



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Learning jQuery

Third Edition

Create better interaction, design, and web development with
simple JavaScript techniques

Foreword by John Resig, Creator of jQuery

Jonathan Chaffer
Karl Swedberg

[PACKT] open source*
PUBLISHING community experience distilled

Learning jQuery

Third Edition

Create better interaction, design, and web development
with simple JavaScript techniques

Jonathan Chaffer

Karl Swedberg



BIRMINGHAM - MUMBAI

Learning jQuery

Third Edition

Copyright © 2011 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2011

Production Reference: 1160911

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK..

ISBN 978-1-84951-654-9

www.packtpub.com

Cover Image by Karl Swedberg (kswedberg@gmail.com)

Credits

Authors

Jonathan Chaffer
Karl Swedberg

Reviewers

Kaiser Ahmed
Kevin Boudloche
Carlos Esteves

Acquisition Editor

Sarah Cullington

Development Editor

Roger D'souza

Technical Editors

Llewellyn F. Rozario
Azharuddin Sheikh

Project Coordinator

Srimoyee Ghoshal

Proofreader

Linda Morris

Indexers

Tejal Daruwale
Rekha Nair

Graphics

Nilesh Mohite

Production Coordinators

Aparna Bhagat
Prachali Bhiwandkar

Cover Work

Aparna Bhagat
Prachali Bhiwandkar

Foreword

I feel honored knowing that Karl Swedberg and Jonathan Chaffer undertook the task of writing *Learning jQuery*. As the first book about jQuery, it set the standard that other jQuery – and, really, other JavaScript books in general – have tried to match. It's consistently been one of the top selling JavaScript books since its release, in no small part due to its quality and attention to detail.

I'm especially pleased that it was Karl and Jonathan who wrote the book as I already knew them so well and knew that they would be perfect for the job. Being part of the core jQuery team, I've had the opportunity to come to know Karl quite well over the past couple years, and especially within the context of his book writing effort. Looking at the end result, it's clear that his skills as both a developer and a former English teacher were perfectly designed for this singular task.

I've also had the opportunity to meet both of them in person, a rare occurrence in the world of distributed Open Source projects, and they continue to be upstanding members of the jQuery community.

The jQuery library is used by so many different people in the jQuery community. The community is full of designers, developers, people who have experience programming, and those who don't. Even within the jQuery team, we have people from all backgrounds providing their feedback on the direction of the project. There is one thing that is common across all of jQuery's users, though: We are a community of developers and designers who want JavaScript development to be made simple.

It's almost a cliché, at this point, to say that an open source project is community-oriented, or that a project wants to focus on helping new users get started. However, it's not just an empty gesture for jQuery; it's the liquid-oxygen fuel for the project. We actually have more people in the jQuery team dedicated to managing the jQuery community, writing documentation, or writing plugins than actually maintaining the core code base. While the health of the library is incredibly important, the community surrounding that code is the difference between a floundering, mediocre project and one that will match and exceed your every need.

How we run the project, and how you use the code, is fundamentally very different from most open source projects – and most JavaScript libraries. The jQuery project and community is incredibly knowledgeable; we understand what makes jQuery a different programming experience and do our best to pass that knowledge on to fellow users.

The jQuery community isn't something that you can read about to understand; it's something that you actually have to participate in for it to fully sink in. I hope that you'll have the opportunity to partake in it. Come join us in our forums, mailing lists, and blogs and let us help guide you through the experience of getting to know jQuery better.

For me, jQuery is much more than a block of code. It's the sum total of experiences that have transpired over the years in order to make the library happen. The considerable ups and downs, the struggle of development together with the excitement of seeing it grow and succeed. Growing close with its users and fellow team members, understanding them and trying to grow and adapt.

When I first saw this book talk about jQuery and discuss it like a unified tool, as opposed to the experiences that it's come to encapsulate for me, I was both taken aback and excited. Seeing how others learn, understand, and mold jQuery to fit them is much of what makes the project so exhilarating.

I'm not the only one who enjoys jQuery on a level that is far different from a normal tool-user relationship. I don't know if I can properly encapsulate why this is, but I've seen it time and time again – the singular moment when a user's face lights up with the realization of just how much jQuery will help them.

There is a specific moment where it just clicks for a jQuery user, when they realize that this tool that they were using was in fact much, much more than just a simple tool all along – and suddenly their understanding of how to write dynamic web applications completely shifts. It's an incredible thing, and absolutely my favorite part of the jQuery project.

I hope you'll have the opportunity to experience this sensation as well.

John Resig

Creator of jQuery

About the Authors

Jonathan Chaffer is a member of Rapid Development Group, a web development firm located in Grand Rapids, Michigan. His work there includes overseeing and implementing projects in a wide variety of technologies, with an emphasis in PHP, MySQL, and JavaScript. He also leads on-site training seminars on the jQuery framework for web developers.

In the open-source community, Jonathan has been very active in the Drupal CMS project, which has adopted jQuery as its JavaScript framework of choice. He is the creator of the Content Construction Kit, a popular module for managing structured content on Drupal sites. He is responsible for major overhauls of Drupal's menu system and developer API reference.

Jonathan lives in Grand Rapids with his wife, Jennifer.

I would like to thank Jenny for her tireless enthusiasm and support, Karl for the motivation to continue writing when the spirit is weak, and the Ars Technica community for constant inspiration toward technical excellence. In addition, I'd like to thank Mike Henry and the Twisted Pixel team for producing consistently entertaining distractions in between writing sessions.

Karl Swedberg is a web developer at Fusionary Media in Grand Rapids, Michigan, where he spends much of his time making cool things happen with JavaScript. As a member of the jQuery team, Karl is responsible for maintaining the jQuery API site at `api.jquery.com`. He also publishes tutorials on his blog, `learningjquery.com`, and presents at workshops and conferences. When he isn't coding, Karl likes to hang out with his family, roast coffee in his garage, and exercise at the local cross-fit gym.

I wish to thank my wife, Sara, and my two children, Benjamin and Lucia, for all the joy that they bring into my life. Thanks also to Jonathan Chaffer for his patience and his willingness to write this book with me.

Many thanks to John Resig for creating the world's greatest JavaScript library and to all the others who have contributed their code, time, and expertise to the project. Thanks to the folks at Packt Publishing, the technical reviewers of this book, the jQuery Cabal, and the many others who have provided help and inspiration along the way.

About the Reviewers

Kaiser Ahmed is a professional web developer. He has gained his Bachelor's Degree from Khulna University of Engineering and Technology (KUET). He is also a co-founder of fully outsourcing company CyberXpress.Net Inc based on Bangladesh.

He has a wide breadth of technical skills, Internet knowledge, and experience across the spectrum of online development in the service of building and improving online properties for multiple clients. He enjoys creating site architecture and infrastructure, backend development using open source toolset (PHP, MySQL, Apache, Linux, and others (that is LAMP)), frontend development with CSS and HTML/XHTML.

He would like to thank his loving wife, Maria Akter, for her support.

Kevin Boudloche is a web developer out of Mississippi. He has been building web pages as a hobby for more than eight years and for three years professionally. Kevin's primary focus is front-end development and web application development.

Carlos Esteves is the founder of Ehxioz (<http://ehxioz.com/>) a Los Angeles-based software development startup that specializes in developing modern web applications and utilizing the latest web development technologies & methodologies. He has over 10 years of web development experience and holds a BSc in Computer Science from California State University, Los Angeles.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Getting Started	9
What jQuery does	9
Why jQuery works well	11
Our first jQuery-powered web page	12
Downloading jQuery	12
Setting up jQuery in an HTML document	13
Adding our jQuery code	16
Finding the poem text	17
Injecting the new class	17
Executing the code	17
The finished product	19
Plain JavaScript vs. jQuery	19
Development tools	20
Firebug	21
Summary	24
Chapter 2: Selecting Elements	25
The Document Object Model	25
The \$() function	27
CSS selectors	28
Styling list-item levels	29
Attribute selectors	31
Styling links	31
Custom selectors	34
Styling alternate rows	34
Form selectors	39
DOM traversal methods	39
Styling specific cells	41
Chaining	43

Accessing DOM elements	44
Summary	45
Further reading	45
Exercises	45
Chapter 3: Handling Events	47
Performing tasks on page load	47
Timing of code execution	47
Multiple scripts on one page	48
Shortcuts for code brevity	50
Passing an argument to the .ready() callback	50
Simple events	51
A simple style switcher	51
Enabling the other buttons	54
Event handler context	55
Further consolidation	57
Shorthand events	59
Compound events	61
Showing and hiding advanced features	61
Highlighting clickable items	63
The journey of an event	65
Side effects of event bubbling	66
Altering the journey: the event object	67
Event targets	68
Stopping event propagation	69
Default actions	70
Event delegation	70
Methods for event delegation	73
Removing an event handler	74
Event namespacing	75
Rebinding events	75
Simulating user interaction	78
Keyboard events	79
Summary	82
Further reading	83
Exercises	83
Chapter 4: Styling and Animating	85
Inline CSS modification	85
Basic hide and show	90
Effects and speed	92
Speeding in	93

Fading in and fading out	94
Sliding up and sliding down	94
Compound effects	95
Creating custom animations	97
Building effects by hand	98
Animating multiple properties at once	99
Positioning with CSS	101
Simultaneous versus queued effects	103
Working with a single set of elements	103
Bypassing the queue	104
Manual queueing	105
Working with multiple sets of elements	107
Callbacks	109
In a nutshell	111
Summary	111
Further reading	112
Exercises	112
Chapter 5: Manipulating the DOM	113
Manipulating attributes	113
Non-class attributes	114
Value callbacks	115
DOM element properties	118
DOM tree manipulation	119
The <code>\$()</code> function revisited	119
Creating new elements	119
Inserting new elements	120
Moving elements	122
Wrapping elements	124
Inverted insertion methods	126
Copying elements	129
Cloning for pull quotes	130
Content getter and setter methods	133
Further style adjustments	135
DOM manipulation methods in a nutshell	136
Summary	137
Further reading	138
Exercises	138
Chapter 6: Sending Data with Ajax	139
Loading data on demand	139
Appending HTML	141

Working with JavaScript objects	144
Retrieving JSON	144
Global jQuery functions	146
Executing a script	149
Loading an XML document	151
Choosing a data format	154
Passing data to the server	155
Performing a GET request	156
Performing a POST request	160
Serializing a form	161
Delivering different content for Ajax requests	164
Keeping an eye on the request	165
Error handling	168
Ajax and events	169
Security limitations	170
Using JSONP for remote data	172
Additional options	174
The low-level Ajax method	174
Modifying default options	175
Loading parts of an HTML page	175
Summary	178
Further reading	178
Exercises	179
Chapter 7: Using Plugins	181
Finding plugins and help	181
How to use a plugin	182
Downloading and referencing the Cycle plugin	182
Simple plugin use	182
Specifying plugin method parameters	184
Parameter defaults	185
Other types of plugins	186
Custom selectors	186
Global function plugins	187
The jQuery UI plugin library	188
Effects	189
Color animations	189
Class animations	190
Advanced easing	190
Additional effects	191
Interaction components	192

Widgets	194
jQuery UI ThemeRoller	197
Summary	197
Exercises	198
Chapter 8: Developing Plugins	199
<hr/>	
Use of the \$ alias in plugins	199
Adding new global functions	200
Adding multiple functions	202
Adding jQuery object methods	205
Object method context	206
Implicit iteration	207
Method chaining	208
Method parameters	209
Parameter maps	211
Default parameter values	212
Callback functions	213
Customizable defaults	214
The jQuery UI widget factory	216
Creating a widget	217
Destroying widgets	219
Enabling and disabling widgets	220
Accepting widget options	220
Adding sub-methods	221
Triggering widget events	222
Plugin design recommendations	223
Plugin distribution	224
Summary	224
Exercises	225
Chapter 9: Advanced Selectors and Traversing	227
<hr/>	
Selecting and traversing revisited	227
Dynamic table filtering	229
Table row striping	231
Combining filtering and striping	233
More selectors and traversal methods	234
Customizing and optimizing selectors	235
Writing a custom selector plugin	235
Selector performance	237
Sizzle selector implementation	238
Testing selector speed	239

DOM traversal under the hood	240
jQuery object properties	241
The DOM element stack	243
Writing a DOM traversal method plugin	244
DOM traversal performance	246
Improving performance using chaining	246
Improving performance using caching	247
Summary	248
Further reading	248
Exercises	248
Chapter 10: Advanced Events	251
Events revisited	251
Loading additional pages of data	253
Displaying data on hover	254
Event delegation	256
Using jQuery's delegation methods	257
Choosing a delegation method	257
Delegating early	259
Using a context argument	260
Custom events	260
Infinite scrolling	262
Custom event parameters	263
Throttling events	264
Other ways to perform throttling	265
Special events	266
More about special events	268
Summary	268
Further reading	269
Exercises	269
Chapter 11: Advanced Effects	271
Animation revisited	271
Observing and interrupting animations	274
Determining the animation state	274
Halting a running animation	275
Caution when halting animations	276
Global effect properties	276
Globally disabling all effects	276
Fine-tuning animation smoothness	277
Defining effect durations	277
Multi-property easing	280

Deferred objects	281
Animation promises	282
Summary	285
Further reading	285
Exercises	286
Chapter 12: Advanced DOM Manipulation	287
<hr/>	
Sorting table rows	287
Server-side sorting	287
Ajax sorting	288
JavaScript sorting	289
Moving and inserting elements, revisited	290
Adding links around existing text	290
Sorting simple JavaScript arrays	291
Sorting DOM elements	292
Storing data alongside DOM elements	294
Performing additional precomputation	295
Storing non-string data	296
Alternating sort directions	299
Using HTML5 custom data attributes	300
Sorting and building rows with JSON	303
Modifying the JSON object	305
Rebuilding content on demand	306
Advanced attribute manipulation	308
Shorthand element creation	308
DOM manipulation hooks	309
Writing a CSS hook	310
Summary	312
Further reading	312
Exercises	313
Chapter 13: Advanced Ajax	315
<hr/>	
Progressive enhancement with Ajax	315
Harvesting JSONP data	317
Ajax error handling	321
The jqXHR object	323
Ajax promises	323
Caching responses	325
Throttling Ajax requests	327
Extending Ajax capabilities	328
Data type converters	328
Ajax prefilters	333

Alternate transports	334
Summary	338
Further reading	338
Exercises	339
Appendix A: JavaScript Closures	341
Inner functions	341
The great escape	343
Variable scoping	344
Interactions between closures	346
Closures in jQuery	347
Arguments to \$(document).ready()	348
Event handlers	348
Binding handlers in loops	350
Named and anonymous functions	352
Memory leak hazards	353
Accidental reference loops	354
The Internet Explorer memory leak problem	355
The good news	356
Summary	356
Appendix B: Testing JavaScript with QUnit	357
Downloading QUnit	358
Setting up the document	358
Organizing tests	359
Adding and running tests	360
Asynchronous testing	363
Other types of tests	364
Practical considerations	364
Further reading	365
Summary	366
Appendix C: Quick Reference	367
Selector expressions	367
Simple CSS	367
Position among siblings	368
Position among matched elements	368
Attributes	369
Forms	369
Other custom selectors	370
DOM traversal methods	370
Filtering	370
Descendants	371

Siblings	371
Ancestors	372
Collection manipulation	372
Working with selected elements	373
Event methods	373
Binding	374
Shorthand binding	374
Special shorthands	376
Triggering	376
Shorthand triggering	376
Utility	377
Effect methods	377
Predefined effects	377
Custom animations	378
Queue manipulation	378
DOM manipulation methods	378
Attributes and properties	378
Content	379
CSS	379
Dimensions	380
Insertion	381
Replacement	381
Removal	382
Copying	382
Data	382
Ajax methods	382
Issuing requests	383
Request monitoring	383
Configuration	384
Utilities	384
Deferred objects	384
Object creation	384
Methods of deferred objects	385
Methods of promise objects	385
Miscellaneous properties and functions	385
Properties of the jQuery object	386
Arrays and objects	386
Object introspection	386
Other	387
Index	389

Preface

In 2005, inspired by pioneers in the field such as Dean Edwards and Simon Willison, John Resig put together a set of functions to make it easy to programmatically find elements on a web page and assign behaviors to them. By the time he first publicly announced his project in January 2006, he had added DOM modification and basic animations. He gave it the name jQuery to emphasize the central role of finding, or querying, parts of a web page and acting on them with JavaScript. In the few short years since then, jQuery has grown in its feature set, improved in its performance, and gained widespread adoption by many of the most popular sites on the Internet. While Resig remains the lead developer of the project, jQuery has blossomed, in true open-source fashion, to the point where it now boasts a core team of top-notch JavaScript developers, as well as a vibrant community of thousands of developers.

The jQuery JavaScript library can enhance your websites regardless of your background. It provides a wide range of features, an easy-to-learn syntax, and robust cross-platform compatibility in a single compact file. What's more, hundreds of plugins have been developed to extend jQuery's functionality, making it an essential tool for nearly every client-side scripting occasion.

Learning jQuery Third Edition provides a gentle introduction to jQuery concepts, allowing you to add interactions and animations to your pages – even if previous attempts at writing JavaScript have left you baffled. This book guides you past the pitfalls associated with Ajax, events, effects, and advanced JavaScript language features, and provides you with a brief reference to the jQuery library to return to again and again.

What This Book Covers

In *Chapter 1, Getting Started*, you'll get your feet wet with the jQuery JavaScript library. The chapter begins with a description of jQuery and what it can do for you. It then walks you through downloading and setting up the library, as well as writing your first script.

In *Chapter 2, Selecting Elements*, you'll learn how to use jQuery's selector expressions and DOM traversal methods to find elements on the page, wherever they may be. You'll use jQuery to apply styling to a diverse set of page elements, sometimes in a way that pure CSS cannot.

In *Chapter 3, Handling Events*, you'll use jQuery's event-handling mechanism to fire off behaviors when browser events occur. You'll see how jQuery makes it easy to attach events to elements unobtrusively, even before the page finishes loading. Also, you'll get an overview of deeper topics, such as event bubbling, delegation, and namespacing.

In *Chapter 4, Styling and Animating*, you'll be introduced to jQuery's animation techniques and see how to hide, show, and move page elements with effects that are both useful and pleasing to the eye.

In *Chapter 5, Manipulating the DOM*, you'll learn how to change your page on command. This chapter will teach you how to alter the very structure of an HTML document, as well as its content, on the fly.

In *Chapter 6, Sending Data with Ajax*, you'll discover the many ways in which jQuery makes it easy to access server-side functionality without resorting to clunky page refreshes. With the basic components of the library well in hand, you will be ready to explore how the library can expand to fit your needs.

In *Chapter 7, Using Plugins*, will show you how to find, install, and use plugins, including the powerful jQuery UI plugin library.

In *Chapter 8, Developing Plugins*, you'll learn how to take advantage of jQuery's impressive extension capabilities to develop your own plugins from the ground up. You'll create your own utility functions, add jQuery object methods, and discover the jQuery UI widget factory. Next, you'll take a second tour through jQuery's building blocks, learning more advanced techniques.

In *Chapter 9, Advanced Selectors and Traversing*, you'll refine your knowledge of selectors and traversals, gaining the ability to optimize selectors for performance, manipulate the DOM element stack, and write plugins that expand selecting and traversing capabilities.

In *Chapter 10, Advanced Events*, you'll dive further into techniques such as delegation and throttling that can greatly improve event handling performance. You'll also create custom and special events that add even more capabilities to the jQuery library.

In *Chapter 11, Advanced Effects*, you'll fine-tune the visual effects jQuery can provide by crafting custom easing functions and reacting to each step of an animation. You'll gain the ability to manipulate animations as they occur, and schedule actions with custom queuing.

In *Chapter 12, Advanced DOM Manipulation*, you'll get more practice modifying the DOM, with techniques such as attaching arbitrary data to elements. You'll also learn how to extend the way jQuery processes CSS properties on elements.

In *Chapter 13, Advanced Ajax*, you'll achieve a greater understanding of Ajax transactions, including the jQuery deferred object system for handling data that may become available at a later time.

In *Appendix A, JavaScript Closures*, you'll gain a solid understanding of closures in JavaScript—what they are and how you can use them to your advantage.

In *Appendix B, Testing JavaScript with QUnit*, you'll learn about the QUnit library for unit testing of JavaScript programs. This library will add to your toolkit for developing and maintaining highly sophisticated web applications.

In *Appendix C, Quick Reference*, you'll get a glimpse of the entire jQuery library, including every one of its methods and selector expressions. Its easy-to-scan format is perfect for those moments when you know what you want to do, but you're just unsure about the right method name or selector.

What you need for this book

In order to run the example code demonstrated in this book, you need a modern web browser such as Mozilla Firefox, Apple Safari, Google Chrome, or Microsoft Internet Explorer.

To experiment with the examples and to work on the chapter-ending exercises, you will also need:

- A basic text editor
- Web development tools for the browser such as Firebug (as described in *Chapter 1* in the *Development Tools* section)
- The full code package for each chapter, which includes a copy of the jQuery library (seen in the following *Downloading the example code* section)

Additionally, to run some of the Ajax examples in *Chapter 6* and beyond, you will need a PHP-enabled web server.

Who this book is for

This book is for web designers who want to create interactive elements for their designs, and for developers who want to create the best user interface for their web applications. Basic JavaScript programming knowledge is required. You will need to know the basics of HTML and CSS, and should be comfortable with the syntax of JavaScript. No knowledge of jQuery is assumed, nor is experience with any other JavaScript libraries required.

By reading this book, you will become familiar with the functionality and syntax of jQuery 1.6.x, the latest version at the time of writing.

History of the jQuery project

This book covers the functionality and syntax of jQuery 1.6.x, the latest version at the time of writing. The premise behind the library – providing an easy way to find elements on a web page and manipulate them – has not changed over the course of its development, but some syntax details and features have. This brief overview of the project history describes the most significant changes from version to version, which may prove helpful to readers working with legacy versions of the library.

- **Public Development Phase:** John Resig first made mention of an improvement on Prototype's *Behavior* library in August of 2005. This new framework was formally released as jQuery on January 14, 2006.
- **jQuery 1.0** (August 2006): This, the first stable release of the library, already had robust support for CSS selectors, event handling, and AJAX interaction.
- **jQuery 1.1** (January 2007): This release streamlined the API considerably. Many rarely-used methods were combined, reducing the number of methods to learn and document.
- **jQuery 1.1.3** (July 2007): This minor release contained massive speed improvements for jQuery's selector engine. From this version on, jQuery's performance would compare favorably to its fellow JavaScript libraries such as Prototype, Mootools, and Dojo.
- **jQuery 1.2** (September 2007): XPath syntax for selecting elements was removed in this release, as it had become redundant with the CSS syntax. Effect customization became much more flexible in this release, and plugin development became easier with the addition of **namespaced events**.

- **jQuery UI** (September 2007): This new plugin suite was announced to replace the popular, but aging, Interface plugin. A rich collection of prefabricated widgets was included, as well as a set of tools for building sophisticated elements such as drag-and-drop interfaces.
- **jQuery 1.2.6** (May 2008): The functionality of Brandon Aaron's popular Dimensions plugin was brought into the main library.
- **jQuery 1.3** (January 2009): A major overhaul of the selector engine (**Sizzle**) provided a huge boost to the library's performance. **Event delegation** became formally supported.
- **jQuery 1.4** (January 2010): This version, perhaps the most ambitious update since 1.0, brought many performance improvements to DOM manipulation, as well as a large number of new or enhanced methods to nearly every aspect of the library. Version 1.4 was accompanied by fourteen days of announcements and videos on a dedicated website, <http://jquery14.com/>.
- **jQuery 1.4.2** (February 2010): Two new event delegation methods, `.delegate()` and `.undelegate()`, were added, and jQuery's entire event system saw a comprehensive overhaul for more flexible use and greater cross-browser consistency.
- **jQuery Mobile** (August 2010): The jQuery Project publicly outlined its strategy, research, and UI designs for mobile web development with jQuery and a new mobile framework at <http://jquerymobile.com/>.
- **jQuery 1.5** (January 2011): The Ajax component underwent a major rewrite, adding greater extensibility and performance. Additionally, jQuery 1.5 included an implementation of the Promise pattern for handling queues of both synchronous and asynchronous functions.
- **jQuery 1.6** (May 2011): The Attribute component was rewritten to more accurately reflect the distinction between HTML attributes and DOM properties. Also, the Deferred object, which was introduced in jQuery 1.5, received two new methods: `.always()` and `.pipe()`.



Historical Details

Release notes for older jQuery versions can be found on the project's website at <http://jquery.org/history>.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "This code illustrates that we can pass any kind of expression into the `console.log()` method."


A block of code is set as follows:


```
$('#button.show-details').click(function() {  
    $('#div.details').show();  
});
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
$('#switcher-narrow').bind('click', function() {  
    $('#body').removeClass().addClass('narrow');  
});
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "The **Console** tab will be of most frequent use to us while learning jQuery, as shown in the following screenshot".

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or e-mail suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Getting Started

Today's World Wide Web is a dynamic environment, and its users set a high bar for both style and function of sites. To build interesting, interactive sites, developers are turning to JavaScript libraries such as jQuery to automate common tasks and simplify complicated ones. One reason for jQuery's popularity is its ability to assist in a wide range of tasks.

It can seem challenging to know where to begin because jQuery performs so many different functions. Yet, there is a coherence and symmetry to the design of the library; many of its concepts are borrowed from the structure of **HTML** and **Cascading Style Sheets (CSS)**. The library's design lends itself to a quick start for designers with little programming experience as many of them have more experience with these technologies than they do with JavaScript. In fact, in this opening chapter, we'll write a functioning jQuery program in just three lines of code. On the other hand, experienced programmers will also be aided by this conceptual consistency, as we'll see in the later, more advanced chapters.

So let's look at what jQuery can do for us.

What jQuery does

The jQuery library provides a general-purpose abstraction layer for common web scripting, and is, therefore, useful in almost every scripting situation. Its extensible nature means that we could never cover all possible uses and functions in a single book, as plugins are constantly being developed to add new abilities. The core features, though, assist us in accomplishing the following tasks:

- Access elements in a document: Without a JavaScript library, web developers often need to write many lines of code to traverse the **Document Object Model (DOM)** tree and locate specific portions of an HTML document's structure. With jQuery, developers have a robust and efficient selector mechanism at their disposal, making it easy to retrieve the exact piece of the document that needs to be inspected or manipulated.

```
$('div.content').find('p');
```

- Modify the appearance of a web page: CSS offers a powerful method of influencing the way a document is rendered, but it falls short when web browsers do not all support the same standards. With jQuery, developers can bridge this gap, relying on the same standards support across all browsers. In addition, jQuery can change the classes or individual style properties applied to a portion of the document even after the page has been rendered.

```
$('#ul > li:first').addClass('active');
```

- Alter the content of a document: Not limited to mere cosmetic changes, jQuery can modify the content of a document itself with a few keystrokes. Text can be changed, images can be inserted or swapped, lists can be reordered, or the entire structure of the HTML can be rewritten and extended – all with a single easy-to-use **Application Programming Interface (API)**.

```
$('#container').append('<a href="more.html">more</a>');
```

- Respond to a user's interaction: Even the most elaborate and powerful behaviors are not useful if we can't control when they take place. The jQuery library offers an elegant way to intercept a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers. At the same time, its event-handling API removes browser inconsistencies that often plague web developers.

```
$('#button.show-details').click(function() {  
    $('#div.details').show();  
});
```

- Animate changes being made to a document: To effectively implement such interactive behaviors, a designer must also provide visual feedback to the user. The jQuery library facilitates this by providing an array of effects such as fades and wipes, as well as a toolkit for crafting new graphic displays.

```
$('#div.details').slideDown();
```

- Retrieve information from a server without refreshing a page: This code pattern has become known as **Ajax**, which originally stood for **asynchronous JavaScript and XML**, but has since come to represent a much greater set of technologies for communicating between the client and the server. The jQuery library removes the browser-specific complexity from this responsive, feature-rich process, allowing developers to focus on the server-end functionality.

```
$('#div.details').load('more.html #content');
```

- Simplify common JavaScript tasks: In addition to all of the document-specific features of jQuery, the library provides enhancements to basic JavaScript constructs such as iteration and array manipulation.

```
$.each(obj, function(key, value) {  
    total += value;  
});
```



Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

Why jQuery works well

With the resurgence of interest in dynamic HTML comes a proliferation of JavaScript frameworks. Some are specialized, focusing on just one or two of the above tasks. Others attempt to catalog every possible behavior and animation, and serve these all up pre-packaged. To maintain the wide range of features outlined above while remaining relatively compact, jQuery employs several strategies:

- Leverage knowledge of CSS: By basing the mechanism for locating page elements on **CSS selectors**, jQuery inherits a terse, yet legible, way of expressing a document's structure. The jQuery library becomes an entry point for designers who want to add behaviors to their pages because a prerequisite for doing professional web development is to have knowledge of CSS syntax.
- Support extensions: In order to avoid *feature creep*, jQuery relegates special-case uses to **plugins**. The method for creating new plugins is simple and well-documented, which has spurred the development of a wide variety of inventive and useful modules. Most of the features in the basic jQuery download are internally realized through the plugin architecture, and can be removed if desired, yielding an even smaller library.
- Abstract away browser quirks: An unfortunate reality of web development is that each browser has its own set of deviations from published standards. A significant portion of any web application can be relegated to handle features differently on each platform. While the ever-evolving browser landscape makes a perfectly browser-neutral code base impossible for some advanced features, jQuery adds an **abstraction layer** that normalizes the common tasks, reducing the size of code while tremendously simplifying it.

- Always work with sets: When we instruct jQuery, "Find all elements with the class `collapsible` and hide them," there is no need to loop through each returned element. Instead, methods such as `.hide()` are designed to automatically work on sets of objects instead of individual ones. This technique, called **implicit iteration**, means that many looping constructs become unnecessary, shortening code considerably.
- Allow multiple actions in one line: To avoid overuse of temporary variables or wasteful repetition, jQuery employs a programming pattern called **chaining** for the majority of its methods. This means that the result of most operations on an object is the object itself, ready for the next action to be applied to it.

These strategies have kept the jQuery package slim — roughly 30 KB, compressed — while at the same time providing techniques to keep our custom code that uses the library compact.

The elegance of the library comes about partly by design, and partly due to the evolutionary process spurred by the vibrant community that has sprung up around the project. Users of jQuery gather to discuss not only the development of plugins, but also enhancements to the core library. The users and developers also assist in continually improving the official project documentation, which can be found at <http://api.jquery.com>.

Despite all of the efforts required to engineer such a flexible and robust system, the end product is free for all to use. This open source project is dually licensed under the **MIT License** (to permit free use of jQuery on any site and facilitate its use within proprietary software) and the **GNU Public License** (appropriate for inclusion in other GNU-licensed open-source projects).

Our first jQuery-powered web page

Now that we have covered the range of features available to us with jQuery, we can examine how to put the library into action. To get started, we need a copy of jQuery.

Downloading jQuery

No installation is required. To use jQuery, we just need a publicly available copy of the file, whether that copy is on an external site or our own. As JavaScript is an interpreted language, there is no compilation or build phase to worry about. Whenever we need a page to have jQuery available, we will simply refer to the file's location from a `<script>` element in the HTML document.

The official jQuery website (<http://jquery.com/>) always has the most up-to-date, stable version of the library, which can be downloaded right from the home page of the site. Several versions of jQuery may be available at any given moment; the most appropriate for us as site developers will be the latest uncompressed version of the library. This can be replaced with a compressed version in production environments.

As jQuery's popularity has grown, companies have made the file freely available through their **Content Delivery Networks (CDNs)**. Most notably, Google (<http://code.google.com/apis/ajaxlibs/documentation/>), and Microsoft (<http://www.asp.net/ajax/cdn>) offer the file on powerful, low-latency servers distributed around the world for fast download regardless of the user's location. While a CDN-hosted copy of jQuery has speed advantages due to server distribution and caching, using a local copy can be convenient during development. Throughout this book we'll use a copy of the file stored on our own system, which will allow us to run our code whether we're connected to the Internet or not.

Setting up jQuery in an HTML document

There are three pieces to most examples of jQuery usage: the HTML document, CSS files to style it, and JavaScript files to act on it. For our first example, we'll use a page with a book excerpt that has a number of classes applied to portions of it. This page includes a reference to the latest version of the jQuery library, which we have downloaded, renamed `jquery.js`, and placed in our local project directory, as follows:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Through the Looking-Glass</title>

    <link rel="stylesheet" href="01.css">

    <script src="jquery.js"></script>
    <script src="01.js"></script>
  </head>

  <body>
    <h1>Through the Looking-Glass</h1>
    <div class="author">by Lewis Carroll</div>

    <div class="chapter" id="chapter-1">
      <h2 class="chapter-title">1. Looking-Glass House</h2>
```

```
<p>There was a book lying near Alice on the table,
and while she sat watching the White King (for she
was still a little anxious about him, and had the
ink all ready to throw over him, in case he fainted
again), she turned over the leaves, to find some
part that she could read,
<span class="spoken">
  "—;for it's all in some language I don't know,"
</span>
she said to herself.
</p>
<p>It was like this.</p>
<div class="poem">
  <h3 class="poem-title">YKCOWREBBAJ</h3>
  <div class="poem-stanza">
    <div>sevot yhtils eht dna ,gillirb sawT'</div>
    <div>;ebaw eht ni elbmig dna eryl diD</div>
    <div>,sevogorob eht erew ysmim llA</div>
    <div>.ebargtuo shtar emom eht dnA</div>
  </div>
</div>
<p>She puzzled over this for some time, but at last
a bright thought struck her.
<span class="spoken">
  "Why, it's a Looking-glass book, of course! And if
  I hold it up to a glass, the words will all go the
  right way again."
</span>
</p>
<p>This was the poem that Alice read.</p>
<div class="poem">
  <h3 class="poem-title">JABBERWOCKY</h3>
  <div class="poem-stanza">
    <div>'Twas brillig, and the slithy toves</div>
    <div>Did gyre and gimble in the wabe;</div>
    <div>All mimsy were the borogoves,</div>
    <div>And the mome raths outgrabe.</div>
  </div>
</div>
</body>
</html>
```

File Paths

The actual layout of files on the server does not matter. References from one file to another just need to be adjusted to match the organization we choose. In most examples in this book, we will use relative paths to reference files (`../images/foo.png`) rather than absolute paths (`/images/foo.png`). This will allow the code to run locally without the need for a web server.

Immediately following the normal HTML preamble, the stylesheet is loaded. For this example, we'll use the following:

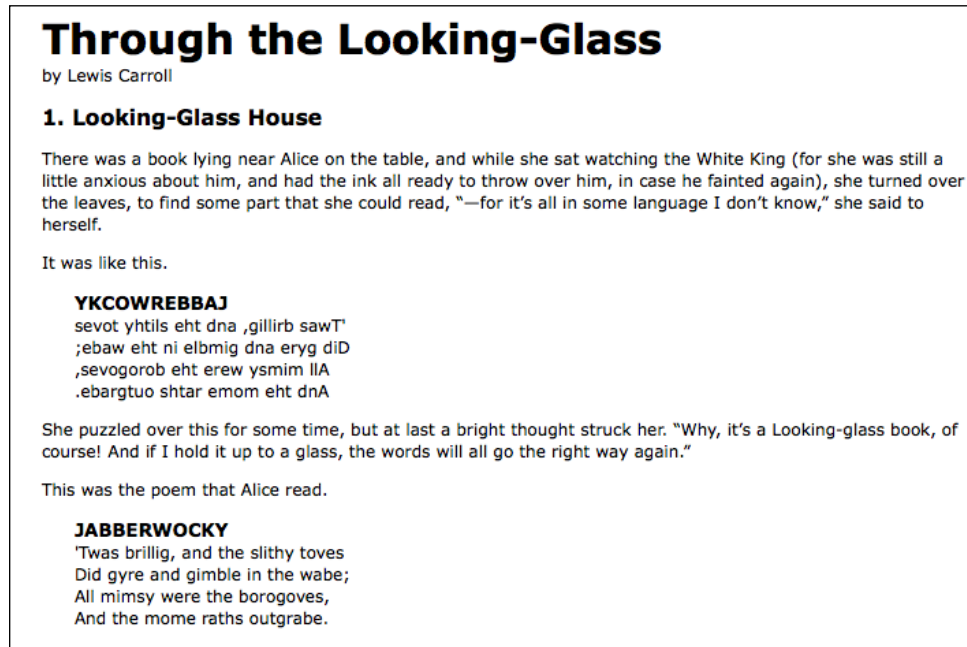
```
body {
  background-color: #fff;
  color: #000;
  font-family: Helvetica, Arial, sans-serif;
}
h1, h2, h3 {
  margin-bottom: .2em;
}
.poe {
  margin: 0 2em;
}
.highlight {
  background-color: #ccc;
  border: 1px solid #888;
  font-style: italic;
  margin: 0.5em 0;
  padding: 0.5em;
}
```

After the stylesheet is referenced, the JavaScript files are included. It is important that the script tag for the jQuery library be placed before the tag for our custom scripts; otherwise, the jQuery framework will not be available when our code attempts to reference it.



Throughout the rest of this book, only the relevant portions of HTML and CSS files will be printed. The files in their entirety are available at the book's companion website <http://book.learningjquery.com>.

Now we have a page that looks similar to the following screenshot:



We will use jQuery to apply a new style to the poem text.



This example is to demonstrate a simple use of jQuery. In real-world situations, this type of styling could be performed purely with CSS.

Adding our jQuery code

Our custom code will go in the second, currently empty, JavaScript file which we included from the HTML using `<script src="01.js"></script>`. For this example, we only need three lines of code, as follows:

```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

Finding the poem text

The fundamental operation in jQuery is selecting a part of the document. This is done with the `$()` function. Typically, it takes a string as a parameter, which can contain any CSS selector expression. In this case, we wish to find all of the `<div>` elements in the document that have the `poem-stanza` class applied to them, so the selector is very simple. However, we will cover much more sophisticated options through the course of the book. We will step through many ways of locating parts of a document in *Chapter 2, Selecting Elements*.

When called, the `$()` function returns a new **jQuery object instance**, which is the basic building block we will be working with from now on. This object encapsulates zero or more DOM elements, and allows us to interact with them in many different ways. In this case, we wish to modify the appearance of these parts of the page, and we will accomplish this by changing the classes applied to the poem text.

Injecting the new class

The `.addClass()` method, like most jQuery methods, is named self-descriptively; it applies a CSS class to the part of the page that we have selected. Its only parameter is the name of the class to add. This method, and its counterpart, `.removeClass()`, will allow us to easily observe jQuery in action as we explore the different selector expressions available to us. For now, our example simply adds the `highlight` class, which our stylesheet has defined as italicized text with a gray background and a border.

Note that no iteration is necessary to add the class to all the poem stanzas. As we discussed, jQuery uses **implicit iteration** within methods such as `.addClass()`, so a single function call is all it takes to alter all of the selected parts of the document.

Executing the code

Taken together, `$()` and `.addClass()` are enough for us to accomplish our goal of changing the appearance of the poem text. However, if this line of code is inserted alone in the document header, it will have no effect. JavaScript code is generally run as soon as it is encountered in the browser, and at the time the header is being processed, no HTML is yet present to style. We need to delay the execution of the code until after the DOM is available for our use.

With the `$(document).ready()` construct, jQuery allows us to schedule function calls for firing once the DOM is loaded – without necessarily waiting for images to fully render. While this event scheduling is possible without the aid of jQuery, `$(document).ready()` provides an especially elegant cross-browser solution that:

- Uses the browser's native DOM ready implementations when available and adds a `window.onload` event handler as a safety net

- Allows for multiple calls to `$(document).ready()` and executes them in the order in which they are called
- Executes functions passed to `$(document).ready()` even if they are added after the browser event has already occurred
- Handles the event scheduling asynchronously to allow scripts to delay it if necessary
- Simulates a DOM ready event in some older browsers by repeatedly checking for the existence of a DOM method that typically becomes available at the same time as the DOM

The `.ready()` method's parameter can accept a reference to an already defined function, as shown in the following code snippet:

```
function addHighlightClass() {  
    $('div.poem-stanza').addClass('highlight');  
}  
  
$(document).ready(addHighlightClass);
```

Listing 1.1

However, as demonstrated in the original version of the script, and repeated in Listing 1.2, as follows, the method can also accept an **anonymous function** (sometimes also called a **lambda function**), as follows:

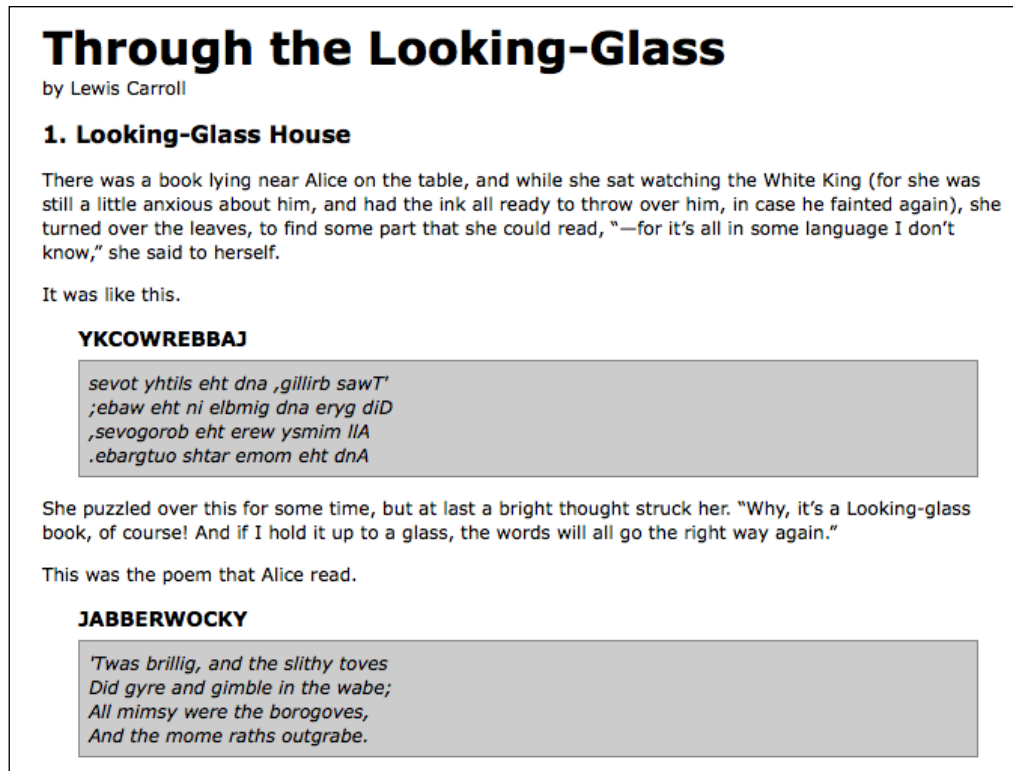
```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

Listing 1.2

This anonymous function idiom is convenient in jQuery code for methods that take a function as an argument when that function isn't reusable. Moreover, the **closure** it creates can be an advanced and powerful tool. However, it may also have unintended consequences and ramifications on memory use, if not dealt with carefully. The topic of closures is discussed fully in *Appendix A, JavaScript Closures*.

The finished product

Now that our JavaScript is in place, the page looks similar to the following screenshot:



The poem stanzas are now italicized and enclosed in boxes, as specified by the `01.css` stylesheet, due to the insertion of the `highlight` class by the JavaScript code.

Plain JavaScript vs. jQuery

Even a task as simple as this can be complicated without jQuery at our disposal. In plain JavaScript, we could add the `highlighted` class as shown in the following code snippet:

```
window.onload = function() {
  var divs = document.getElementsByTagName('div');
  for (var i = 0; i < divs.length; i++) {
    if (hasClass(divs[i], 'poem-stanza')
      && !hasClass(divs[i], 'highlight')) {
      divs[i].className += ' highlight';
    }
  }
}
```



```
    }  
  }  
  
  function hasClass( elem, cls ) {  
    var reClass = new RegExp(' ' + cls + ' ' );  
    return reClass.test(' ' + elem.className + ' ' );  
  }  
};
```

Listing 1.3

Despite its length, this solution does not handle many of the situations that jQuery takes care of for us in Listing 1.2, such as the following:

- Properly respecting other `window.onload` event handlers
- Acting as soon as the DOM is ready
- Optimizing element retrieval and other tasks with modern DOM methods

We can see that our jQuery-driven code is easier to write, simpler to read, and faster to execute than its plain JavaScript equivalent.

Development tools

As this code comparison has shown, jQuery code is typically shorter and clearer than its basic JavaScript equivalent. However, this doesn't mean we will always write code that is free from bugs, or that we will intuitively understand what is happening on our pages at all times. Our jQuery coding experience will be much smoother with the assistance of standard development tools.

High-quality development tools are available in all modern browsers. We can feel free to use the environment that is most comfortable to us. Options include:

- The Internet Explorer Developer Tools:
<http://msdn.microsoft.com/en-us/library/dd565628.aspx>
- The Safari Web Inspector:
<http://developer.apple.com/technologies/safari/developer-tools.html>
- The Chrome Developer Tools:
<http://code.google.com/chrome/devtools/>
- Firebug for Firefox: <http://getfirebug.com/>

Each of these toolkits offers similar development features, including:

- The ability to explore and modify aspects of the DOM
- Investigating the relationship between CSS and its effect on page presentation
- Convenient tracing of script execution through special methods
- Pausing execution of running scripts and inspecting variable values

While the details of these features vary from one browser to the next, the general concepts remain constant. In this book, some examples will require the use of one of these toolkits; we will use Firebug for these demonstrations, but development tools for other browsers are fine alternatives.

Firebug

Up-to-date instructions for installing and using Firebug can be found on the project's home page at <http://getfirebug.com/>. The tool is too involved to explore in great detail here, but a survey of some of the most relevant features will be useful to us.

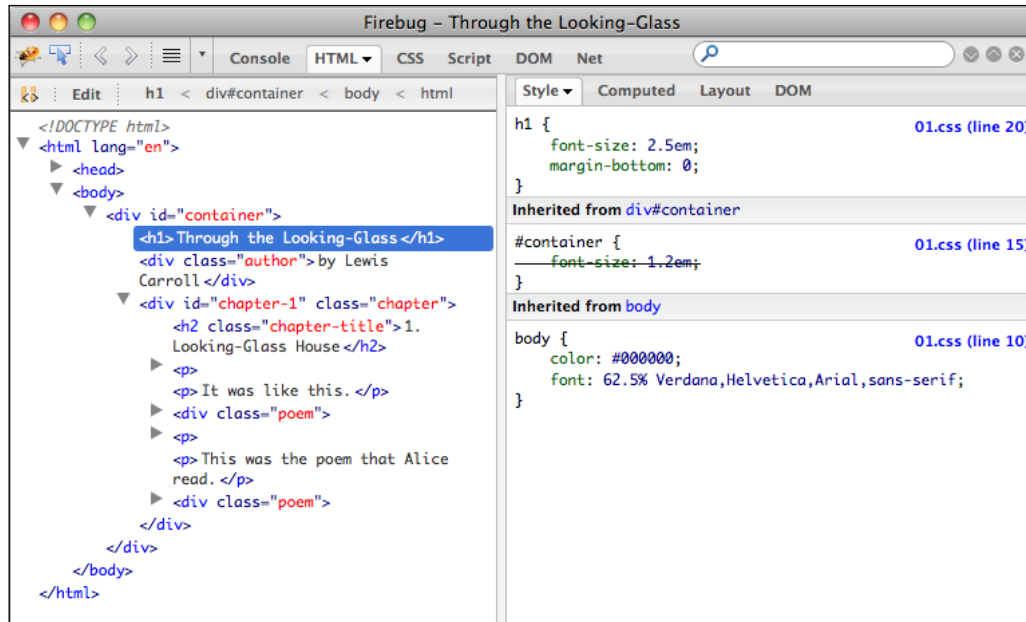


Understanding these screenshots

Firebug is a quickly-evolving project, so the following screenshots may not exactly match your environment. Some of the labels and buttons are provided by the optional **FireQuery** add-on:
<http://firequery.binaryage.com/>.

When Firebug is activated, a new panel appears offering information about the current page.

In the default **HTML** tab of this panel, we can see a representation of the page structure on the left side, and details of the selected element (such as the CSS rules that apply to it) on the right side. This tab is especially useful for investigating the structure of the page and debugging CSS issues, as shown in the following screenshot:



The **Script** tab allows us to view the contents of all loaded scripts on the page, as shown in the preceding screenshot. By clicking on a line number, we can set a breakpoint; when the script reaches a line with a breakpoint, it will pause until we resume execution with a button click. On the right side of the page, we can enter a list of variables and expressions we wish to know the value of at any time.

