averting the strange zoomed views that plague some websites that are viewed on mobile devices.

Line 6 is where we include the jQuery Mobile CSS file. This file contains all the default styles that will be applied when using jQuery Mobile, and it also contains some theme styles that can be overwritten.

Line 7 contains the include for the standard jQuery JavaScript library. Note that this include must be called before you call the jQuery Mobile JavaScript library or you will receive JavaScript errors on the page.

Line 8 is the include for the jQuery Mobile JavaScript library.

Lines 9 through 13 show basic HTML markup for the rest of the page.

You can try running the code from Listing 4.1 by running basic_layout.html in your browser. Don't be surprised or worried that the site doesn't look very mobile and that it only contains one line of text. This is normal and rendering exactly as expected. In fact it brings us to using data roles.

Using Data Roles

Before we dive into data roles, you need to first learn what they are. We know that HTML elements are limited in the attributes they can contain to be considered valid. For example an `img` element requires an `src` and an `alt` attribute to be valid. While there are some optional attributes that you can also put in the `img` element that would still allow it to be valid, if you were to throw in an attribute of `imgtitle` it would not validate. Listing 4.2 demonstrates this with two `img` elements; one is valid and the other is not.

LISTING 4.2 An Example of Valid and Invalid HTML Markup

```
1: <!-- The following img element is valid and will pass HTML validation -->
2: <img src="images/logo.png" alt="Site Logo" />
3: <!-- The following img element has been customized and will fail HTML
validation -->
4: <img src="images/logo.png" alt="Site Logo"
imgtitle="Tremble before our mighty logo of fury!" />
```

Let's walk through the code snippet in Listing 4.2. Line 1 is an HTML comment that I included to help you understand what is going on. Line 2 is a perfectly valid `img` element that contains both `src` and `alt` attributes. Line 3 is another HTML comment, and line 4 is an invalid `img` element because it contains a custom attribute with a custom value.

We know that the code in line 4 is invalid, but what happens when we need that extra text in that `img` element? For example, what if we had a tooltip function that was built in JavaScript that we wanted to use that would take the text from the custom `imgtitle` attribute and display it? This is why data attributes exist.

Data attributes are part of the HTML5 specification. They were created so that developers who need to use custom tags can use them without fear of breaking validation of the element they are customizing. To create a data attribute you must start the attribute with `data-` you may then use any letters you want as long as they are lowercase.

NOTE

In Reference to a data attribute

Due to the way data attributes are created some people refer to them as `data-*`. The asterisk is used to denote that after typing `data-` almost anything can be used. Creating data attributes named `data-mine`, `data-billingaddress`, `data-phone`, `data-userid`, `data-a`, and `data-z` are all perfectly acceptable. However, actually using a data attribute named `data-*` in your code, is not.

Experienced jQuery users may already be familiar with data attributes and the use of the jQuery `.data()` function when working with them. If you decide to reference data attributes with this function, be aware that after the first dash any extra text will be formatted in CamelCase. For example, `data-my-attribute` will become `data-myAttribute`. To get around this you can use the jQuery `.attr()` method instead to specify your custom attribute. For more information on the `.attr()` method, visit http://api.jquery.com/attr/.

Using a `data-attribute` and an `HTML5 DOCTYPE` informs the browser to ignore the attribute and any data contained in it, thereby allowing it to validate.

jQuery Mobile uses these data attributes to create data roles for data storage. This allows the jQuery Mobile library to find the data contained in custom attributes and manipulate them without making any of the code invalid.

Creating a Simple Page

We have already covered the code it takes to build the basic structure for a mobile site, but we have not actually made it look like a mobile site. Let's expand that into a one-page and one-button site so that we can get a better feel for how jQuery Mobile works.

In Listing 4.3 we add several elements to the page and use data attributes to apply functionality and style.

LISTING 4.3 Expanding the Mobile Site to Include a Header and a Button

```
 1: <!DOCTYPE html>
 2: <html>
 3:   <head>
 4:     <title>Developing with jQuery Mobile</title>
 5:     <meta name="viewport" content="width=device-width, initial-scale=1">
 6:     <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
 7:     <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 8:     <script
src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"> </script>
 9:   </head>
10:   <body>
11:     <div data-role="page">
12:       <div data-role="header"><h1>Single Page Site</h1></div>
13:       <div data-role="content">
14:         <p>Look at the button!</p>
15:         <a href="#" data-role="button">I am a button</a>
16:       </div>
17:     </div>
18:   </body>
19: </html>
```

The preceding code was based on the code from Listing 4.1 and can be viewed by opening the file expanded_layout.html. Let's go over the changes that were made from the original code.

Inside the `body` element we have added a new `div` element on line 11. This `div` has an attribute of `data-role="page"`, which allows jQuery Mobile to treat this `div` as a single page. This functionality also allows you to have a multipage site contained in one HTML file. Continuing

on to line 12 you can see that it contains a `div` with an attribute of `data-role="header"`. This tells jQuery Mobile to treat this as a container for the `header` section of the page. Inside this `div` element is an `h1` element. It is important that we have this element in the `header`, not only for Search Engine Optimization (SEO) purposes, but because it gets a specific style applied to it that helps complete the look of the header section.

Line 13 contains another `div` with an attribute of `data-role="content"`. This applies some padding and other styles and designates this `div` as the container for the `content` section of the page. Line 14 shows a `p` element with some text inside it. This is included as an example of what text looks like inside the content section. Also inside the `content` section line 15 shows a link that we have set up. While this link currently does not go anywhere (the `href` attribute is set to #), it is important to notice that the `a` element contains the `data-role="button"` attribute. jQuery Mobile uses this attribute to transform the standard link into a fully styled button.

If you were to render the page in a browser, the whole page should now look more like a mobile site. The default theme is applied including a clearly defined `header` section and `content` section. The button that we added is in the `content` section and automatically expands to fit the size of the screen it is being viewed on.

I mentioned previously that the button that was added does not actually do anything when clicked. This is fine for personal amusement and practical jokes, but we can make it do something if we set the `href` attribute to link somewhere. In Listing 4.4, which shows the contents of button_click.html, you can see that the button has been set to point to another page, and a second page section has been set up within our HTML file.

LISTING 4.4 The Mobile Site with a Working Button and Second Page Section

```
 1: <!DOCTYPE html>
 2: <html>
 3:   <head>
 4:     <title>Developing with jQuery Mobile</title>
 5:     <meta name="viewport" content="width=device-width, initial-scale=1">
 6:     <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
 7:     <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 8:     <script
src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"> </script>
 9:     </head>
10:   <body>
11:     <div data-role="page">
12:       <div data-role="header"><h1>Single Page Site</h1></div>
13:       <div data-role="content">
```

```
14:         <p>Look at the button!</p>
15:         <a href="#dpop" data-role="button" data-rel="dialog">I am a button</a>
16:       </div>
17:     </div>
18:     <div data-role="page" id="dpop" data-theme="d">
19:       <div data-role="header"><h1>Clicked!</h1></div>
20:       <div data-role="content">
21:         <p>clicked content!</p>
22:         <a href="#" data-rel="back" data-role="button">Go back</a>
23:       </div>
24:     </div>
25:     </body>
26: </html>
```

The first change is on line 15. We changed the link to now point at a populated anchor instead of an empty one. We also added another attribute to the `a` element. Using `data-rel="dialog"` allows jQuery Mobile to display the linked element as a `dialog` page instead of a standard page. Using a `dialog` gives a different style and feel than a standard page and also shows the page with a special `pop` transition by default.

If you look farther down in the code for the referenced anchor you will find it on line 18. This is where we set up a second `page`. We gave the second `page` an `id` of `dpop`, and just like the `page` on line 11, it is a `div` element. Something else we have done is added `data-theme="d"`, which styles the dialog window a little differently than the current color scheme.

There are currently five basic swatches in the jQuery Mobile default theme. The swatches are selected by passing either a, b, c, d, or e into the `data-theme` attribute.

Line 19 shows that we set up another `header` section and added another `h1`. Line 20 is the same as line 13, which is a `div` with an attribute of `data-role="content"`, and just like line 13 it tells jQuery Mobile that this `div` element will be used as a `content` section. Within this section we created a back button on line 22. The back button works through the use of the `data-rel="back"`. This attribute sends the browser back one step in history. This is why the `href` has an empty anchor value.

Figure 4.1 shows the first page, and Figure 4.2 shows the dialog page.

As you can see from the preceding coding exercise, jQuery Mobile allows you to easily make mobile sites with very little code. You can also create your own events, or even tweak the default ones. To get started adding your own customization we need cover some subtle differences between the standard jQuery framework and the jQuery Mobile framework.

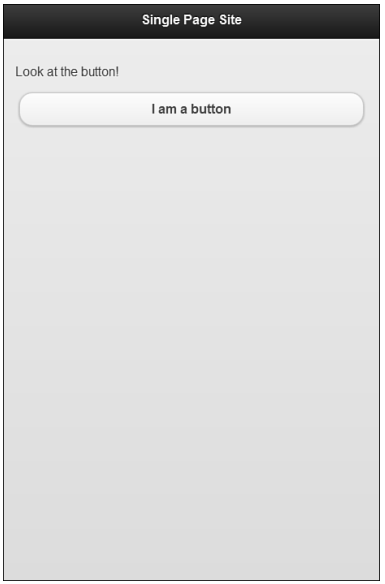

FIGURE 4.1
A simple page with one button linking to the dialog page

FIGURE 4.2
The dialog page that appears when linked from the first page

Understanding the Mobile Initialization Event

You already learned that one of the problems with using standard JavaScript happens when including scripts and attaching events or even when trying to manipulate data that may not exist at the time the function is called. The jQuery framework uses the `$(document).ready()` function to circumvent manipulation and loading problems by giving you access to your functions as soon as possible. While this is fantastic for single page sites, it becomes a small problem for the jQuery Mobile framework.

jQuery Mobile uses AJAX to load the contents of each page rather than reload the entire DOM structure. The `$(document).ready()` function only runs once per page load, not per AJAX call. In jQuery Mobile, the `$(document).ready()` function doesn't run once per page, but rather once per site unless a page refresh is requested or performed by the user. This means that some of the default settings that need to be set by jQuery Mobile cannot be set in the `$(document).ready()` function because they would not be applied to pages included through AJAX.

The answer to setting and changing these defaults is to use the `mobileinit` event because it runs before the `$(document).ready()` function ever does. To use the `mobileinit` event you must first include the jQuery framework and then either inline or include an external JavaScript file that contains an event binding for the `mobileinit` event and finally the include for jQuery Mobile. That may sound a little confusing, so let's look at Listing 4.5 for an example of this process.

LISTING 4.5 Including jQuery, an Inline `mobileinit` Script, and jQuery Mobile

```
1: <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
2: <script type="text/javascript">
3:   $(document).on("mobileinit", function() {
4:     $.extend(  $.mobile , {
5:       pageLoadErrorMessage:
'Either the page cannot be found or it cannot be loaded.'
6:     });
7:   });
8: </script>
9: <script
src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"></script>
```

Line 1 starts out with the include for the jQuery framework. Line 2 is the beginning of our inline JavaScript for setting the `mobileinit` event. Line 3 is where we actually set up the `mobileinit` event by using the `.on()` function (which is part of the version 1.7 of the jQuery framework and is why we needed to include the jQuery framework before our inline script). If we had been using a version of jQuery Mobile prior to 1.1 we would be using the `.bind()` function instead of the `.on()` function. The `.on()` function can take quite a few arguments, but in our case we pass two. The first argument passed is the event you want to bind, and the second is usually an

anonymous function that contains the code you want to run when the event runs. You can learn more about the `.on()` function by visiting http://api.jquery.com/on/. On line 4 we used the `$.extend()` function, which allows us to merge two objects together, and passed `$.mobile` to be used as the target object that we want to add or merge to. We then used an opening brace to begin the array of settings that we want to merge or change into the `$.mobile` object. Continuing to line 5 you can see that we are going to overwrite the default for `pageLoadErrorMessage` by setting the value to `"Either the page cannot be found or it cannot be loaded"`. The notation used here may look a familiar. It is a name-value pair and is commonly known as JavaScript Object Notation (JSON). It is commonly used in jQuery plug-ins, configurations, and functions (including the jQuery `.css()` function). Line 6 shows the closing brace and closing parentheses of the `$.extend()` function. Line 7 is the closing brace and closing parentheses of the `.on()` function. Line 8 is the closing tag for our inline script. Line 9 then includes the jQuery Mobile framework that now has the default value for `pageLoadErrorMessage` changed.

Now that we know that we can change some of the default settings of jQuery Mobile with the `mobileinit` event what do we do when we want an event or function to be triggered when a new page is loaded? I'm fairly sure that you are thinking, "That's easy! Just use the `$(document).ready()` function on the page you are loading and you'll be all set!" While that is normally the correct answer, we have to remember that all pages are inserted into the DOM through AJAX. This means that the DOM is only loaded once, making the `$(document).ready()` function load on the first page only. Luckily jQuery Mobile has a solution for this problem: You just need to use the `pageinit` event.

Using the `pageinit` Event Instead of `$(document).ready()`

To use the `pageinit` event on your page, you have to take a slightly less dynamic and more planned approach to your code. There are a few different ways you can attach the `pageinit` event in your code. When using a version of jQuery Mobile prior to 1.1, you will be using jQuery 1.6.4, which means you use the `.bind()` function instead of the `.on()` function. When using jQuery Mobile 1.1+ you use the `.on()` function to bind the event.

The `.on()` function introduced in jQuery 1.7 is a unification of previous functions used to bind events. Instead of having to worry about using `.bind()`, `.live()`, or `.delegate()`, you can now use the `.on()` function to find events. More about this function can be found by visiting http://api.jquery.com/on/. If you are using a version of jQuery Mobile prior to 1.1, you should not use the `.on()` method, but should instead use the `.delegate()` or `.live()` function.

In Listing 4.6 the contents of multipage_one.html, which is the first page of a multipage site are shown. It includes a button that then loads a second page through AJAX.

LISTING 4.6 A Page Containing the Setup for the `pageinit` Event

```
 1: <!DOCTYPE html>
 2: <html>
 3:   <head>
 4:     <title>Developing with jQuery Mobile</title>
 5:     <meta name="viewport" content="width=device-width, initial-scale=1">
 6:     <link rel="stylesheet"
 href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
 7:     <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 8:     <script type="text/javascript">
 9:       $(document).on("mobileinit", function() {
10:         $.extend(  $.mobile , {
11:           pageLoadErrorMessage: 'Either the page cannot be found or it
cannot be loaded.'
12:         });
13:       });
14:       $(document).on("pageinit","#pageinit2", function() {
15:         alert("pageinit is bound!");
16:       });
17:     </script>
18:     <script
src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"></script>
19:   </head>
20:   <body>
21:     <div data-role="page">
22:       <div data-role="header"><h1>pageinit event example</h1></div>
23:       <div data-role="content">
24:         <p>The button below will use AJAX to load another page and trigger
 a bound event</p>
25:         <a href="multipage_two.html" data-role="button">Click to open a new
 page</a>
26:       </div>
27:     </div>
28:   </body>
29: </html>
```

Looking at the preceding code, do you see the `.on()` function and the `pageinit` event? Let's walk through the code and I will explain what we have done and how it makes things work.

The beginning lines should seem familiar by now, so we'll start at line 9. Line 9 uses `$(document)` as the selector, and then uses the `.on()` function to bind the `mobileinit` event to the current page. In comparison line 14 also uses the `.on()` function to bind the `pageinit` event. The binding takes place through the second parameter passed in the `.on()` function. The `.on()` function allows events to be delegated to elements that do not currently exist in the DOM, but that will exist in the future. This is why the `pageinit` event will be bound to the object that will have an `id` of `pageinit2` once it is added to the DOM.

Continuing onto line 15 we can see that an `alert()` function is going to be called as soon as the page is initialized into the DOM. Line 16 then closes the `.on()` function.

Continuing onto line 18 we see the include for jQuery Mobile. Line 19 closes the `head` element. Line 20 starts the `body` element. Lines 21 through 27 make up the actual code that will be presented as a page to the user. Line 25 shows the setup for a button, and if you look closely you see that our button is set to link to another file instead of an anchor tag farther down the page. We cover multipage sites in Hour 7, "Learning About Page Layout."

To finish out the file, lines 28 and 29 are closing tags for the `body` and `html` elements that complete our page.

We set up a function inside the `pageinit` event that triggers anytime a page with an `id` of `pageinit2` is loaded. Even though we are binding the `pageinit` event on the first page, it will not run on there because it does not have an `id` of `pageinit2`.

Now let's look at Listing 4.7, which shows the contents of multipage_two.html and is the second page that Listing 4.6 links to.

LISTING 4.7 This Page Has an Event Tied to It That Will Trigger on Page Load

```
 1: <!DOCTYPE html>
 2: <html>
 3:   <head>
 4:     <title>Developing with jQuery Mobile</title>
 5:     <meta name="viewport" content="width=device-width, initial-scale=1">
 6:     <link rel="stylesheet"
 href="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-rc.1.min.css"
 />
 7:     <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
 8:     <script
 src="http://code.jquery.com/mobile/1.1.0-rc.1/jquery.mobile-1.1.0-rc.1.min.js">
 </script>
 9:   </head>
10:   <body>
11:     <div data-role="page" id="pageinit2">
12:       <div data-role="header"><h1>pageinit event example </h1></div>
13:       <div data-role="content">
14:         <p>Fantastic! I am a new page and was loaded through AJAX.</p>
15:         <a href="pageinit.html" data-role="button" data-rel="back">
Amazing, now take me back</a>
16:       </div>
17:     </div>
18:   </body>
19: </html>
```

Look closely at the code for a minute and see if you can spot anything that would make it trigger an event when it loads.

A quick glance over lines 1 through 9 shows the typical setup of a mobile site. We can see the HTML5 DOCTYPE being used on line 1. The head element starts on line 3 and includes a title element set on line 4, the meta element set for mobile devices on line 5, the include for the jQuery Mobile style sheet on line 6, the include for the jQuery library on line 7, the include jQuery Mobile library on line 8, and the closing tag for the head element on line 9.

The only thing that appears different on this page from the page in Listing 4.6 is the lack of binding for both the mobileinit event and the pageinit event. Let's keep looking down the code and see if we can see anything else.

Skipping over line 10, line 11 shows us setting up a div element with a data-role="page" attribute and an id="pageinit2". That is the line that contains the attribute that ties this whole thing together. The attribute is id="pageinit2". That is the id that we used for the .on() function to bind the pageinit event to. Since this page contains that particular id, as soon as it is loaded into the DOM it triggers the pageinit event and calls the alert function that we placed inside the event.

Continuing down the rest of the page, you can see the content section being created on line 13. We also added a button that allows us to link back to the first page being created on line 15. This button contains both a link back to the first page through the attribute href="pageinit.html" and an attribute of data-rel="back" that defaults to a sliding-back page transition to the first page. Lines 16 through 19 are closing tags for various HTML elements that complete our page.

Perceived pageinit Processing Speed

When using the pageinit event, the transition between pages does not actually take place until whatever code has been placed into the pageinit event has finished processing. This can cause some delay between page transitions and may make users who are unprepared for the wait think the site has some lag or is slow. Plan ahead and keep the functions that you need to have on page-load short and direct.

The following figures show the use of the pageinit event. These figures were taken on an Android device running version 4.0.1. Depending on what browser and device you use, the alert message will appear styled differently. Figure 4.3 shows the page we built in Listing 4.6. Figure 4.4 shows the alert that is triggered by the function called in the pageinit event. Figure 4.5 shows the page that triggered the pageinit event and is the code from Listing 4.7.

You can set up more pageinit events for other pages by using a unique id for each page you want like to attach the event to. By doing this you can trigger functions that you only want happening when certain pages load.

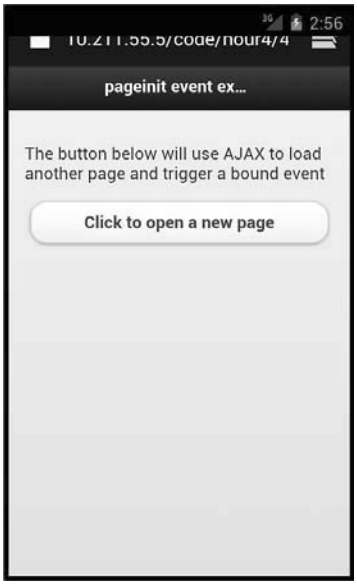

FIGURE 4.3
When the button is pressed, the `pageinit` event will be called.

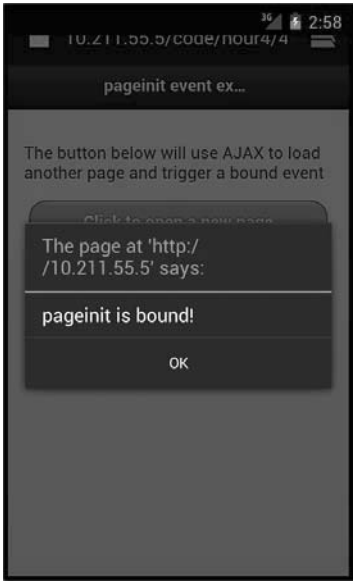

FIGURE 4.4
This alert message was triggered by the `pageinit` event.

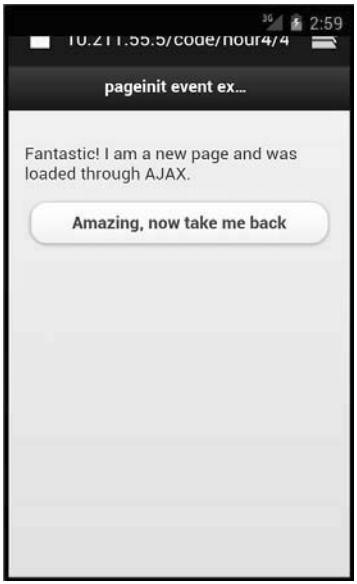

FIGURE 4.5
This is the second page. It appears only after the code in the `pageinit` event has completed.

Summary

This hour discussed the jQuery Mobile framework. You learned that it is dependent on the standard jQuery framework to function. You learned what files are included and required to use the jQuery Mobile framework, and also learned about HTML5 data attribute and how jQuery Mobile leverages them to apply style and functionality to the page.

The basic requirements needed for a mobile page were discussed as well as how to add some extra HTML elements into the page to complete the look and feel of a mobile site.

Finally you learned how to overwrite some of the default jQuery Mobile settings and that you can use the `pageinit` event to run a function when a page is called to get around the problems associated with the `$(document).ready` function only running on the first page load.

Q&A

Q. Can I use my own data attributes within jQuery Mobile or will that break the site?

A. Generally you are free to use whatever data attribute you want, but quite a few data attributes are reserved by jQuery Mobile. For a short list stay away from the following: `data-theme`, `data-ajax`, `data-filter`, `data-icon`, `data-grid`, `data-rel`, `data-icon`, `data-url`, `data-role`, and `data-type`. A good practice is to add a prefix that is unique to you or your site so you would end up with a custom attribute that looked similar to `data-mcn-username`, where the prefix "mcn" stands for "my custom name." Just as a reminder, if you use a dash in your data-attribute, you should avoid using the jQuery `.data()` function and instead use the `.attr()` function.

Q. Regarding the `pageinit` event, can I bind the event on the page I want it to run on instead of on the first page using the `.on()` function?

A. You can, but keep a few things in mind. When running jQuery Mobile 1.1+, if you use the `.on()` function on the page you want to have the `pageinit` event work on, then the first time that page is loaded, your `pageinit` function will run once. The second time that page is loaded, the function will run twice instead of once. This is because the event is being bound over and over again instead of just once. You can get around this with the `.off()` function. If you forget to unbind the `pageinit` event you run the risk of crashing the browser on the mobile device that is viewing your site due to the amount of memory that will be taken up in binding and running your function over and over again. See http://api.jquery.com/off/ for information about unbinding events set with the `.on()` function.

Q. You warned me to watch out when adding scripts outside the `head` element. Can you explain why the scripts are "run once"?

A. When you are browsing a mobile site that uses jQuery Mobile, any scripts included in the `head` element will be read only on the page that you start from. So you start on the home page and then the scripts that are in the `head` element of that page will be run. If you use a button or a navigation bar to move to another page, the scripts in the `head` element on those pages will be ignored because only the `title` element and the `div` element that contains the `data-role="page"` attribute will be parsed and brought into the DOM.

Workshop

The workshop contains a quiz and some exercises to help you check your comprehension and understanding.

Quiz

1. What files make up the jQuery Mobile framework?
2. True or False: The `$(document).ready()` function cannot be used on pages that are added to the DOM through AJAX.
3. How is the `.on()` function different from the `.bind()` function?
4. True or False: You can use the `.on()` function with jQuery 1.6.4.
5. True or False: The use of data attributes in HTML has been around since HTML 3.2.

Answers

1. The jQuery Mobile framework is made up of a JavaScript file and a CSS style sheet. However, to function, the jQuery framework must be included.
2. True. The document is only loaded once per page and will not run on subsequent pages that are inserted into the DOM through AJAX.
3. The main difference is that you can use the `.on()` function to bind events to objects that do not exist on initial page load. This is extremely helpful with binding events to elements that appear through AJAX.
4. False. The `.on()` function was not added to the jQuery framework until version 1.7. Prior to that you could use `.delegate()` or the `.bind()` functions for event binding.
5. False. The use of data attributes is new to HTML5 and has been added to the specification specifically for easier data manipulation and storage.

Exercises

1. Create your own `data-attribute` and add it to a page. Try using some jQuery scripts to get data from it or as a way to help create a selector for the element.
2. Take some of the code from the chapter and add more links and more pages. Experiment with setting up multiple `pageinit` events for your new pages.
3. Try using the `.on()` function with another event and with an object that is on a separate page. Try binding different events with it. Something as simple as a `click` event that triggers an `alert()` function is a good way to get started with understanding how the `.on()` function works with events.

This page intentionally left blank

Index

Symbols & Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U–V

W

X–Y–Z