"The Missing Manual series is simply the most intelligent and usable series of guidebooks..."
—KEVIN KELLY, CO-FOUNDER OF *WIRED*

JavaScript & jQuery

the missing manual®

The book that should have been in the box®

Second Edition

O'REILLY®

David Sawyer McFarland

WEB AUTHORING AND DESIGN

Answers found here!

JavaScript lets you supercharge your HTML with animation, interactivity, and visual effects—but many web designers find the language hard to learn. This jargon-free guide covers JavaScript basics and shows you how to save time and effort with the jQuery library of prewritten JavaScript code. You'll soon be building web pages that feel and act like desktop programs, without a lot of programming.

the missing manual

The book that should have been in the box®

The important stuff you need to know

- **Make your pages interactive.** Create JavaScript events that react to visitor actions.
- **Use animations and effects.** Build drop-down navigation menus, pop-ups, automated slideshows, and more.
- **Improve your user interface.** Learn how the pros make websites fun and easy to use.
- **Collect data with web forms.** Create easy-to-use forms that ensure more accurate visitor responses.
- **Add a dash of Ajax.** Enable your web pages to communicate with a web server without a page reload.
- **Practice with living examples.** Get step-by-step tutorials for web projects you can build yourself.

David Sawyer McFarland president of Sawyer McFarland Media, Inc., has spent the last 15 years building and managing websites for Macworld.com and UC Berkeley, among other clients. Also a trainer, David has written books including Missing Manuals on Dreamweaver CS5, and JavaScript.

US $39.99 CAN $41.99
ISBN: 978-1-449-39902-3

O'REILLY®
missingmanuals.com
twitter: @missingmanuals
facebook.com/MissingManuals

JavaScript & jQuery

the missing manual®

The book that should have been in the box®

JavaScript & jQuery

2nd Edition

the missing manual®

The book that should have been in the box®

David Sawyer McFarland

O'REILLY®

Beijing | Cambridge | Farnham | Köln | Sebastopol | Tokyo

JavaScript & jQuery: The Missing Manual, Second Edition
by David Sawyer McFarland

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles: *safari.oreilly.com*. For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Printing History:

July 2008:	First Edition.
October 2011:	Second Edition.

Nutshell Handbook, the Nutshell Handbook logo, the O'Reilly logo, and "The book that should have been in the box" are registered trademarks of O'Reilly Media, Inc. *JavaScript & jQuery: The Missing Manual*, The Missing Manual logo, Pogue Press, and the Pogue Press logo are trademarks of O'Reilly Media, Inc.

ISBN: 978-1-449-3-9902-3

[M]

Table of Contents

Part Three: Building Web Page Features

Part Four: Ajax: Communication with the Web Server

Part Five: Tips, Tricks, and Troubleshooting

The Missing Credits

About the Author

David Sawyer McFarland is president of Sawyer McFarland Media, Inc., a web development and training company in Portland, Oregon. He's been building websites since 1995, when he designed his first site—an online magazine for communication professionals. He's served as webmaster at the University of California at Berkeley and the Berkeley Multimedia Research Center, and oversaw a complete CSS-driven redesign of Macworld.com.

In addition to building websites, David is also a writer, trainer, and instructor. He's taught web design at UC Berkeley Graduate School of Journalism, the Center for Electronic Art, the Academy of Art College, Ex'Pressions Center for New Media, and Portland State University. He's written articles about the web for *Practical Web Design*, *MX Developer's Journal*, *Macworld* magazine, and CreativePro.com.

He welcomes feedback about this book by email: *missing@sawmac.com*. (If you're seeking technical help, however, please refer to the sources listed in Appendix A.)

About the Creative Team

Nan Barber (editor) has worked with the Missing Manual series since its inception—long enough to remember HyperCard stacks.

Holly Bauer (production editor) lives in Ye Olde Cambridge, MA. She's a production editor by day and an avid home cook, prolific DIYer, and mid-century modern design enthusiast by evening/weekend. Email: *holly@oreilly.com*.

Carla Spoon (proofreader) is a freelance writer and copy editor. An avid runner, she works and feeds her tech gadget addiction from her home office in the shadow of Mount Rainier. Email: *carla_spoon@comcast.net*.

Angela Howard (indexer) has been indexing for more than 10 years, mostly for computer books, but occasionally for books on other topics, such as travel, alternative medicine, and leopard geckos. She lives in California with her husband, daughter, and two cats.

Acknowledgements

Many thanks to all those who helped with this book, including Shelley Powers and Steve Suehring, whose watchful eyes saved me from potentially embarrassing mistakes. Thanks also to my many students at Portland State University who have sat through my long JavaScript lectures and struggled through my programming assignments—especially the members of Team Futzbit (Combination Pizza Hut and Taco Bell) for testing the tutorials: Julia Hall, Amber Brucker, Kevin Brown, Josh Elliott, Tracy O'Connor, and Blake Womack. Also, we all owe a big debt of gratitude to John Resig and the jQuery team for creating the best tool yet for making JavaScript fun.

Finally, thanks to David Pogue for getting me started; Nan Barber for making my writing sharper and clearer; my wife, Scholle, for putting up with an author's crankiness; and thanks to my kids, Graham and Kate, because they're just awesome.

The Missing Manual Series

Missing Manuals are witty, superbly written guides to computer products that don't come with printed manuals (which is just about all of them). Each book features a handcrafted index and cross-references to specific page numbers (not just "see Chapter 14").

Recent and upcoming titles include:

- *Access 2010: The Missing Manual* by Matthew MacDonald
- *Buying a Home: The Missing Manual* by Nancy Conner
- *CSS: The Missing Manual*, Second Edition, by David Sawyer McFarland
- *Creating a Website: The Missing Manual*, Third Edition, by Matthew MacDonald
- *David Pogue's Digital Photography: The Missing Manual* by David Pogue
- *Dreamweaver CS5.5: The Missing Manual* by David Sawyer McFarland
- *Droid X2: The Missing Manual* by Preston Gralla
- *Droid 2: The Missing Manual* by Preston Gralla
- *Excel 2010: The Missing Manual* by Matthew MacDonald

- *Facebook: The Missing Manual*, Third Edition, by E.A. Vander Veer
- *FileMaker Pro 11: The Missing Manual* by Susan Prosser and Stuart Gripman
- *Flash CS5.5: The Missing Manual* by Chris Grover
- *Galaxy Tab: The Missing Manual* by Preston Gralla
- *Google Apps: The Missing Manual* by Nancy Conner
- *Google SketchUp: The Missing Manual* by Chris Grover
- *The Internet: The Missing Manual* by David Pogue and J.D. Biersdorfer
- *iMovie '11 & iDVD: The Missing Manual* by David Pogue and Aaron Miller
- *iPad 2: The Missing Manual* by J.D. Biersdorfer
- *iPhone: The Missing Manual*, Fourth Edition, by David Pogue
- *iPhone App Development: The Missing Manual* by Craig Hockenberry
- *iPhoto '11: The Missing Manual* by David Pogue and Lesa Snider
- *iPod: The Missing Manual*, Ninth Edition, by J.D. Biersdorfer and David Pogue
- *Living Green: The Missing Manual* by Nancy Conner
- *Mac OS X Snow Leopard: The Missing Manual* by David Pogue
- *Mac OS X Lion: The Missing Manual* by David Pogue
- *Microsoft Project 2010: The Missing Manual* by Bonnie Biafore
- *Motorola Xoom: The Missing Manual* by Preston Gralla
- *Netbooks: The Missing Manual* by J.D. Biersdorfer
- *Office 2010: The Missing Manual* by Nancy Connor, Chris Grover, and Matthew MacDonald
- *Office 2011 for Macintosh: The Missing Manual* by Chris Grover
- *Palm Pre: The Missing Manual* by Ed Baig
- *Personal Investing: The Missing Manual* by Bonnie Biafore
- *Photoshop CS5.5: The Missing Manual* by Lesa Snider
- *Photoshop Elements 10: The Missing Manual* by Barbara Brundage
- *PowerPoint 2007: The Missing Manual* by E.A. Vander Veer
- *Premiere Elements 8: The Missing Manual* by Chris Grover
- *QuickBase: The Missing Manual* by Nancy Conner
- *QuickBooks 2011: The Missing Manual* by Bonnie Biafore
- *QuickBooks 2012: The Missing Manual* by Bonnie Biafore

- *Switching to the Mac: The Missing Manual,* Snow Leopard Edition, by David Pogue
- *Switching to the Mac: The Missing Manual,* Lion Edition, by David Pogue
- *Wikipedia: The Missing Manual* by John Broughton
- *Windows Vista: The Missing Manual* by David Pogue
- *Windows 7: The Missing Manual* by David Pogue
- *Word 2007: The Missing Manual* by Chris Grover
- *Your Body: The Missing Manual* by Matthew MacDonald
- *Your Brain: The Missing Manual* by Matthew MacDonald
- *Your Money: The Missing Manual* by J. D. Roth

Introduction

The Web was a pretty boring place in its early days. Web pages were constructed from plain old HTML, so they could display information, and that was about all. Folks would click a link and then wait for a new web page to load. That was about as interactive as it got.

These days, most websites are almost as responsive as the programs on a desktop computer, reacting immediately to every mouse click. And it's all thanks to the subjects of this book—JavaScript and its sidekick, jQuery.

What Is JavaScript?

JavaScript is a programming language that lets you supercharge your HTML with animation, interactivity, and dynamic visual effects.

JavaScript can make web pages more useful by supplying immediate feedback. For example, a JavaScript-powered shopping cart page can instantly display a total cost, with tax and shipping, the moment a visitor selects a product to buy. JavaScript can produce an error message immediately after someone attempts to submit a web form that's missing necessary information.

JavaScript also lets you create fun, dynamic, and interactive interfaces. For example, with JavaScript, you can transform a static page of thumbnail images into an animated slideshow (as you'll learn how to do on page 314). Or you can do something more subtle like stuff more information on a page without making it seem crowded by organizing content into bite-size panels that visitors can access with a simple click of the mouse (page 301). Or add something useful and attractive, like pop-up tooltips that provide supplemental information for items on your web page (page 326).

Another one of JavaScript's main selling points is its immediacy. It lets web pages respond instantly to actions like clicking a link, filling out a form, or merely moving the mouse around the screen. JavaScript doesn't suffer from the frustrating delay associated with server-side programming languages like PHP, which rely on communication between the web browser and the web server. Because it doesn't rely on constantly loading and reloading web pages, JavaScript lets you create web pages that feel and act more like desktop programs than web pages.

If you've visited Google Maps (*http://maps.google.com*), you've seen JavaScript in action. Google Maps lets you view a map of your town (or pretty much anywhere else for that matter), zoom in to get a detailed view of streets and bus stops, or zoom out to get a birds-eye view of how to get across town, the state, or the nation. While there were plenty of map sites before Google, they always required reloading multiple web pages (usually a slow process) to get to the information you wanted. Google Maps, on the other hand, works without page refreshes—it responds immediately to your choices.

The programs you create with JavaScript can range from the really simple (like popping up a new browser window with a web page in it) to full-blown web applications like Google Docs (*http://docs.google.com*), which let you create presentations, edit documents, and create spreadsheets using your web browser with the feel of a program running directly on your computer.

A Bit of History

Invented by Netscape back in 1995, JavaScript is nearly as old as the web itself. While JavaScript is well respected today, it has a somewhat checkered past. It used to be considered a hobbyist's programming language, used for adding less-than-useful effects such as messages that scroll across the bottom of a web browser's status bar like a stock-ticker, or animated butterflies following mouse movements around the page. In the early days of JavaScript, it was easy to find thousands of free JavaScript programs (also called *scripts*) online, but many of those scripts didn't work in all web browsers, and at times even crashed browsers.

Note: JavaScript has nothing to do with the Java programming language. JavaScript was originally named LiveScript, but the marketing folks at Netscape decided they'd get more publicity if they tried to associate the language with the then-hot Java. Don't make the mistake of confusing the two…especially at a job interview!

In the early days, JavaScript also suffered from incompatibilities between the two prominent browsers, Netscape Navigator and Internet Explorer. Because Netscape and Microsoft tried to outdo each other's browsers by adding newer and (ostensibly) better features, the two browsers often acted in very different ways, making it difficult to create JavaScript programs that worked well in both.

Note: After Netscape introduced JavaScript, Microsoft introduced jScript, their own version of JavaScript included with Internet Explorer.

Fortunately the worst of those days is nearly gone and contemporary browsers like Firefox, Safari, Chrome, Opera, and Internet Explorer 9 have standardized much of the way they handle JavaScript, making it easier to write JavaScript programs that work for most everyone. (There are still a few incompatibilities among current web browsers, so you'll need to learn a few tricks for dealing with cross-browser problems. You'll learn how to overcome browser incompatibilities in this book.)

In the past several years, JavaScript has undergone a rebirth, fueled by high-profile websites like Google, Yahoo, and Flickr, which use JavaScript extensively to create interactive web applications. There's never been a better time to learn JavaScript. With the wealth of knowledge and the quality of scripts being written, you can add sophisticated interaction to your website—even if you're a beginner.

Note: JavaScript is also known by the name ECMAScript. ECMAScript is the "official" JavaScript specification, which is developed and maintained by an international standards organization called Ecma International: *http://www.ecmascript.org/*

JavaScript Is Everywhere

JavaScript isn't just for web pages, either. It's proven to be such a useful programming language that if you learn JavaScript you can create Yahoo Widgets and Apple's Dashboard Widgets, write programs for the iPhone, and tap into the scriptable features of many Adobe programs like Acrobat, Photoshop, Illustrator, and Dreamweaver. In fact, Dreamweaver has always offered clever JavaScript programmers a way to add their own commands to the program.

In addition, the programming language for Flash—ActionScript—is based on JavaScript, so if you learn the basics of JavaScript, you'll be well prepared to learn Flash programming.

What Is jQuery?

JavaScript has one embarrassing little secret: writing it is hard. While it's simpler than many other programming languages, JavaScript is still a programming language. And many people, including web designers, find programming difficult. To complicate matters further, different web browsers understand JavaScript differently, so a program that works in, say, Chrome may be completely unresponsive in Internet Explorer 9. This common situation can cost many hours of testing on different machines and different browsers to make sure a program works correctly for your site's entire audience.

That's where jQuery comes in. jQuery is a JavaScript library intended to make JavaScript programming easier and more fun. A JavaScript library is a complex JavaScript program that both simplifies difficult tasks and solves cross-browser problems. In other words, jQuery solves the two biggest headaches with JavaScript—complexity and the finicky nature of different web browsers.

jQuery is a web designer's secret weapon in the battle of JavaScript programming. With jQuery, you can accomplish tasks in a single line of code that would otherwise take hundreds of lines of programming and many hours of browser testing to achieve with your own JavaScript code. In fact, an in-depth book solely about JavaScript would be at least twice as thick as the one you're holding; and, when you were done reading it (if you could manage to finish it), you wouldn't be able to do half of the things you can accomplish with just a little bit of jQuery knowledge.

That's why most of this book is about jQuery. It lets you do so much, so easily. Another great thing about jQuery is that you can add advanced features to your website with thousands of easy-to-use jQuery plug-ins. For example, the FancyBox plug-in (which you'll meet on page 222) lets you take a simple page of thumbnail graphics and turn it into an interactive slideshow—all with a single line of programming!

Unsurprisingly, jQuery is used on millions of websites (*http://trends.builtwith.com/javascript/JQuery*). It's baked right into popular content management systems like Drupal and WordPress. You can even find job listings for "jQuery Programmers" with no mention of JavaScript. When you learn jQuery, you join a large community of fellow web designers and programmers who use a simpler and more powerful approach to creating interactive, powerful web pages.

HTML: The Barebones Structure

JavaScript isn't much good without the two other pillars of web design—HTML and CSS. Many programmers talk about the three languages as forming the "layers" of a web page: HTML provides the *structural* layer, organizing content like pictures and words in a meaningful way; CSS (Cascading Style Sheets) provides the *presentational* layer, making the content in the HTML look good; and JavaScript adds a *behavioral* layer, bringing a web page to life so it interacts with web visitors.

In other words, to master JavaScript, you need to have a good understanding of both HTML and CSS.

Note: For a full-fledged introduction to HTML and CSS, check out *Head First HTML with CSS and XHTML* by Elisabeth Freeman and Eric Freeman. For an in-depth presentation of the tricky subject of Cascading Style Sheets, pick up a copy of *CSS: The Missing Manual* by David Sawyer McFarland (both O'Reilly).

HTML (Hypertext Markup Language) uses simple commands called *tags* to define the various parts of a web page. For example, this HTML code creates a simple web page:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>Hey, I am the title of this web page.</title>
</head>
<body>
Hey, I am some body text on this web page.
</body>
</html>
```

It may not be exciting, but this example has all the basic elements a web page needs. This page begins with a single line—the document type declaration, or *doctype* for short—that states what type of document the page is and which standards it conforms to. HTML actually comes in different versions, and you use a different doctype with each. In this example, the doctype is for HTML5; the doctype for an HTML 4.01 or XHTML document is longer and also includes a URL that points the web browser to a file on the Internet that contains definitions for that type of file.

In essence, the doctype tells the web browser how to display the page. The doctype can even affect how CSS and JavaScript work. With an incorrect or missing doctype, you may end up banging your head against a wall as you discover lots of cross-browser differences with your scripts. If for no other reason, always include a doctype in your HTML.

There are five types of HTML commonly used today: HTML 4.01 Transitional, HTML 4.01 Strict, XHTML 1.0 Transitional, XHTML 1.0 Strict, and HTML5 (the new kid on the block). All five are very much alike, with just slight differences in how tags are written and which tags and attributes are allowed. Most web page editing programs add an appropriate doctype when you create a new web page, but if you want examples of how each is written, you can find templates for the different types of pages at *www.webstandards.org/learn/reference/templates*.

It doesn't really matter which type of HTML you use. All current web browsers understand each of the five common doctypes and can display web pages using any of the four document types without problem. Which doctype you use isn't nearly as important as making sure you've correctly written your HTML tags—a task that's helped by validating the page, as described in the box on page 7.

Note: XHTML was once heralded as the next big thing for web designers. Although you'll still find people who think you should only use XHTML, the winds of change have turned. The World Wide Web Consortium (W3C) has stopped development of XHTML in favor of HTML5. You can learn more about HTML5 by picking up a copy of *HTML5: The Missing Manual* by Matthew MacDonald or *HTML5: Up and Running* by Mark Pilgrim (both from O'Reilly).

How HTML Tags Work

In the example on the previous page, as in the HTML code of any web page, you'll notice that most commands appear in pairs that surround a block of text or other

commands. Sandwiched between brackets, these *tags* are instructions that tell a web browser how to display the web page. Tags are the "markup" part of the Hypertext Markup Language.

The starting (*opening*) tag of each pair tells the browser where the instruction begins, and the ending tag tells it where the instruction ends. Ending or *closing* tags always include a forward slash (/) after the first bracket symbol (<). For example, the tag marks the start of a paragraph, while marks its end.

For a web page to work correctly, you must include at least these three tags:

- The **<html>** tag appears once at the beginning of a web page (after the doctype) and again (with an added slash) at the end. This tag tells a web browser that the information contained in this document is written in HTML, as opposed to some other language. All of the contents of a page, including other tags, appear between the opening and closing <html> tags.

  If you were to think of a web page as a tree, the <html> tag would be its trunk. Springing from the trunk are two branches that represent the two main parts of any web page—the *head* and the *body*.

- The *head* of a web page, surrounded by **<head>** tags, contains the title of the page. It may also provide other, invisible information (such as search keywords) that browsers and web search engines can exploit.

  In addition, the head can contain information that's used by the web browser for displaying the web page and for adding interactivity. You put Cascading Style Sheets, for example, in the head of the document. The head of the document is also where you often include JavaScript programming and links to JavaScript files.

- The *body* of a web page, as set apart by its surrounding **<body>** tags, contains all the information that appears inside a browser window: headlines, text, pictures, and so on.

Within the <body> tag, you commonly find tags like the following:

- You tell a web browser where a paragraph of text begins with a ** ** (opening paragraph tag), and where it ends with a ** ** (closing paragraph tag).

- The **** tag emphasizes text. If you surround some text with it and its partner tag, , you get boldface type. The HTML snippet Warning! tells a web browser to display the word "Warning!" in bold type.

- The **<a>** tag, or anchor tag, creates a *hyperlink* in a web page. When clicked, a hyperlink—or *link*—can lead anywhere on the web. You tell the browser where the link points by putting a web address inside the <a> tags. For instance, you might type *Click here!*.

  The browser knows that when your visitor clicks the words "Click here!" it should go to the Missing Manual website. The *href* part of the tag is called an *attribute* and the URL (the Uniform Resource Locator or web address) is the *value*. In this example, *http://www.missingmanuals.com* is the *value* of the *href* attribute.