

LEARNING **jQ**uery

A Hands-on Guide to Building Rich Interactive Web Frontends



RALPH STEYER

FREE SAMPLE CHAPTER

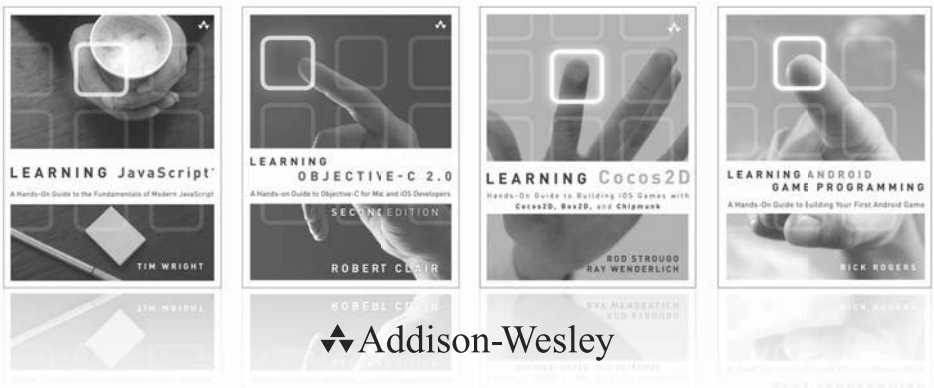


SHARE WITH OTHERS

Learning jQuery

A Hands-on Guide to Building
Rich Interactive Web Front Ends

Addison-Wesley Learning Series



Visit informit.com/learningseries for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

 Addison-Wesley

 **informit**
The online technology learning source

 **Safari**
Books Online

Learning jQuery

A Hands-on Guide to Building Rich
Interactive Web Front Ends

Ralph Steyer

◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Learning jQuery: A Hands-on Guide to Building Rich Interactive Web Front Ends

Copyright © 2013 by Pearson Education, Inc.

First published in the German language under the title jQuery by Addison-Wesley, an imprint of Pearson Education Deutschland GmbH, München. Copyright © 2011 by Pearson Education Deutschland GmbH.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-321-81526-2

ISBN-10: 0-321-81526-2

Library of Congress Cataloging-in-Publication Data is on file.

Printed in the United States of America

First Printing May 2013

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Pearson 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

Pearson 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

Acquisitions Editor

Mark Taber

Managing Editor

Sandra Schroeder

Project Editor

Mandie Frank

Copy Editor

Keith Cline

Indexer

Larry Sweazy

Proofreader

Megan Wade

Translator

Almut Dworak

Technical Editor

Brad Dayley

Publishing

Coordinator

Vanessa Evans

Designer

Chuti Prasertsith

Composer

Jake McFarland

Contents at a Glance

- 1 Introduction 1
- 2 First Examples with jQuery 17
- 3 Basic Knowledge 31
- 4 How jQuery Works 51
- 5 Selectors and Filters 83
- 6 Access to the Elements of a Web Page 131
- 7 Formatting with Style Sheets Under jQuery 205
- 8 Event Handling Under jQuery 247
- 9 Effects and Animations 279
- 10 AJAX 297
- 11 jQuery UI 345
- 12 Plug-Ins 393
- 13 jQuery Mobile 417
- Appendix 457
- Index 467

Table of Contents

1 Introduction 1

- 1.1 What Is This Book About? 2
 - 1.1.1 What You Can Learn from This Book 4
- 1.2 Writing Conventions 5
- 1.3 Who Is the Target Audience for This Book? 6
- 1.4 What Do You Need? 6
 - 1.4.1 Hardware and Operating System 6
 - 1.4.2 jQuery and jQuery UI 7
 - 1.4.3 The Browsers 9
 - 1.4.4 Different Operating Systems and Virtual Machines for Testing 10
 - 1.4.5 The Web Server for Realistic Testing 11
 - 1.4.6 The Development Tools 13
- 1.5 About the Author 16

2 First Examples with jQuery 17

- 2.1 Accessing Elements and Protecting the DOM 17
- 2.2 Editing the Web Page with DHTML à la jQuery 22
- 2.3 Animatedly Reducing and Enlarging of an Element 25
- 2.4 Changing Attributes Dynamically 28

3 Basic Knowledge 31

- 3.1 The Web, Web 2.0, and the Client/Server Principle on the Internet 32
 - 3.1.1 Programming on the Web 32
 - 3.1.2 The Web 2.0 33
- 3.2 JavaScript and Its Relationship to jQuery 33
 - 3.2.1 The General Integration of JavaScript in Websites 34
- 3.3 AJAX and XMLHttpRequest (XHR) 37
 - XML 38
 - JSON 41
 - More Details on Processing JSON for JavaScript Pros 43
- 3.4 DOM and Objects 46
- 3.5 Style Sheets and DHTML 48
 - 3.5.1 CSS: The Web's Standard Language 48
 - 3.5.2 The Specific Syntax of CSS Declarations 50
 - 3.5.3 Selectors 50

4 How jQuery Works 51

- 4.1 Accessing Elements of the Web Page 52
- 4.2 The jQuery Namespace and the jQuery Object 54
- 4.3 Special Data Types and Structures in jQuery 55
 - 4.3.1 Options 55
 - 4.3.2 Map 56
 - 4.3.3 The Array<Type> Notation 56
 - 4.3.4 jqXHR 57
- 4.4 The Function `jQuery()` and the Alias `$()` 57
 - 4.4.1 The Context 59
- 4.5 Executing Functions After DOM Has Been Built 60
 - 4.5.1 Callback or Anonymous Function as a Parameter of `jQuery()` 60
 - 4.5.2 Placing `document.ready()` into an External JavaScript File 63
 - 4.5.3 Example of Creating a Basic Structure for a Modularized jQuery Web Application 63
- 4.6 Creating an Element with `jQuery()` and Inserting It into the Web Page 66
 - 4.6.1 Options for Initializing Attributes 68
- 4.7 Wrapping Existing Elements with `jQuery()` 70
 - 4.7.1 Direct Access to DOM elements via `get()` 71
- 4.8 Using jQuery in Combination with Other Frameworks 72
 - 4.8.1 The Function `noConflict()` 73
- 4.9 More About Context 74
 - 4.9.1 `context`, `selector`, and `nodeName` 75
- 4.10 Chaining jQuery Objects 77
 - 4.10.1 Executing Function Calls Sequentially: The jQuery Queue 78
- 4.11 New Core Techniques Since Version 1.5 78
 - 4.11.1 `jQuery.sub()` 78
 - 4.11.2 `jQuery.when()` 79
 - 4.11.3 Version 1.6: What's New? 79
 - `attr()`, `prop()`, and `removeProp()` 80
 - `data()` 81

5 Selectors and Filters 83

- 5.1 The Basics 84
 - 5.1.1 What Is a Selector? 84
 - 5.1.2 What Are Filters? 84
 - 5.1.3 XPath as Basis 85

- 5.2 The Basic Selectors and the Hierarchical Selectors 86
 - 5.2.1 Examples 88
 - 5.2.2 Potential Pitfalls 97
- 5.3 Filtering Selectors 99
 - 5.3.1 Basic Filters 99
 - 5.3.2 Content Filters 106
 - 5.3.3 Visibility Filters 109
 - 5.3.4 Child Filters 112
 - 5.3.5 Attribute Filters 114
 - 5.3.6 Filters for Form Elements and Form Filters 118
- 5.4 Filter Methods 123
 - 5.4.1 `eq()` 123
 - 5.4.2 `not()` 123
 - 5.4.3 `first()` and `last()` 124
 - 5.4.4 `slice()` 124
 - 5.4.5 `filter()` 125
 - 5.4.6 `is()` 126
 - 5.4.7 `map()` 127

6 Accessing the Elements of a Web Page 131

- 6.1 General Info on Checking, Changing, Adding, and Removing Nodes 131
- 6.2 Checking and Changing Node Contents: `html()` and `text()` 132
- 6.3 Content of Form Fields: `val()` 135
- 6.4 Accessing Attributes via `attr()` 137
- 6.5 Inserting Nodes into a Web Page 137
 - 6.5.1 `append()` and `prepend()` 138
 - 6.5.2 `appendTo()` and `prependTo()` 143
- 6.6 Inserting Nodes Before or After 148
 - 6.6.1 `after()` and `before()` 149
 - 6.6.2 `insertAfter()` and `insertBefore()` 152
- 6.7 Wrapping 154
 - 6.7.1 Wrapping Individually with `wrap()` 154
 - 6.7.2 Wrapping All with `wrapAll()` 156
 - 6.7.3 Wrapping Inner Areas with `wrapInner()` 158
 - 6.7.4 Unwrapping with `unwrap()` 159

- 6.8 Replacing with `replaceWith()` and `replaceAll()` 159
 - 6.8.1 Replacing with `replaceWith()` 160
 - 6.8.2 Replacing All with `replaceAll()` 164
 - 6.9 Removing with `empty()` and `remove()/detach()` plus `removeAttr()` 166
 - 6.9.1 The Alternative of `remove():detach()` 171
 - 6.9.2 Deleting Attributes 171
 - 6.10 Cloning with `clone()` 172
 - 6.11 Search and Find 176
 - 6.11.1 Of Children and Parents: `children()` and `parent()` plus `parents()/parentsUntil()` 176
 - 6.11.2 `offsetParent()` and `closest()` 180
 - 6.11.3 Siblings 182
 - 6.11.4 Searching Descendants with `has()` 184
 - 6.12 Finding with `find()` and `contents()` 184
 - 6.13 The jQuery Method `each()` for Iterating over Arrays and Objects 186
 - 6.13.1 `jQuery.each()` 188
 - 6.13.2 The Method `each()` 192
 - 6.14 The `add()` Method 193
 - 6.14.1 The `end()` and `andSelf()` Methods 195
 - 6.15 A More Comprehensive Example at the End: A Date Component 196
- 7 Formatting with Style Sheets Under jQuery 205**
- 7.1 The `css()` Method 206
 - 7.1.1 Getting Style Properties 206
 - 7.1.2 Setting Properties 207
 - 7.2 Changing Classes of Elements 209
 - 7.2.1 Adding Classes: `addClass()` 210
 - 7.2.2 Removing Classes: `removeClass()` 218
 - 7.2.3 Toggling Classes with `toggleClass()` 219
 - 7.2.4 Testing for a Class: `hasClass()` 221
 - 7.3 Positioning Methods 223
 - 7.3.1 Determining the Position with `position()` 224
 - 7.3.2 Position in Relation to the Document: `offset()` 228
 - 7.4 Scrolling Methods 236

- 7.5 Height and Width 239
 - 7.5.1 `height()` and `width()` 239
- 7.6 Inner and Outer Dimensions 242

8 Event Handling Under jQuery 247

- 8.1 Basic Information on Events, Event Handlers, Triggers, and Data Binding 247
 - 8.1.1 Events 247
 - 8.1.2 General Information on Event Handlers 248
 - 8.1.3 HTML Event Handlers 248
 - 8.1.4 JavaScript Event Handler 249
 - 8.1.5 The Event Object 250
 - 8.1.6 Bubbling 251
 - 8.1.7 Data Binding 251
 - 8.1.8 Trigger 252
- 8.2 The Event Object in jQuery 252
 - 8.2.1 The Constructor of `jQuery.Event` 252
 - 8.2.2 The Properties of the Event Object `jQuery.Event` 253
 - 8.2.3 The Methods of an Object of the Type `jQuery.Event` 256
- 8.3 Ready, Steady, Go: `$(document).ready()` 258
- 8.4 Event Helpers 258
- 8.5 Expanded Methods for Event Handling 262
 - 8.5.1 The `bind()` and `unbind()` Methods 262
 - 8.5.2 The One and Only: `one()` 266
 - 8.5.3 The Method `trigger()` 267
 - 8.5.4 `triggerHandler()` 269
 - 8.5.5 Live Events: The `live()` and `die()` Methods plus `delegate()` and `undelegate()` 270
 - 8.5.6 Auxiliary Functions for Interaction 274

9 Effects and Animations 279

- 9.1 Basic Use 279
 - 9.1.1 Speed Is All You Need 279
 - 9.1.2 Specifying a Callback 280
 - 9.1.3 Chaining 281
 - 9.1.4 Queues 281
 - 9.1.5 Stopping via `stop()` and `jQuery.fx.off` 282

- 9.1.6 Endless Animations 282
 - 9.1.7 Types of Animation 282
 - 9.2 Showing and Hiding: The `show()` and `hide()` Methods 283
 - 9.3 Sliding Effects: `slideDown()`, `slideUp()`, and `slideToggle()` 284
 - 9.4 Opacity Effects: `fadeIn()`, `fadeOut()`, and `fadeTo()`
(Plus `toggle()`) 287
 - 9.5 Individual Animations with `animate()` 289
- 10 AJAX 297**
- 10.1 AJAX and XMLHttpRequest (XHR) Basics 297
 - 10.1.1 Creating an XMLHttpRequest Object Manually 298
 - 10.1.2 The Methods of an XHR Object 299
 - 10.1.3 The Properties of an XHR Object 300
 - 10.1.4 A Practical Example of Data Request Without Special jQuery Methods 300
 - 10.1.5 The Data Format in an AJAX Communication 302
 - 10.1.6 AJAX Request Process 303
 - 10.2 Special AJAX Support in jQuery 304
 - 10.2.1 JSONP and Remote Requests 304
 - 10.2.2 The `jqXHR` Object 305
 - 10.2.3 Methods in jQuery for AJAX Requests 305
 - 10.2.4 Specifying the Data Type 305
 - 10.2.5 Avoiding Caching 307
 - 10.3 `$.get()` and `$.post()` 307
 - 10.3.1 Just Requesting Plain Text from the Web Server 307
 - 10.3.2 Sending Data to the Web Server via `$.get()` and `$.post()` 309
 - 10.3.3 Getting and Parsing XML Data 312
 - 10.4 Getting and Parsing JSON Data: `getJSON()` and `parseJSON()` 316
 - 10.4.1 A Simple Application with JSON 316
 - 10.4.2 Requesting Twitter Tweets via JSONP 317
 - 10.5 Loading a Script Later via AJAX:
`jQuery.getScript()` 320
 - 10.6 The General Variation for Loading Data: `load()` 322
 - 10.6.1 Specifying Filters 323
 - 10.7 Serializing Data 327
 - 10.7.1 The `serialize()` Method 327
 - 10.7.2 The `serializeArray()` Method 329
 - 10.7.3 The General Version: `param()` 329

10.8	Default Values for AJAX	330
10.9	AJAX Events and AJAX Event Handlers	330
10.9.1	Local Events	330
10.9.2	Global Events	332
10.10	Complete Control	333
10.10.1	jQuery.ajax()	333
10.10.2	A JSONP Request	339
10.10.3	Loading and Executing a JavaScript File	340
10.10.4	Sending Data Plus Evaluating the Success	340
10.10.5	Extended Techniques for \$.ajax()	341
11	jQuery UI	345
11.1	What Is jQuery UI?	345
11.1.1	Components for Supporting Interaction	346
11.1.2	Widgets	346
11.1.3	Extended Effects	347
11.1.4	The Theme Framework and ThemeRoller	347
11.2	Getting Started	348
11.3	How Is jQuery UI Used?	349
11.3.1	Downloading and ThemeRoller	349
11.3.2	Using jQuery UI on a Web Page	353
11.3.3	A Sample Web Page for jQuery UI	355
11.4	Using the Components in jQuery UI	355
11.4.1	The Default Setting	356
11.4.2	Some Basic Rules on Components and Widgets	358
11.4.3	Properties/Options of Components	359
11.4.4	Methods of Components	363
11.4.5	Events in Components and Widgets	366
11.5	An Overview of the Components and Widgets	370
11.5.1	The Interaction Components	370
11.5.2	The Widgets	372
11.5.3	Utilities	385
11.6	Effects	385
11.6.1	The effect() Method	385
11.6.2	Color Animations with animate()	386
11.7	A Complete Website Based on jQuery UI	387

12 Plug-Ins 393

- 12.1 The jQuery Plug-In Page 393
 - 12.1.1 Searching For and Using an Existing Plug-In 394
 - 12.1.2 Validation Plug-Ins 397
- 12.2 Creating Custom Plug-Ins 405
 - 12.2.1 Why Create Custom Plug-Ins? 405
 - 12.2.2 Creating Your First Plug-In 405
 - 12.2.3 The Main Rules for Creating a Simple Plug-In 409
 - 12.2.4 Rules for Creating More Complex Plug-Ins 409
 - 12.2.5 An Example for a Plug-In with Options 411
 - 12.2.6 Another Example for a Plug-In with Options 413
- 12.3 Publishing a Plug-In 415

13 jQuery Mobile 417

- 13.1 Basics 417
 - 13.1.1 The Platforms 419
 - 13.1.2 Downloading and Integrating the Framework 420
 - 13.1.3 Alternatives 421
- 13.2 The Role System and `data-role` 422
- 13.3 The Basic Structure of a Mobile Web Page 422
- 13.4 Linking Pages 424
 - 13.4.1 External Links via Hijax 424
 - 13.4.2 Internal Links and the Special Interpretation of a Page 425
- 13.5 The Transitions 428
- 13.6 Dialogs 428
- 13.7 Buttons 429
 - 13.7.1 Buttons with Icons 430
 - 13.7.2 Block Element or Inline Element 431
 - 13.7.3 Grouping 431
 - 13.7.4 A Practical Example 432
- 13.8 Toolbars and Navigation Bars 435
- 13.9 Lists 439
- 13.10 Form Elements 443
 - 13.10.1 Field Containers 444
 - 13.10.2 The Various Form Elements 444
 - 13.10.3 Plug-In Methods for Form Elements 447
 - 13.10.4 Sending the Form Data 448

- 13.11 Special Events 448
 - 13.11.1 Touch Events 448
 - 13.11.2 Orientation Change 448
 - 13.11.3 Scroll Events 449
 - 13.11.4 Page Events 449
- 13.12 The Theme Framework and General Content Design 452
- 13.13 Collapsed and Expanded Content 454

Appendix 457

- A.1 Overview of Basic Information on JavaScript 457
 - A.1.1 Case Sensitivity 457
 - A.1.2 Variables, Literals, and Data Types 457
 - A.1.3 Functions and Methods 459
 - A.1.4 Objects in JavaScript 461
 - A.1.5 Arrays 463
- A.2 Available DOM Objects 465

Index 467

About the Author

Ralph Steyer is a computer programmer, consultant, journalist, and book author with decades of experience in a wide variety of computer programming languages and technologies. He has a degree in mathematics from Frankfurt/Main University and is the author of several books on web programming, including *JavaScript Handbook* and *AJAX Frameworks* (Addison-Wesley).

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 directly to let us 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 we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we 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.

Email: feedback@developers-library.info
Mail: Reader Feedback
Addison-Wesley Developer's Library
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

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

Your purchase of this book includes access to a free online edition for 45 days through the Safari Books Online subscription service. Details are on the last page of this book.

This page intentionally left blank

First Examples with jQuery

In this chapter, we make first contact with jQuery without any further preparations. In other words, we are jumping right into the deep end. I am anxious for you to get a feeling for what you can do with jQuery and what you can get out of this framework. Just accept for now that many questions regarding the source text have to remain open at this stage. Don't worry, though; these questions are answered over the next few chapters. The explanations on the listings also remain somewhat superficial at this stage, to avoid going off topic. We want to get into the practical application of jQuery as quickly as possible and just have some fun playing around, which means creating examples.

Note

For the examples in this chapter, but also most examples in the following chapters, it is not relevant which specific version of jQuery you are using. The examples in this book have been created with jQuery 1.8.2 or later, but often any version from 1.3 or at least 1.4.1 onward is sufficient.

2.1 Accessing Elements and Protecting the DOM

If you already have some basic knowledge of programming on the Web,¹ you already know that you can access the components of a web page via JavaScript or another script language in the browser via an object model with the name Document Object Model (DOM). For this type of access, there are several standard techniques,² each of which has its own weaknesses.

1. Given the target audience of this book, I assume you do.
2. For example, the methods `getElementById()` and `getElementsByName()` plus access via object fields or names.

In particular, you usually have to enter many characters when accessing just a single element of the web page (or a group). This involves a lot of effort and is susceptible to errors. Most frameworks therefore offer a system via which this access can take place with an abbreviated, unified approach. Plus the underlying mechanisms compensate for various weaknesses of the standard access methods, above all by compensating for browser-dependent particularities and supplementing various missing functions of the pure DOM concept. Particularly important is that this compensation has generally been tested on all officially supported browsers and therefore works rather reliably.

The following example demonstrates another extremely important function of jQuery—protecting the DOM. More on what this is all about later. For now, let's just say that different browsers process the web page differently on loading (parsing) the page, which can lead to a number of problems when the elements of the web page are accessed (especially if you try to access the elements of the web page too soon in a script—in other words, before the browser has correctly constructed the DOM). Here, jQuery offers a reliably method for mastering these problems.

The example also shows you in passing, as it were, how you can use jQuery as a standardized way of accessing contents of elements with text and reacting to events. But enough introduction. Here is our very first listing (ch2_1.html):³

Listing 2.1 The First jQuery Example

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 01
    Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04   <meta http-equiv="Content-Type"
05     content="text/html; charset=utf-8" />
06   <title>The first jQuery example</title>
07   <script type="text/javascript"
08     src="lib/jquery-1.8.min.js"></script>
09   <script type="text/javascript">
10     $(document).ready(function(){
11       $("#a").click(function(){
12         $("#output").html("Boring :-(");
13       });
14       $("#b").click(function(){
15         $("#output").html("A nice game :-)");
16       });
17       $("#c").click(function(){
```

3. The quotations are from the movie *War Games*—one of the first movies about a hacker. I highly recommend that you watch it the next time it is on TV.

```
18     $("#output").html("A strange game. " +
19         "The only winning move " +
20         "is not to play.");
21     });
22     });
23 </script>
24 </head>
25 <body>
26 <h1>Welcome to WOPR</h1>
27 <h3>Shall we play a game</h3>
28 <button id="a">Tic Tac Toe</button>
29 <button id="b">Chess</button>
30 <button id="c">
31     Worldwide Thermonuclear War</button>
32 <div id="output"></div>
33 </body>
34 </html>
```

Just create the HTML file in a separate directory and save it under the listed name.

In practice, you would usually save all your resources that are part of a project within a separate directory. For a web project, the best solution is to create these directories in the shared folder of your web server. In the case of Apache/XAMPP, this is usually the directory `htdocs`. This has the advantage that—if the web server is running—you can run the test directly via HTTP and a proper web call, not just load the file via the FILE protocol into the browser (in other words, the classic opening as file or simply dragging the file into the browser). The latter is not a realistic, practice-related test because later the pages also have to be requested by the visitor via a web server.

If you are working with an integrated development environment (IDE) such as Aptana or the Visual Studio Web Developer, you can usually display a web page directly from the IDE via an integrated web server. In Aptana, this is done via the **Run** command, and in Web Developer (a Firefox add-on) you can use the shortcut **Ctrl+F5**.

Note

In this book, all examples are sorted by chapter and listed accordingly on the companion website (<http://jquery.safety-first-rock.de>).

In lines 7 and 8, you see the reference to an external JavaScript file—the jQuery library that in this specific case resides in the subdirectory `lib` of the project directory where the website is saved. This structure has now become widely accepted in practice. This means that the jQuery library also has to be located in exactly that place. But, of course, you can instead choose to use a different path structure.

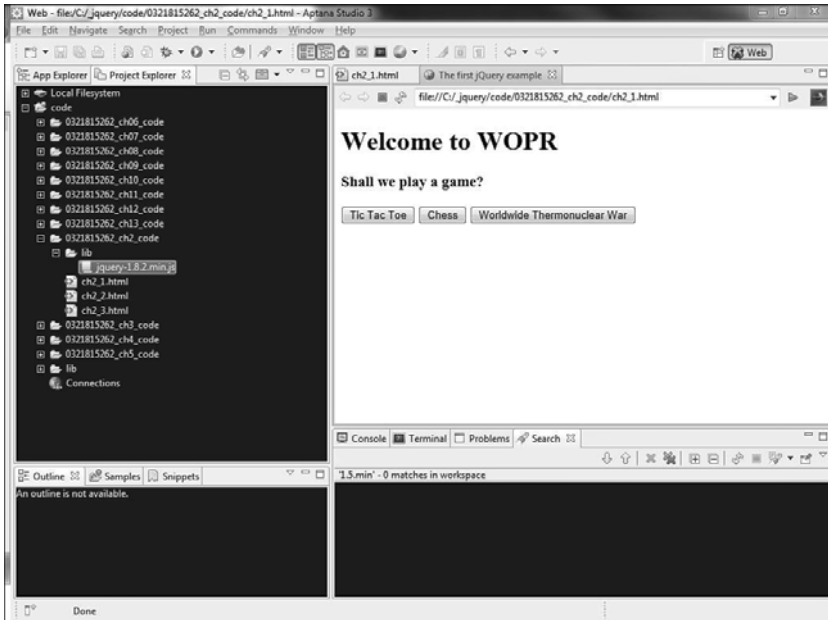


Figure 2.1 In this project, the jQuery library is located in the directory `lib`, seen from the perspective of the website.

Line 9 to 23 contains a normal JavaScript container. In it, the web page is addressed with `$(document)` (line 10). The function `$()` is a shorthand notation for referencing an element of the web page. You also see these shortened access notations in lines 11, 12, 14, 15, 17, and 18. But here, an element ID is used as a parameter.

Note

Note that an element (in terms of jQuery) as a parameter of `$()` is not enclosed in quotation marks, whereas an ID (or another selector) is enclosed in quotation marks.

Let's now take a quick look at the method `ready()` that starts in line 10 and goes up to line 22. This method ensures that the calls it contains are only executed when the web page has been fully loaded and the DOM is correctly constructed. As hinted at before and without going into too much detail, this is already a feature whose value cannot be appreciated highly enough.

Note

For readers with the corresponding knowledge and experience, the method `ready()` is an alternative for the event handler `onload` that you can write in HTML in the body of a web page or under JavaScript for the corresponding DOM object. But this event handler is seen as extremely *unreliable* because it is insufficiently implemented in various browsers. It is a good idea to avoid it wherever possible.

Within the `ready()` method, three event handlers each specify the reaction when clicking the listed elements. In our examples, these are three buttons marked with a unique ID.

Note

The method `click()` encapsulates (you probably guessed it) the function call of the event handler `onclick`.

The allocation to the correct function is achieved via the ID and triggering the function within the method `click()`. Note that we are using an anonymous function here (without an identifier).

It also gets interesting if a user clicks one of the buttons. This displays a specific text output in a section of the web page. We are again using `$()` and an ID for selecting the section (a `div` block) and the method `html()` for accessing the content.



Figure 2.2 The web page with the three buttons; the user has just clicked the third button.

Note

In all following examples, we omit writing or using the `DOCTYPE` statement. For the sake of completeness, it does belong there, but omitting it does not have any effect for us, and because it is always the same, writing it down over and over again is just a waste of space in this book. In the examples on the companion website, the statement is included because it forms part of the correct standard.

2.2 Editing the Web Page with DHTML à la jQuery

Generally, you can design the visual appearance of a web page much better and more effectively with style sheets than with pure HTML. In particular, they make it easier to separate layout and structure of the site. These statements are probably old hat to you, as true as they are.

If you now change the style sheets of a site dynamically via JavaScript, we are talking about Dynamic Hypertext Markup Language (DHTML). But animation effects such as showing and hiding parts of a web page via other JavaScript techniques also form part of this. In the following example, we look at how you can carry out animated web page changes with jQuery quickly, simply, conveniently, and yet reliably in the various browsers. In this example, we change the Cascading Style Sheets (CSS) class of an element dynamically.

First, let's look at a little CSS file that should be integrated into the following web page and saved in the lib directory (ch2_2.css):

Listing 2.2 The File with the External Style Sheets

```
01 body {
02  background: lightgray;color: blue;
03 }
04 div {
05  background: white;font-size: 14px;
06 }
07 .mClass {
08  background: red; color: yellow; font-size: 24px;
09 }
```

Nothing much happens in the CSS file. It determines the background and foreground color of the entire web page and all elements of the type `div`, plus the font size for all elements of the type `div`.

Of primary interest is the class described in lines 7–9. It is not yet to be used on loading the following web page, but is to be assigned dynamically in case of a user action (ch2_2.html):

Listing 2.3 Changing the CSS Class

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03 <meta http-equiv="Content-Type"
04   content="text/html; charset=utf-8" />
05 <title>The second jQuery example</title>
06 <link type="text/css" rel="stylesheet"
07   href="lib/ch2_2.css" />
08 <script type="text/javascript"
09   src="lib/jquery-1.8.2.min.js"></script>
```

```

10 <script type="text/javascript">
11   $(document).ready(function(){
12     $("#a").click(function(){
13       $("#c").addClass("mClass");
14     });
15     $("#b").click(function(){
16       $("#c").removeClass("mClass");
17     });
18   });
19 </script>
20 </style>
21 </head>
22 <body>
23 <h1>Editing Style Sheets with jQuery</h1>
24 <button id="a">Add CSS class</button>
25 <button id="b">Remove CSS class</button><hr/>
26 <div id="c">He who knows all the answers
27   has not been asked all the questions.
28 </div><hr/>
29 <div id="c">Be not afraid of going slowly,
30   be afraid only of standing still.</div>
31 </body>
32 </html>

```

In the example, you can see two buttons below a heading and two texts within a `div` section that is separated by a separator in each case. This is pure HTML. Plus in lines 6 and 7 you can see the link to the CSS file.



Figure 2.3 The site after loading

But now we want to use jQuery to manipulate the text below the buttons or the first `div` container. That is why the `div` container has an ID. The text below it is intended for comparison purposes.

For accessing the elements of the web page, the example uses jQuery mechanisms already mentioned in the first example. To react to the relevant click on a button, we again use the method `click()`. So far, there is nothing new.

Now you should notice that we do not yet assign the CSS class from the linked CSS file to an element on loading the web page. But take a look at line 13.

Listing 2.4 Adding a CSS Class

```
$("#c").addClass("mClass");
```

As the name of the method `addClass()` already implies, calling this method assigns the indicated style sheet class to the preceding element. This happens dynamically without the web page having to be reloaded in any way. The function is triggered when the user clicks the corresponding button, as you can see from the surrounding `click()` method.

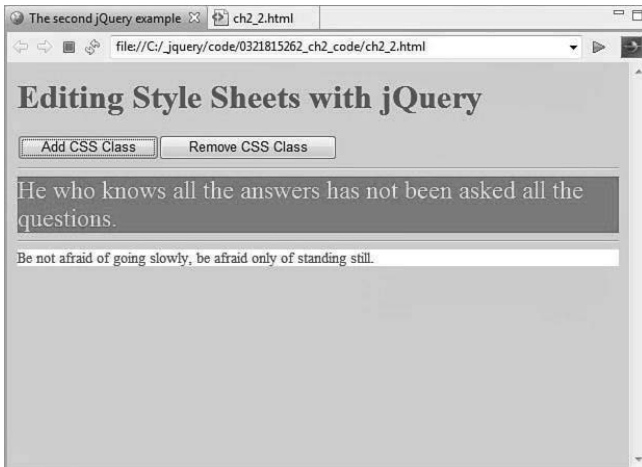


Figure 2.4 The CSS class has been assigned.

In line 16, you can see how the class is removed again following the same pattern. This time, we use the method `removeClass()`. If you test the example, you will see that font and background are changed accordingly.

Tip

Alternatively, you could use the method `toggleClass()` in this example. This removes or adds a CSS class, always depending on the state. If the class is already assigned, it is then removed, and vice versa.

2.3 Animateably Reducing and Enlarging of an Element

Now we want to use jQuery to animateably reduce and enlarge an element to hide or show it. First, let's look at the external CSS file in the subdirectory `lib`, in which a property is defined that has specific consequences for the following animation (`ch2_3.css`):

Listing 2.5 The CSS File

```
01 body {
02  background: lightgray;color: blue;
03 }
04 #i1 {
05  width:300px; height:225px;
06 }
07 #i2 {
08  height:225px;
09 }
10 #h2{
11  background: white; color:#0000FF; font-size: 18px;
12 }
```

The specification that is interesting for the following example is the width data in line 5. The ID used as selector references an image. The width specification influences the type of the animation that follows. But first, let's look at the web page itself. It basically contains two images and some text below. We want to animate all three elements (`ch2_3.html`):

Listing 2.6 Reducing or Enlarging Two Images and Some Text

```
...
06 <link type="text/css" rel="stylesheet"
07   href="lib/ch2_3.css" />
08 <script type="text/javascript"
09   src="lib/jquery-1.8.min.js"></script>
10 <script type="text/javascript">
11   $(document).ready(function(){
12     $("#toggle1").click(function(event){
13       $('#i1').slideToggle('slow');
14     });
15     $("#toggle2").click(function(event){
16       $('#i2').slideToggle('slow');
17     });
```

```

18     $('#toggle3').click(function(event){
19         $('#h2').slideToggle('slow');
20     });
21 });
22 </script>
23 </head>
24 <body>
25     <h1>Animated showing and hiding
26     of an image and text with jQuery</h1>
27     <button id="toggle1">Toggle Image 1</button>
28     <button id="toggle2">Toggle Image 2</button>
29     <button id="toggle3">Toggle Text</button><hr/>
30     
31     <hr/>
32     <h2 id="h2">A ski jump</h2>
33 </body>
34 </html>

```

At the core of this animation is the method `slideToggle()`. This name is also very telling. You can use this effect to show or hide objects depending on the current state, or to reduce or enlarge them. In other words, the current state is toggled to the opposite state. You can see it applied in lines 13, 16, and 19.

Tip

As you can probably see, a temporal interval is specified as a parameter. It determines how long the animation should take. You can pass such parameters for the speed in most animations in jQuery. Permitted parameters are `slow`, `normal`, `fast`, or a specification of time in milliseconds. The specification in milliseconds is not enclosed by quotation marks.



Figure 2.5 The original looks like this.

If you reconstruct the animation of the first image, you will see that reducing the image results in a reduction in image height and the image then disappears altogether. Vice versa, the image grows upward from that point if you enlarge it.

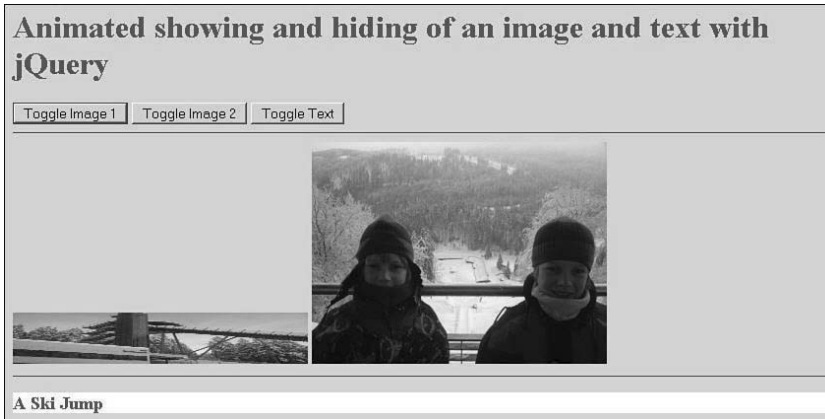


Figure 2.6 Here, the first image is squashed down.

Of massive importance for this behavior is that the width of this image is specified via the CSS rule for the ID `i1`. This prevents the width from also being reduced. The animation of the second image whose width is not specified shows what that looks like. You will see that on reducing the image, it shrinks into the lower-left corner of the image and then disappears altogether. Vice versa, the image grows outward from this point to the top right.

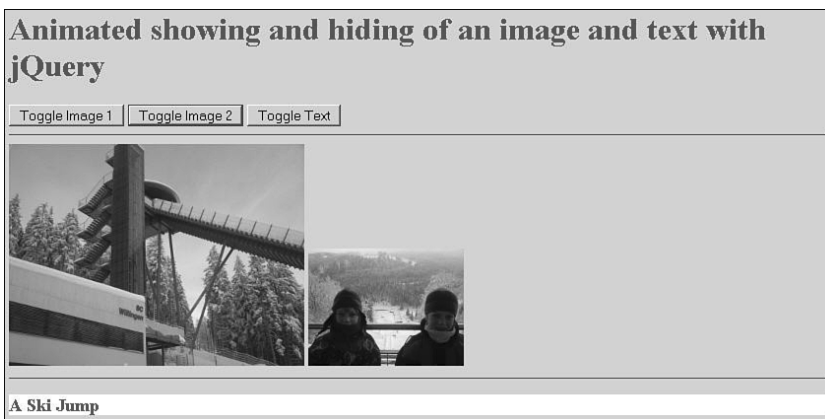


Figure 2.7 The second image shrinks animatedly in width and height.

But now observe what happens to the text if you click the third button. The heading also disappears but only in terms of height.

For the effect of `slideToggle()`, it matters to what type of element the animation technique is applied, and the CSS rules that have been previously applied to an element also play a role.

The animations in the example are basically independent from one another. If you set the interval for running the animation long enough, you can have the animations run in parallel.

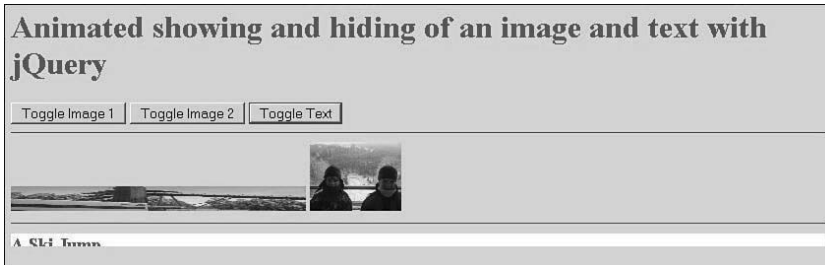


Figure 2.8 The three elements are animated in parallel.

In that case, note that the content positioned lower down is moved upward or could be repositioned vertically if a preceding element has disappeared completely. (In effect, it is removed from the text flow.)

But what happens in the example if you click the same button several times? This answer might surprise you: The events are cumulated. This means they are executed consecutively; the next event is executed only when the previous one has been fully processed. So, clicking the button again does not cause the current animation to stop and the next one to commence immediately. If that is what you want to achieve, you have to program it explicitly.

2.4 Changing Attributes Dynamically

This section shows how you can dynamically change attributes for an element of the web page. To this purpose, jQuery offers the extremely flexible and useful method `attr()`. With this, you can dynamically change one or several attributes of an element. In curly brackets, you set a value pair as parameter, first specifying the attribute, followed by a colon and then a string with the new value. Alternatively, you can also specify two string parameters. In this variation, the first parameter represents the attribute name and the second parameter the value. (In this case, you can change only one attribute.) If you only want to request the value of an attribute, you just enter the name of the attribute as a string parameter.

Note

For the sake of simplicity, we change only one attribute in the following example. If you want to change several attributes at once, however, you just need to write additional value pairs in the curly brackets, separated by commas.

For our example, we want to replace an image in the web page by changing the value of the attribute `src` of an `` tag (`ch2_4.html`).

Listing 2.7 Changing Attributes on an Element

```

...
06  <script type="text/javascript"
07      src="lib/jquery-1.8.min.js"></script>
08  <script type="text/javascript">
09      $(document).ready(function(){
10          $("#toggle1").click(function(){
11              $("img").attr({
12                  src: "images/i1.jpg"
13              });
14          });
15          $("#toggle2").click(function(){
16              $("img").attr(
17                  "src", "images/i2.jpg"
18              );
19          });
20      });
21  </script>
22 </head>
23 <body>
24  <h1>Replacing an image</h1>
25  <button id="toggle1">Image 1</button>
26  <button id="toggle2">Image 2</button>
27  <hr/>
28 </html>

```

We change the value via the notation in the curly brackets once, and via the two string parameters once. As mentioned earlier, we replace the value `src` in each case.

Summary

This chapter provided just a few examples, but they already serve well to demonstrate central key facts of jQuery. In particular, you should memorize the function `$()` and the method `ready()`. But techniques for specifying reactions, such as the method `click()`, are also of elementary importance. And animation techniques such as `addClass()`, `toggleClass()`, `removeClass()`, and `slideToggle()` are also going to be helpful in practice later for DHTML effects. In this chapter, you also learned about changing attribute values (`attr()`). You will more fully understand these techniques as you work through other chapters of this book and have delved deeper into the overall concept of jQuery.

Index

SYMBOLS

" (quotation marks), 42, 56
\$.get method, 307-315
\$.post method, 307-315
\$(document).ready() method, 258
() (parentheses), 459
* selector, 88
: (colon), 42
[] (square brackets), 42
\ (backslash), 85
{ } (curly brackets), 42

A

abort() method, 299
accept() method, 403
accessing
 arrays, elements, 463
 attr() method, 137
 DOM, 71-72, 83
 elements, 17, 47
 height/width of, 240
 modifying nodes, 132-135
 web pages, 52-54, 131
 form fields, 135
 Mobile, 417
 objects, 186, 462
 verifying, 310-312
Accordion widget, 346, 372, 390
addClass() method, 210-217

add() method, 193-196**adding**

- classes, 24
- content, 149
- nodes, 131
- plug-ins, 415
- symbols to buttons, 430

add-ons (Firebug), 84**after() method, 149-151****AJAX (Asynchronous JavaScript and XML), 5, 37-48, 297**

- caches, 307
- communication, 333-343
- converters, 342
- data types, 305-307
- default values, 330
- events, 330-333
- external links (Hijax), 424
- filters, 323-327
- getJSON()/parseJSON() methods, 316-319
- jqXHR type, 57
- JSONP extension (Twitter requests), 317-319
- load() method, 322-327
- remote requests, 304
- requests, 303-304
 - loading, 307-315
 - methods, 305
- scripts, 320-322
- serialization, 327-329
- support, 304-307
- Web 2.0, 33
- XML data, 312-315
- XMLHttpRequest, 297-304

ajax() method, 305, 333**alert() method, 215, 378****algorithms, animation. See animation****aliases, defining, 73****all object, 466****alternatives for Mobile frameworks, 421****American National Standards Institute.**

See ANSI

ancestors, 86, 88**Android, 419****andSelf() method, 195-196****animate() method, 386-387****:animated filter, 100****animation, 279**

- animate() method, 289-295
- callbacks, 280
- chaining, 281
- colors, 386
- CSS, 279
- effects
 - opacity, 287-289
 - sliding, 284-287
- elements, 25-28, 102
- endless, 282
- length of, 279-280
- objects, 286
- overview of, 279
- plug-ins, 411
- queues, 281
- rates, 280
- show()/hide() methods, 283-284
- stopping, 282
- types of, 282-283
- UIs, 347

anonymous functions, 249, 460

- binding, 263
- executing, 262
- as parameters, 60-63
- specifying, 280

ANSI (American National Standards Institute) code, 40**append() method, 138-142, 201****appendChild() method, 143****appendTo() method, 66, 143-148**

applications

- AJAX. *See* AJAX
- JSON, 316
- loading, 455
- logic, 303
- Mobile. *See* Mobile
- RIAs, 1
- web. *See* web applications

applying

- add() method, 194
- animate() method, 289-295
- appendTo() method, 147
- callback functions, 210-217
- constructors, 461
- existing plug-ins, 394-397
- fading methods, 288
- filters, 101
- getters/setters, 362
- selectors, 84, 87-97
- style sheets, 49
- UIs, 349-355

areas

- data output, 319
- div, 94, 96, 101, 161
- span, 94, 96, 155
- wrapping, 158

arguments, 460**arrays**

- autocomplete() method, 375
- declaring, 463
- elements, 463
- iterating, 186-193
- JavaScript, 463-464
- JSON, 42
- literals, 464
- notation, 462

Array<Type> notation, 56-57**arrows, formatting, 430****ASP.NET, 3****assigning animation speed, 280****asynchronous communication, 38, 298****Asynchronous JavaScript and XML. *See* AJAX****attr() function, 80, 137****attribute filter, 115****attributes**

- attr() method, 137
- class, 89
- data-icon, 430
- data-role (Mobile), 422
- date-rel=dialog, 428
- deleting, 171
- disabled, 171
- filters, 114-118
- HTML event handlers, 248
- initializing, 68-70
- modifying, 28-29
- selectors, 50
- style, 357
- XML tags, 39

attr() method, 28**autocomplete() method, 374-375****Autocomplete widget, 346, 348****auxiliary functions, 274-277****availability, 32, 393****avoiding caches, 307****axes, 85**

B
background colors, 70, 194, 206**backslash (\), 85****Bada, 419****beforeSend event, 330****bind() method, 259, 262-266****BlackBerry, 419****block elements (Mobile), 431****blocking**

- pop-ups, 377
- web pages, 379

blogs, micro, 317

blur() method, 260**Boolean values, 458****breaks, lines, 178****browsers**

caching, 307

DOM access, 83

events, 250

Mobile, 420

support, 34

web, 32

bubbling events, 251, 265**:button filter, 119****buttons**

footers, 434

graphics, 431

groups, 431

with icons, 430-431

Mobile, 429-435

button type, 376**Button widget, 346**

C

caches, avoiding, 307**calendars, 197, 377. See also dates****callbacks**

AJAX, 304

animation, 280

events, 331

functions, 210-217, 460

as parameters, 60-63

queues, 331

calls, functions, 460**CamelCase names, 81****Cascading Style Sheets. See CSSs****case sensitivity (JavaScript), 457****CDNs (Content Distribution Networks),****350, 354****chaining**

animation, 281

CSS formatting, 209

events, 331

methods, 77

objects, 77-78

change() method, 260**:checkbox filter, 119****:checked filter, 121****checking nodes, 131-135****child filters, 112-114****child nodes, 86****children() method, 176-180****Chrome for Android, 419****.class1 selector, 88****.class2 selector, 88****class attribute, 89****.class selector, 88****classes**

adding, 24

elements, 463

Float, 458

Integer, 458

Math, 458

modifying, 22, 209-222

naming, 210

Number, 458

pseudo, 50

selectors, 92

testing, 221-222

toggling, 219-221

UIs, 356

clearQueue() method, 78, 281**click() method, 21, 24, 255, 260****clients, 32****cloning elements, 172-175****close parameter, 429****closest() method, 180-181****code**

ANSI, 40

plug-ins, 406

snippets, 34

collapsed/expanded content (Mobile), 454-455

collecting static functions in objects, 409

colon (:), 42

colors

- animation, 386
- backgrounds, 70, 194, 206
- fonts, 209

comments (HTML), 35

communication (AJAX), 333-343

compilers, just-in-time, 34

complete event, 330

complete option, 290

components

- dates, 196-203
- objects, 462
- rules, 358-359
- spinner, 384
- UIs, 355-369
 - events, 366-369
 - interaction, 370-372
 - methods, 363-346
 - properties, 359-362

configuring

- defaults
 - plug-in options, 410
 - UI settings, 356-358
- font colors, 209
- positioning, 231-235
- properties, 207-209
- tabs, 381
- values
 - css() method, 207
 - options, 361

conflicts

- naming, 411
- noConflict() function, 73-74

connections, <script> containers in websites, 34-36

console errors, 66

constructors

- event objects, 252
- objects, 461

containers

- div, 140
 - Accordion widgets, 373
 - inserting, 153
 - resizing, 371
- fields, 444
- <script>, 34-36

:contains (text) filter, 106

content

- adding, 149
- collapsed/expanded (Mobile), 454-455
- design (Mobile), 452-443
- div elements, 163
- nodes, 132-135
- val() method, 135-137
- web pages, 154-159

Content Distribution Networks. See CDNs

content filters, 106-108

contents() method, 184-185

context, 59, 74-76

conventions, writing, 5-6

converters (AJAX), 342

converters option, 306

creditcard() method, 403

cross-over scripting, 304

css() method, 58, 206-209, 406

CSSs (Cascading Style Sheets), 2, 34

- animation, 279
- classes
 - adding, 24
 - modifying, 22
- dates, 197
- files, 25
- Mobile, 435
- modifying, 22
- overview of, 48-50
- properties, 351

- rules, 434
- selectors, 89
- syntax, 50
- themes, 350
- transitions, 428
- widgets, 346

curly brackets ({ }), 42

customizing

- animation, 292
- downloads, 349-350
- icons, 430
- JavaScript functionality, 434
- plug-ins, 405-414
- themes, 453

D

data binding

- events, 251-252
- sliders, 367

data-icon attribute, 430

data() method, 81

data-role attribute (Mobile), 422

data structures, 42. See also JSON

data types, 55-57

- AJAX, 305-307
- formatting, 463
- JavaScript, 457, 458

date() method, 403

dateDE() method, 403

dateISO() method, 403

date-rel=dialog attribute, 428

Datepicker widget, 346, 376-377

dates

- examples, 196-203
- selecting, 377

days, 201. See also dates

dblclick() method, 260

declaring

- arrays, 463
- CSS syntax, 50
- empty elements (XML), 39
- functions, 460
- objects, 462

default event handlers, 256-258

default options, plug-ins, 410

default themes, 350

default UI settings, 356-358

default values (AJAX), 330

defining

- aliases, 73
- animation, 292

delay() method, 78, 281

delegate() method, 273

deleting

- attributes, 171
- content, 163
- nodes, 131

dequeue() method, 78, 281

descendants, 86-88

- rules, 93
- searching, 184

design. See also formatting

- Mobile, 452-443
- premade, 347

detach() method, 166-172

devices, mobile. See Mobile

DHTML (Dynamic Hypertext Markup Language), 5, 34

- AJAX, 302
- style sheets, 48-50
- web pages, 22-25

dialog() method, 429

dialogs (Mobile), 428-429

Dialog widget, 346, 377-379

die() method, 274

digits() method, 403

dimensions, inner/outer, 242-244

directories, plug-ins, 401. See also plug-ins

disabled attribute, 171

:disabled filter, 121

disabling objects, 363
div areas, 94-96, 101, 161
div containers, 140
 Accordion widgets, 373
 inserting, 153
 resizing, 371
div elements
 deleting, 163
 formatting, 276
 marking, 424
div type, 57
DOCTYPE statements, 21, 422
documentation, plug-ins, 394
Document Object Model. See DOM
document.ready() method, 63
documents
 objects 466
 XML, 39. *See also* XML
DOM (Document Object Model), 18, 34, 461
 accessing, 83
 elements, 52, 71-72
 event handlers, 251
 functions, 60-65
 mobile widgets, 424
 objects, 46-48, 465-466
 server responses, 313
DOM Inspector, 53
domains, executing, 321
dot notation, chaining via, 281
Download Builder (UI), 349-350
downloading frameworks (Mobile), 420
Draggable component, 370
draggable () method, 346, 359, 406
dragging. See also moving
 images, 356, 407
 sliders, 367
dragwithstatuslight() method, 406
Droppable component, 370
droppable() method, 346
duration option, 290

Dynamic Hypertext Markup Language. See DHTML

E

each() method, 186-193
easing option, 290
editing web pages, 22-25
effect() method, 385
effects, 279
 animation. *See* animation
 hover, 274-276
 opacity, 287-289
 sliding, 284-287
 transitions, 428
 UIs, 347, 385-387
elements
 accessing, 17, 47, 132-135
 animation, 25-28, 102
 appendTo()/prependTo() methods, 143-148
 arrays, 463
 block (Mobile), 431
 classes, 463
 cloning, 172-175
 content
 adding, 149
 deleting, 163
 context, 76
 Dialog widgets, 378
 div
 formatting, 276
 marking, 424
 DOM, 71-72
 finding, 180, 184-185
 forms
 filters, 118-123
 Mobile, 443-448
 height/width of, 240
 inline (Mobile), 431
 jQuery() function, 70-72

- moving, 141
- nesting, 40
- objects, 466
- offset() method, 228-235
- positioning, 226
- removing, 166-172
- replacing, 159-166
- <script>, 35
- selectors, 50, 84, 88, 91
- spacing, 214
- style sheets, 209-222
- unwrapping, 159
- url types, 315
- web pages
 - accessing, 52-54, 131
 - inserting, 66-70
 - wrapping, 154-155
 - XML, 39
 - XPath, 85
- email() method, 403**
- embedding style sheets, 49**
- empty elements (XML), declaring, 39**
- :empty filter, 106**
- empty() method, 166-172**
- empty web pages, 67**
- :enabled filter, 121**
- endless animation, 282**
- end() method, 195-196**
- end tags (XML), 39**
- enlarging elements, animation, 25-28**
- :eq (index), 100**
- eq() method, 123**
- errorClass option, 402**
- error() method, 260**
- errors**
 - consoles, 66
 - events, 330
 - plug-ins, 402
- eval() method, 306**
- :even filter, 100**
- even indexes, 104**
- event.currentTarget property, 253**
- event.data property, 253**
- event handlers, 247**
 - context, 75
 - HTML, 248-249
 - JavaScript, 249-250
 - methods, 262-277
 - onload, 60
 - overview of, 247-252
- event.pageX property, 253**
- event.pageY property, 253**
- event.relatedTarget property, 253**
- event.result property, 253**
- events, 247**
 - \$ (document).ready() method, 258
 - AJAX, 330-333
 - bubbling, 251, 265
 - callbacks, 331
 - data binding, 251-252
 - global, 332-333
 - helpers, 258-262
 - live, 270-274
 - local, 330
 - Mobile, 448-451
 - objects, 250-258, 466
 - methods, 256-258
 - properties, 253-256
 - page, 449
 - scroll, 449
 - touch, 448
 - triggers, 252
 - UI components, 366-369
- event.screenX property, 253**
- event.screenY property, 253**
- event.target property, 253**
- event.timeStamp property, 253**
- event.type property, 253**

examples

- dates, 196-203
- Mobile, 432-435
- requests, 300-302
- selectors, 87-97
- UIs, 348

exceptions, XMLHttpRequest objects, 299**executing**

- anonymous functions, 262
- foreign domains, 321
- functions
 - after DOM has been built, 60-65
 - sequentially, 78
- JavaScript
 - files, 340
 - libraries, 322

existing plug-ins, searching, 394-397**expressions, filters, 176****extended effects, UIs, 347****Extensible HTML. See XHTML****Extensible Markup Language. See XML****extensions, 463**

- animation, 283
- jqXHR objects, 305
- JSONP, 304
- Twitter requests, 317-319

external files

- CSS, 49
- document.ready() method, 63
- integration, 36
- JavaScript, 36

external links (Hijax), 424**external style sheets, 22****F****fadeOut() method, 287****fadeOut() method, 287****fadeOut() method, 287, 290****features, new core techniques, 79****fields**

- containers, 444
- forms, 401
- objects, 83
- rules, 404

:file filter, 119**files**

- CSS, 25, 49. *See also* CSSs
- external, 36
- external style sheets, 22
- JavaScript, 36
 - executing/loading, 340
 - loading, 321
- XML, 38-41, 313. *See also* XML

filter() method, 125, 126, 288**filters, 83, 84**

- AJAX, 323-327
- attributes, 114-118
- child, 112-114
- content, 106-108
- expressions, 176
- forms, 118-123
- methods, 123-128
- overview of, 99-106
- selectors, 99-123
- visibility, 109-111

find() method, 184-185**finding, 176, 184-185. See also searching**

- elements, 180

Firebug, 84**Firefox. See also UIs**

- DOM trees, 53
- Mobile, 419

:first-child filter, 112**:first filter, 100****first() method, 124****Float class, 458****focus() method, 260****:focus filter, 100, 121****focusin() method, 260**

focusout() method, 260**following**

- nodes, 86
- sibling nodes, 86

font colors, configuring, 209**footers, 434-436****foreign domains, executing, 321****formatting**

- AJAX communication, 301-303
- array literals, 464
- arrows, 430
- buttons, 429-435
- case sensitivity, 457
- CSS, 49. *See also* CSSs
- data output areas, 319
- data types, 463
- default UI settings, 356-358
- elements
 - div, 276
 - jQuery() function, 66-70
- empty web pages, 67
- event objects, 252
- indexes, 104
- level 1 headings, 58
- plug-ins
 - customizing, 405
 - rules, 409-411
- style sheets, 205-206
 - css() method, 206-209
 - modifying classes, 209-222
 - removeClass() method, 218-221
 - testing classes, 221-222
 - tooggling classes, 219-221
- tables, 325
- text() method, 134
- web pages, 424
- XMLHttpRequest, 298-299

forms

- filters, 118-123
- JSON data, 339

Mobile, 443-448

- objects, 466
- serializing, 328
- validation, 401
- val() method, 135-137

fragments (HTML), 325**frame object, 466****frameworks, 1, 72-74**

- design, 452-443
- Mobile, 420
- Theme, 347

functionality

- AJAX, 310
- JavaScript, 434

functions. *See also* methods

- anonymous, 249. *See also* anonymous functions
 - binding, 263
 - executing, 262
 - as parameters, 60-63
- arguments, 460
- attr(), 80, 137
- auxiliary, 274-277
- callbacks, 210-217, 304
- calls, 460
- declaring, 460
- DOM, 60-65
- hover(), 274
- JavaScript, 459-460
- jQuery(), 57-60
 - creating elements, 66-70
 - wrapping elements, 70-72
- jQuery.sub(), 78
- jQuery.when(), 79
- myfction(), 249
- named, 263
- noConflict(), 73-74
- prop(), 80
- references, 460
- removeProp(), 80

G

Gallery (ThemeRoller), 352
generation selectors, 50
get() method, 71-72, 305
getAllResponseHeaders() method, 299
getElementsByTagName() method, 315
getJSON() method, 316-319
getResponseHeader() method, 299
getScript() method, 298, 320-322
getters, 362
global events, 332-333
graphical user interfaces. See GUIs
graphics, positioning, 431
groups
 buttons, 431
 selectors, 85
:gt (index), 100
GUIs (graphical user interfaces), 2, 346, 443-448. See also UIs

H

hardware, 6
has() method, 184
hasClass() method, 221-222
:has selector, 106
<head> tag, 37
:header filter, 100
headers, 102
headings
 level 1, 58
 modifying, 161
 moving, 151
height() method, 239-242
helpers, events, 258-262
hide() method, 283-284
 opacity, 288
 sliding effects, 284
hiding
 variables, 410
 web pages, 450

hierarchies. See also trees
 nodes, 85
 selectors, 85-99
highlight option, 402
history
 div elements, 162
 objects, 466
history.back() method, 426-428
horizontal lines, 178
hover() function, 274
HTML (Hypertext Markup Language), 32
 Accordion widget, 390
 comments, 35
 content, 155
 elements, 53
 event handlers, 248-249
 fragments, 325
 loading, 418
 statements, 422
 style sheets, 49
 tags, 134, 302. *See also* web pages
 widgets, 346

html() method, 132-137
HTTP (Hypertext Transfer Protocol), 32, 425
 transactions, 38
Hypertext Markup Language. See HTML
Hypertext Transfer Protocol. See HTTP

I

icons
 buttons with, 430-431
 positioning, 431
#id selector, 88
identifiers
 " (quotation marks), 56
 JavaScript, 55
IDEs (integrated development environments), 19
IDs, selecting, 90

:image filter, 119

images

- cloning, 173
- dragging, 356, 407
- modifying, 25
- objects, 466
- overlapping, 226
- positioning, 226, 233
- scrolling, 239
- sizing, 241
- spacing, 237

include() method, 37

indexes, 104

init() method, 348

initializing

- AJAX requests, 330
- attributes, 68-70

inline elements (Mobile), 431

inner child elements, wrapping, 158

innerHTML property, 66

:input filter, 119

insertAfter() method, 152-153

insertBefore() method, 152-153

inserting

- div containers, 153
- elements into web pages, 66-70
- nodes into web pages, 137-153
- text, 139

Integer class, 458

integrated development environments. See IDEs

integration

- JavaScript in websites, 34-37
- libraries, 424
- Mobile frameworks, 420
- new plug-ins, 443
- plug-in libraries, 399

interaction

- auxiliary functions, 274-277
- calendars, 377

form elements, 443-448

UIs

- components, 370-372
- support, 346

interfaces. See also browsers

GUIs, 2

Mobile, 420

UIs. *See* UIs

internal links, 425-426

Internet, 32

Internet Explorer, 53, 346. See also UIs

intervals

- jQuery.fx.interval, 280
- temporal, 26

iOS, 419

iPhone plug-ins, 422

is() method, 80, 126

isDefaultPrevented() method, 256-258

iterating objects, 186-193

J

JavaScript

- arrays, 463-464
- case sensitivity, 457
- data types, 457-458
- event handlers, 248-250
- external files, 36
- files
 - executing/loading, 340
 - loading, 321
- functionality, 434
- functions, 459-460
- integration in websites, 34-37
- libraries, 322
- namespaces, 55
- objects, 461-463
- plug-in integration, 400
- relationship to jQuery, 33-37

JavaScript Object Notation. See JSON

jQuery, 421

jQuery() function, 57-60, 66-72
 jQuery.fx.interval, 280
 jQuery.sub() function, 78
 jQuery.when() function, 79
 jqXHR, 57, 305
 JSON (JavaScript Object Notation), 4

- applications, 316
- getJSON()/parseJSON() methods, 316-319
- overview of, 41-45

 JSONP extension, 304

- requests, 339
- Twitter requests, 317-319

 just-in-time compilers, 34

K

keydown() method, 260
 keypress() method, 260
 keyup() method, 260
 keywords, return, 459
 Kindle 3/Fire, 419

L

:last-child filter, 112
 :last filter, 100
 last() method, 124
 left values, 224
 length of animation, 279-280
 level 1 headings, 58
 library random.js structure, 322
 libraries, 19, 459

- integration, 424
- JavaScript, 322
- jQuery
 - applying frameworks in combination, 72-74
 - references, 36-37
- plug-in integration, 399
- UIs, 354

lines

- breaks, 178
- horizontal, 178

 <link> tag, 49
 linking

- CSS themes, 350
- Download Builder (UI), 349-350
- external links (Hijax), 424
- internal links, 425-426
- web pages (Mobile), 424-428

 listings

- AJAX
 - basic XML files, 314
 - binding to global events, 332
 - chaining callbacks, 331
 - converters, 342
 - custom data types, 342
 - default method for sending, 330
 - different ways of using load()
 - method, 324
 - ending requests, 300
 - executing/loading JavaScript files, 340
 - formatting data output areas, 319
 - forms with data to serialize, 328
 - HTML fragments, 325
 - HTML tags that do not exist, 326
 - initializing requests, 330
 - JSONP requests, 339
 - JSON structures, 303
 - library random.js structure, 322
 - loading HTML via GET, 323
 - loading JavaScript files, 321
 - loading Twitter tweets via JSONP, 317
 - loading via filters, 323
 - load() with filters, 324
 - patterns of load() method, 323
 - prefilter callback functions, 341
 - processing JSON, 316
 - processing XML responses, 315

- reactions to errors, 332
- requests, 301
- requests via GET/POST, 308
- schematic form of get()/post() methods, 307
- sending data, 311
- sending data via POST, 323, 340
- sending JSON data to forms, 339
- serializeArray() method, 329
- serializing forms, 328
- text fragments with scripts, 320
- transports, 343
- verifying access, 310
- web pages loading via, 301
- XML file requests, 313
- XMLHttpRequest objects, 299
- aliases, defining, 73
- All Headings, selecting, 59
- All Input Elements, 59
- animation
 - applying animate() method, 289
 - calling animate() method, 290
 - chaining via dot notation, 281
 - chaining without dot notation, 281
 - CSS files, 285
 - defining custom, 292
 - endless, 282
 - fading methods, 288
 - initial CSS specifications, 291
 - modifying opacity, 288
 - patterns, 289
 - sliding methods, 284-286
 - specifying anonymous functions, 280
 - starting chained, 295
 - stopping, 282
 - variations of animate() method, 291, 294
- anonymous functions, 62
- attributes
 - modifying, 29
 - value of src, 137
- background colors, 70
- Boolean property values, 80
- callbacks to functions, 61
- code snippets, 34
- context, requesting, 75
- CSS
 - adding classes, 24
 - declaring, 50
 - files, 25
 - modifying classes, 22
- div type, 57
- document.ready() method, 65
- DOM, accessing properties, 71-72
- elements
 - creating with options, 68
 - initializing div, 68
 - integrating into web sites, 67
- empty elements, 39
- empty web pages, 67
- end tags (XML), 39
- event handlers
 - anonymous functions, 249
 - applying bind()/unbind() methods, 264
 - applying new operators, 252
 - binding anonymous functions, 263
 - binding named functions, 263
 - bound, 263
 - bubbling, 265
 - calling functions in HTML, 249
 - click/double-click handlers, 268
 - click() method, 261
 - creating new objects, 252
 - delegate() method, 273
 - evaluating attributes, 256
 - evaluating properties, 254
 - executing anonymous functions, 262
 - formatting div elements, 276
 - hover effects, 274-276

- hover() method, 275-277
- HTML, 248
- interpreting attributes, 254
- JavaScript, 249
- live events, 271
- Map type objects, 266
- multiple events, 265
- passing on data, 266
- preventing redirection, 257
- submitting form data, 265
- triggers, 267, 272
- external CSS files, 49
- external file integration, 36
- files with external style sheets, 22
- filters
 - alt attribute, 117
 - animated elements, 101
 - applying filter() method, 126
 - based on selection state, 122
 - basic file for, 99
 - basis of filtering, 125
 - check boxes, 122
 - checking for attributes, 116
 - different content, 108
 - exclusion of elements, 106
 - filtering index ranges, 124
 - first and last children, 113
 - forms, 118
 - four images and a button, 110
 - headers, 103
 - :hidden/:visible, 110
 - indexes, 104
 - is() method, 126
 - mapping objects into arrays, 127
 - nested containers, 112-113
 - new basis file, 108
 - remove() method, 170
 - returning values in input fields, 128
 - selected children/only children, 114
 - selection based on word you, 107
 - slice() method, 124
 - specification of alternative text, 116
 - specific form filters, 120
 - tags with attributes, 114
- forms
 - accessing fields, 135
 - applying val() method, 136
- functions, declaring, 64
- generic types, 56
- HTML pages (JSON), 43
- images, enlarging/reducing, 25
- is() method, 80
- jQuery example, 18
- JSON structures, 42
- level 1 headings, 58
- methods, chaining, 77
- Mobile
 - applying lists, 439
 - browser sniffers, 425
 - closing dialogs, 429
 - collapsed/expanded content, 454
 - CSS files, 435
 - custom JavaScript functionality, 434
 - fixed footers, 436
 - formatting themes, 453
 - form elements, 443
 - grouping buttons, 431
 - hiding/showing web pages, 450
 - history.back() method, 426
 - HTML code for, 432
 - icons, buttons with, 430
 - lists, 439
 - navigation bars, 436
 - new plug-ins, 443
 - opening dialogs, 428
 - overview of buttons, 429
 - page segments, linking to, 426
 - positioning graphics in buttons, 431
 - reacting to events, 450

- specifying transition effects, 428
 - web page structures, 423
- modularized web applications, 64
- notation of options, 56
- options, 55
- plug-ins
 - applying, 400, 407
 - calling plug-ins, 410
 - collecting in objects, 409
 - customizing, 407
 - different ways of using, 411
 - dragging images, 407
 - JavaScript file for modified, 411
 - methods, 408
 - options, 414
 - random generators in, 413
 - rules for fields, 404
 - source code, 406
 - tree view based on, 396
 - validation, 399
 - variations of, 412
 - web pages that reference, 395
- prologs (XML), 40
- ready() method, 62, 74
- referenced JavaScript files, 44
- results in 1.5.2, 81
- results in 1.6, 81
- schemas, 49
- script language specifications, 35
- selectors
 - basic structures, 89
 - children/parent relationships, 95
 - classes, 92
 - descendants, 94
 - div elements in paragraphs, 98
 - div/span areas, 96
 - elements, 91
 - IDs, 90
 - sibling relationships, 97
 - specifying two classes, 93
- sequence of integration errors, 65
- start tags (XML), 39
- style sheets
 - accessing height/width of elements, 240
 - addClass() method, 211
 - adding classes, 210
 - applying, 208
 - assigning CSS classes, 213, 218-220
 - background colors, 206
 - basic file with four images, 224
 - callback functions, 234
 - configuring positioning, 231
 - configuring values for properties, 207
 - CSS files, 212, 225
 - document offset in action, 229
 - getting CSS properties, 208
 - hasClass() method, 221
 - height()/width() methods, 240
 - inner/outer dimensions, 243
 - modifying CSS files, 227
 - naming properties, 207
 - offset() method, 229
 - parameters, callback functions with, 215
 - positioning, 224
 - removeClass() method, 218
 - removing CSS classes, 218-220
 - scrolling methods, 236-238
 - spacing images, 237
 - specifying callback functions, 210-211
 - specifying dimensions of blocks, 243
 - toggleClass() method, 219
- UIs
 - Accordion elements from HTML, 390
 - Accordion widgets, 372
 - animate() method, 386
 - animating colors, 386

- applying classes, 356
- autocomplete() method, 374
- button type, 376
- creating Tooltips, 384
- CSS formatting, 388
- Datepicker widget, 376
- Dialog widget, 378
- disabling objects, 363
- div areas converted into tabs, 383, 389
- div container converts to
 - Accordion, 373
- dragging images, 356
- dragging sliders, 367
- Draggable component, 371
- enabling/disabling draggability, 363
- events, 367
- getters/setters, 362
- interactive calendars, 377
- outputting option values, 362
- positioning, 385
- processing property values, 365
- progress bars, 381
- Resizable component, 371
- resizing div containers, 371
- sliders, 365, 380
- starting URLs, 353
- tabs, 382
- templates, 355
- viewing tabs, 366
- websites, 387
- web pages
 - accessing objects, 186
 - add() method, 194
 - after()/before() methods, 149-151
 - appending nodes, 145
 - appendTo() method, 143
 - append() vs. appendTo(), 144
 - basic web pages, 152
 - callback functions as parameters, 190
 - cloning elements, 172
 - closest() method, 181
 - contents()/find() methods, 185
 - contents() method, 185
 - content with HTML, wrapping, 155
 - date components, 197-198
 - days written into tables, 202
 - default template files, 197
 - descendent rules, 184
 - determining parent/child elements, 177
 - disabled attribute, 171
 - each() method, 188-189, 192
 - elements, wrapping with, 157
 - elements with parents/multiple children, 177
 - empty cells appended in tables, 202
 - enumerated lists, 195
 - finding elements, 180
 - formatting lists, 195
 - html()/text() method, 133
 - inner child elements, 158
 - insertAfter()/insertBefore(), 153
 - inserting elements, 149-150, 169-170
 - iteration over objects, 187
 - moving elements, 141
 - with multiple images, 52
 - offsetParent() method, 181
 - processing return values, 162
 - remove() method, 169
 - removing elements, 167
 - removing nodes, 167
 - removing parent elements, 159
 - replacing elements, 164
 - replacing parts of, 160
 - replacing separator lines, 165
 - selectors, formatting, 202
 - siblings, 179, 182-184
 - text nodes, 132
 - wrapAll() method, 157

- wrapInner() method, 158
- wrap() method, 154
- XML files, 40
- lists (Mobile), 437-443**
- literals, 457**
 - arrays, 464
 - objects
 - declaring, 462
 - options, 55
- live events, 270-274**
- load() method, 260, 322-327**
- loading**
 - applications, 455
 - HTML, 418
 - JavaScript
 - files, 321, 340
 - libraries, 322
 - requests, 307-315
 - scripts (AJAX), 320-322
 - tables, 325
 - web pages, 144
- local events, 330**
- location object, 466**
- logic, moving, 303**
- lowercase (XML), 40**
- :lt (index), 100**

M

- Map, 56**
- map() method, 127**
- marking div elements, 424**
- Math class, 458**
- maxlength() method, 403**
- max() method, 403**
- MeeGo, 419**
- menus**
 - plug-in navigation, 394
 - widgets, 384
- messages option, 402**

- methods**
 - \$.get, 307-315
 - \$.post, 307-315
 - \$(document).ready(), 258
 - abort(), 299
 - accept(), 403
 - add(), 193-196
 - addClass(), 210-217
 - after(), 149-151
 - ajax(), 305, 333
 - AJAX requests, 305
 - alert(), 215, 378
 - andSelf(), 195-196
 - animate(), 289-295, 386-387
 - append(), 138-142, 201
 - appendChild(), 143
 - appendTo(), 66, 143-148
 - attr(), 28, 137
 - autocomplete(), 374-375
 - before(), 149-151
 - bind(), 259, 262-266
 - blur(), 260
 - chaining, 77
 - change(), 260
 - children(), 176-180
 - clearQueue(), 78, 281
 - click(), 21, 24, 255, 260
 - closest(), 180-181
 - contents(), 184-185
 - creditcard(), 403
 - css(), 58, 206-209, 406
 - data(), 81
 - date(), 403
 - dateDE(), 403
 - dateISO(), 403
 - dblclick(), 260
 - delay(), 78, 281
 - delegate(), 273
 - dequeue(), 78, 281
 - detach(), 166-172

dialog(), 429
die(), 274
digits(), 403
document.ready(), 63
draggable(), 406
draggable (), 346, 359
dragwithstatuslight(), 406
droppable(), 346
each(), 186-193
effect(), 385
email(), 403
empty(), 166-172
end(), 195-196
eq(), 123
error(), 260
eval(), 306
event handlers, 262-277
event objects, 256-258
fadeIn(), 287
fadeOut(), 287
fadeTo(), 287, 290
filter(), 125-126, 288
filters, 123-128
find(), 184-185
first(), 124
focus(), 260
focusin(), 260
focusout(), 260
get(), 71-72, 305
getAllResponseHeaders(), 299
getElementsByTagName(), 315
getJSON(), 316-319
getResponseHeader(), 299
getScript(), 298, 320-322
has(), 184
hasClass(), 221-222
height(), 239-242
hide(), 283-284
 opacity, 288
 sliding effects, 284
history.back(), 426-428
html(), 137
include(), 37
init(), 348
insertAfter(), 152-153
insertBefore(), 152-153
is(), 80, 126
isDefaultPrevented(), 256-258
JavaScript, 459-461
keydown(), 260
keypress(), 260
keyup(), 260
last(), 124
load(), 260, 322-327
map(), 127
max(), 403
maxlength(), 403
min(), 403
minlength(), 403
mousedown(), 260
mouseenter(), 260
mouseleave(), 260
mousemove(), 260
mouseout(), 255, 260
mouseover(), 260
mouseup(), 260
next(), 182
nextAll(), 182
not(), 123
number(), 403
numberDE(), 403
offset(), 228-235
offsetParent(), 180-181
one(), 266
open(), 299
param(), 329
parent(), 176-180
parents(), 176
parentsUntil(), 176
parseJSON(), 306, 316-319

- plug-ins, 447
 - position(), 224-228
 - positioning, 223-235
 - post(), 305
 - prepend(), 138-142
 - prependTo(), 143-148
 - prev(), 182
 - prevAll(), 182
 - preventDefault(), 256-258, 265
 - queue(), 78, 281
 - rangelength(), 403
 - ready(), 20, 36, 62, 74, 201, 435
 - remove(), 166-172
 - removeAttr(), 166-172
 - removeAttrs(), 403
 - removeClass(), 24, 218-221
 - replaceAll(), 164-166
 - replaceWith(), 160-164
 - require(), 37
 - required(), 403
 - resize(), 260
 - resizeable(), 346
 - rules(), 403
 - scroll(), 260
 - scrolling, 236-238
 - ScrollLeft, 236
 - ScrollTop, 236
 - select(), 260
 - selectable(), 346, 372
 - send(), 299, 304
 - serialize(), 327-328
 - serializeArray(), 329
 - setMimeType(), 300
 - setRequestHeader(), 300
 - show(), 283-284
 - opacity, 288
 - sliding effects, 284
 - size(), 464
 - slice(), 124
 - slideDown(), 284-287
 - slideToggle(), 26, 102, 284-287
 - slideUp(), 284-287
 - sortable(), 346, 372
 - stopPropagation(), 265
 - submit(), 260
 - text(), 137, 162
 - toggle(), 275, 287
 - toggleClass(), 25, 219, 263
 - transports, 342
 - treeview(), 397
 - trigger(), 267-269
 - triggerHandler(), 269-270
 - UI components, 363-346
 - unbind(), 262-266
 - undelegate(), 274
 - unload(), 260
 - unwrap(), 159
 - url(), 403
 - val(), 135-137
 - valid(), 403
 - validate(), 399, 402-403
 - width(), 239-242
 - wrap(), 154-155
 - wrapAll(), 155-157
 - wrapInner(), 158
 - XMLHttpRequest, 299-300
- micro blogs, 317**
 - min() method, 403**
 - minlength() method, 403**
 - mixing themes, 453**
 - Mobile, 417**
 - block elements, 431
 - buttons, 429-435
 - collapsed/expanded content, 454-455
 - design, 452-443
 - dialogs, 428-429
 - examples, 432-435
 - forms, 443-448
 - frameworks, 420
 - history.back() method, 426-428
 - lists, 437-443
 - overview of, 417-422

- role system, 422
- special events, 448-451
- toolbars, 435-438
- transitions, 428
- web pages
 - linking, 424-428
 - structures, 422-424

modifying

- attributes, 28-29
- classes, 22, 209-222
- CSSs, 22
- headings, 161
- images, 25
- nodes, 131-135, 142
- opacity, 288
- orientation, 448

modularized web applications, 63-65**months, 201. See also dates**

- `mousedown()` method, 260
- `mouseenter()` method, 260
- `mouseleave()` method, 260
- `mousemove()` method, 260
- `mouseout()` method, 255, 260
- `mouseover()` method, 260
- `mouseup()` method, 260

moving

- elements, 141
- headings, 151
- icons, 431
- images, 356, 407
- logic, 303
- widgets, 347

multimedia (AJAX), 302**multiple events, `bind()` method, 265****`myfction()` function, 249**

N

name-value pairs (JSON), 41**named functions, 263, 460****namespaces, 54-55****naming**

- classes, 210
- conflicts, 411
- properties, 207

navigating. See also Uls

- buttons, 429-435
- `history.back()` method, 426-428
- trees, 85

navigation bars (Mobile), 435-438**navigation menus, plug-ins, 394****navigator object, 466****nesting, 155**

- elements, 40
- themes, 453

Netscape, 83**new core techniques, 78-81****new operators, 252****new plug-ins, integrating, 443****`next()` method, 182****`nextAll()` method, 182****`noConflict()` function, 73-74****`nodeName` property, 76****nodes**

- `appendTo()/prependTo()` methods, 143-148
- checking, 131
- content, 132-135
- modifying, 142
- objects, 466
- removing, 167
- testing, 85
- text, 140
- web pages, 137-153
- XPath, 85

Nook Color, 419**notation**

- arrays, 462
- `Array<Type>`, 56-57
- CSS properties, 207
- dot, 281

- objects, 186
- options, 56
- not() method, 105, 123**
- :not selector, 100**
- :nth-child filter, 112**
- Number class, 458**
- numberDE() method, 403**
- number() method, 403**

O

- object-oriented programming (OOP), 46**
- objects, 54-55, 458**
 - accessing, 186, 462
 - animation, 286
 - chaining, 77-78
 - declaring, 462
 - disabling, 363
 - DOM, 46-48, 465-466
 - events, 250-258
 - methods, 256-258
 - properties, 253-256
 - fields, 83
 - JavaScript, 461-463
 - jqXHR, 305
 - literals, 55
 - standardization, 309
 - static functions in, 409
 - transports, 342
 - web pages, 186-193
 - XDomainRequest, 298
 - XMLHttpRequest, 38, 297-304
- :odd filter, 100**
- odd indexes, 104**
- offset() method, 228-235**
- offsetParent() method, 180-181**
- one() method, 266**
- onload event handler, 60**
- :only-child filter, 112**
- onreadystatechange property, 300, 305**
- OOP (object-oriented programming), 46**

- opacity, animation, 287-289**
- open() method, 299**
- Opera Mobile, 419**
- operating systems, 6, 419-420**
- operators, new, 252**
- options, 55-56**
 - animate() method, 290
 - attributes, 68-70
 - converters, 306
 - draggable () method, 359
 - plug-ins, 410, 411-414
 - UI components, 359-362
 - validate() method, 402
 - values, 361
- ordered value lists (JSON), 41**
- orientation, modifying, 448**
- orientationchange event, 448**
- outer dimensions, 242-244**
- outputting properties, 187**
- overlapping images, 226**

P

- page events, 449**
- pagebeforecreate event, 449**
- pagebeforehide event, 449**
- pagebeforeshow event, 449**
- pagecreate event, 449**
- pagehide event, 449**
- pageshow event, 449**
- Palm webOS, 419**
- paragraphs, removing parent elements, 159**
- param() method, 329**
- parameters, 26**
 - anonymous functions as, 60-63
 - attributes, 28-29
 - callbacks as, 60-63, 210
 - close, 429
 - showing/hiding, 283-284
- parent > child selector, 88**
- :parent filter, 106**

parent() method, 176-180

parent nodes, 86

parentheses (()), 459

parents() method, 176

parentsUntil() method, 176

parseJSON() method, 306, 316-319

parsing XML data, 312-315

passing on data, 266

:password filter, 119

paths, DOM trees, 54

patterns, load() method, 323

plain text, server requests, 307

platforms (Mobile), 419-420

plug-ins, 2, 393

animation, 283, 411

availability, 393

customizing, 405-414

documentation, 394

existing, 394-397

iPhone, 422

libraries, 399

methods, 447

options, 410-414

publishing, 415-416

rules, 409-411

validation, 397-405

pop-ups, 377

position() method, 224-228

positioning

configuring, 231-235

icons, 431

methods, 223-235

offset() method, 228-235

toolbars, 436

widgets, 347

post() method, 305

preceding

nodes, 86

sibling nodes, 86

premade designs, 347

prepend() method, 138-142

prependTo() method, 143-148

presets, form field values, 136

prevAll() method, 182

preventDefault() method, 256-258, 265

prev() method, 182

prev + next selector, 88

prev ~ siblings selector, 88

preventing

 caching, 307

 redirection, 257

processing JSON, 316

progress bars, 380

Progressbar widget, 346

programming

 namespaces, 54

 OOP, 46

 on the WWW, 32

prologs (XML), 40

prop() function, 80

properties

 configuring, 207-209

 context, 75

 CSSs, 351

 event objects, 253-256

 innerHTML, 66

 naming, 207

 nodeName, 76

 selectors, 75

 styles, 206-207

 tables, 187

 UI components, 359-362

 XMLHttpRequest, 300

protocols, HTTP, 32, 425

prototyping, 463

pseudo classes, 50

publishing plug-ins, 415-416

Q

queue() method, 78, 281

queues, 78

animation, 281

callbacks, 331

options, 290

quotation marks ("), 42, 56

R

:radio filter, 119

rangelength() method, 403

rates, animation, 280

ready() method, 20, 36, 62, 74, 201, 435

readyState property, 300

rectangles, 287. See also animation; objects

recursive function calls, 461

redirection, preventing, 257

reducing elements, animation, 25-28

references

functions, 460

jQuery libraries, 36-37

scripts, 4

UIs, 354

registering callback functions (AJAX), 304

relationships

data binding, 251-252

JavaScript to jQuery, 33-37

siblings, 182-184

remote requests (AJAX), 304

remove() method, 166-172

removeAttr() method, 166-172

removeAttrs() method, 403

removeClass() method, 24, 218-221

removeProp() function, 80

removing. See also deleting

CSS classes, 218

elements, 166-172

nodes, 167

replaceAll() method, 164-166

replaceWith() method, 160-164

replacing

elements, 159-166

separator lines, 165

requests

AJAX, 303-304

methods, 305

examples, 300-302

JSON, 316

JSONP extension, 339

loading, 307-315

Twitter, 317-319

XMLHttpRequest, 297-304

require() method, 37

required() method, 403

:reset filter, 119

Resig, John, 2

Resizable component, 371

resize() method, 260

resizeable() method, 346

resizing div containers, 371

responseText property, 300

return keyword, 459

RIAs (Rich Internet Applications), 1, 34, 297

Mobile. *See* Mobile

UIs, 346

role system (Mobile), 422

rules

components, 358-359

CSS, 434

descendants, 93, 184

Download Builder, 350

fields, 404

namespaces, 55

options, 402

plug-ins, 409-411

style, 49. *See also* CSSs

validation, 399, 403

widgets, 358-359

XML, 303
XMLHttpRequest objects, 299
rules() method, 403

S

schemas, CSS declarations, 50
screen object, 466
<script> tag, 34-36, 304
scripts
AJAX, 320-322
cross-over, 304
references, 4
scroll events, 449
scroll() method, 260
scrolling methods, 236-238
ScrollLeft method, 236
scrollstart event, 449
scrollstop event, 449
scrollTop() method, 236
searching
descendants, 184
existing plug-ins, 394-397
web pages, 176-184
select() method, 260
selectable() method, 346, 372
:selected filter, 121
selecting
dates, 377
themes (ThemeRoller), 352
selector1, 88
selector2, 88
selectorN, 88
selectors, 50, 83
examples, 87-97
filters, 99-123
hierarchies, 85-99
overview of, 84-85
properties, 75
testing, 90

send() method, 299, 304
sending
form data, 448
JSON data to forms, 339
objects, 309
separator lines, replacing, 165
serialization (AJAX), 327-329
serializeArray() method, 329
serialize() method, 327-328
servers
AJAX requests, 309-312
data types, 305-307
plain text requests, 307
web, 32
setMimeType() method, 300
setRequestHeader() method, 300
setters, 362
show() method, 283-284
showing. See viewing
opacity, 288
sliding effects, 284
siblings, 182-184
size() method, 464
sizing
effects, 284-287
images, 241
specifications, 244
slice() method, 124
slideDown() method, 284-287
sliders, 380-381
dragging, 367
UIs, 365
values, 365
Slider widget, 346
slideToggle() method, 26, 102, 284-287
slideUp() method, 284-287
sliding effects, 284-287
sortable() method, 346, 372
source code, 406. See also code

spacing

- elements, 214

- images, 237

span areas, 94-96, 155

specialEasing option, 290

special effects, strings, 283

special events, 448-451. See also events

specifications

- class names, 210

- positioning, 233

- of script languages, 35

- selectors, 84

- sizing, 244

specifying

- callbacks, 280

- data types, 305-307

- filters (AJAX), 323-327

speed of animation, 279-280

spinner components, 384

square brackets ([]), 42

standardization

- Mobile, 418

- objects, 309

start tags (XML), 39

statements

- DOCTYPE, 21, 422

- HTML, 422

static functions in objects, collecting, 409

status property, 300

step option, 290

stopping animation, 282

stopPropagation() method, 265

strings, 283, 458

structures, 55-57

- JSON, 42

- modularized web applications, 63-65

- UIs, 354

- web pages, 152, 422-424. *See also* web pages

style sheets. See also CSSs

- classes

- adding, 210-217

- testing, 221-222

- toggling, 219-221

- css() method, 206-209

- DHTML, 48-50

- elements, 209-222

- formatting, 205-206

- height()/width() methods, 239-242

- inner/outer dimensions, 242-244

- methods

- positioning, 223-235

- scrolling, 236-238

- offset() method, 228-235

- removeClass() method, 218-221

styles

- attributes, 357

- objects, 466

- properties, 206-207

:submit filter, 119

submit() method, 260

success event, 330

support

- AJAX, 304-307

- browsers, 34, 250

- Mobile, 419

- UI interaction, 346

swipe event, 448

swipeleft event, 448

swiperight event, 448

Symbian, 419

symbols

- buttons, 430

- Unicode, 40

synchronization, 380

syntax

- CSS, 50

- JSON, 42

- XML, 38

T

tables

- formatting, 325
- properties, 187

tabs, 366, 381-384**Tabs widget, 347****tags**

- <head>, 37
- HTML, 134. *See also* HTML; web pages
 - AJAX, 302
 - event handlers, 248
- <link>, 49
- <script>, 304
- XML, 38-41. *See also* XML

tap event, 448**taphold event, 448****templates**

- dates, 203
- UIs, 355
- web pages, 376

temporal intervals, 26**testing**

- classes, 221-222
- nodes, 85
- selectors, 90

text

- AJAX communication, 301-303
- inserting, 139
- JSON, 41. *See also* JSON
- server requests, 307
- XML, 38. *See also* XML

:text filter, 119**text() method, 132-135, 162****Theme framework, 347****ThemeRoller tool, 347, 351-353, 359****themes, 434**

- CSS links, 350
- default, 350
- Gallery (ThemeRoller), 352
- Mobile, 452-443

time of animation, 279-280**Tizen, 419****toggle() method, 275, 287****toggleClass() method, 25, 219, 263****toggleing classes, 219-221****tokens (XPath), 86****toolbars (Mobile), 435-438****toolkits, 1****tools**

- DOM Inspector, 53
- ThemeRoller, 347, 351-353, 359
- UIs, 385

Tooltip widget, 347, 384**top values, position() method, 224****touch events, 448****transforming web pages, 425****transitions, 428, 433****transports, 342****trees**

- DOM, 53, 313
- navigating, 85

treeview() method, 397**triggerHandler() method, 269-270****trigger() method, 267-269****triggers, events, 252****tweets, 317. *See also* Twitter****Twitter**

- Mobile, 439
- requests, 317-319

types

- of animation, 282-283
- button, 376
- data, 305-307. *See also* data types
- div, 57
- jqXHR, 57
- of nodes, 85
- of selectors, 50
- url, 315

U

Uls (user interfaces), 5, 345

- applying, 349-355
- components, 355-369
 - events, 366-369
 - interaction, 370-372
 - methods, 363-346
 - properties, 359-362
 - rules, 358-359
- default UI settings, 356-358
- effects, 385-387
- examples, 348
- extended effects, 347
- overview of, 345-348
- references, 354
- sliders, 365
- support, 346
- templates, 355
- utilities, 385
- web pages, 353-355
- websites, 387-391
- widgets, 346-347, 372-384

unbind() method, 262-266**undelagate() method, 274****Unicode symbols, 40****universal selectors, 50****unload() method, 260****unwrap() method, 159****unwrapping elements, 159****uppercase (XML), 40****url() method, 403****url type, 315****user interfaces. See Uls****utilities, 385. See also tools**

V

valid() method, 403**validate() method, 399, 402-403****validating plug-ins, 397-405****val() method, 135-137****values**

- AJAX, 330
- Boolean, 458
- css() method, 207
- options, 361
- position() method, 224
- sliders, 365

variables, 410, 457**verifying access, 310-312****versions**

- Mobile, 417
- new core techniques, 79

viewing

- DOM trees, 54
- tabs, 366
- web pages, 450

visibility filters, 109-111

W

Web 2.0, 5, 33**web applications**

- AJAX, 38. *See also* AJAX
- modularized structures, 63-65

web browsers, 32**web pages, 395. See also plug-ins**

- add() method, 193-196
- attr() method, 137
- blocking, 379
- content, 154-159
- date examples, 196-203
- editing, 22-25
- elements
 - accessing, 47, 52-54, 131
 - cloning, 172-175
 - finding, 184-185
 - inserting, 66-70
 - replacing, 159-166
- form content, 135-137
- history.back() method, 426-428

loading, 144

Mobile

linking, 424-428

structures, 422-424

nodes

inserting, 137-153

modifying, 132-135

objects, 186-193

searching, 176-184

style sheets, 49, 205-206

templates, 376

transforming, 425

transition effects, 428

UIs, 353-355

web servers, 32. See also servers

websites

JavaScript, 34-37

Mobile, 417

<script> containers in, 34-36

UIs, 387-391

widgets

events, 366-369

menus, 384

Mobile,

rules, 358-359

toolbars. *See* toolbars

UIs, 346-347, 372-384

width() method, 239-242

window object, 466

Windows Phone/Mobile, 419

wrapAll() method, 155-157

wrapInner() method, 158

wrap() method, 154-155

wrapping

areas, 158

content, 154-159

elements, 70-72

writing conventions, 5-6

WWW (World Wide Web), 32

X-Z

XDomainRequest object, 298

XHTML (Extensible HTML), 32, 346

XML (Extensible Markup Language), 4

file requests, 313

namespaces, 54

overview of, 38-41

parsing, 312-315

XMLHttpRequest

AJAX, 297-304

formatting, 298-299

methods, 299-300

properties, 300

XMLHttpRequest objects, 38

XPath, 85