Phil Ballard
Michael Moncur

Fifth Edition
Covers JavaScript 1.8+, Ajax, and jQuery

Sams Teach Yourself

JavaScript™ in 24 Hours

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Phil Ballard
Michael Moncur

Sams **Teach Yourself**

JavaScript™

Fifth Edition

in 24 Hours

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

Sams Teach Yourself JavaScript™ in 24 Hours, Fifth Edition

Copyright © 2013 by Pearson Education, Inc.

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: 978-0-672-33608-9
ISBN-10: 0-672-33608-1

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

Printed in the United States of America

First Printing October 2012

Trademarks

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

Warning and Disclaimer

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

Bulk Sales

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

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

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

International Sales
international@pearsoned.com

Editor-in-Chief
Mark Taub

Acquisitions Editor
Mark Taber

Managing Editor
Kristy Hart

Project Editor
Anne Goebel

Copy Editor
Geneil Breeze

Indexer
Erika Millen

Proofreader
Chrissy White,
Language Logistics

Publishing Coordinator
Vanessa Evans

Technical Editor
Joseph Greenspan

Cover Designer
Anne Jones

Compositor
Nonie Ratcliff

Contents at a Glance

Introduction **1**

Part I: First Steps with JavaScript

HOUR 1 **Introducing JavaScript** **7**

HOUR 2 **Writing Simple Scripts** **21**

HOUR 3 **Using Functions** **37**

HOUR 4 **Objects and Built-In Objects** **49**

HOUR 5 **Different Types of Data** **67**

Part II: More Advanced JavaScript

HOUR 6 **Scripts That Do More** **85**

HOUR 7 **Object Oriented Programming** . . . **103**

HOUR 8 **Meet JSON** **121**

HOUR 9 **Responding to Events** **135**

HOUR 10 **JavaScript and Cookies** **155**

Part III: Working with the Document Object Model (DOM)

HOUR 11 **Navigating the DOM** **171**

HOUR 12 **Scripting the DOM** **185**

HOUR 13 **JavaScript and CSS** **201**

HOUR 14 **Good Coding Practice** **217**

HOUR 15 **Graphics and Animation** **235**

Part IV: Ajax

HOUR 16 **Introducing Ajax** **251**

HOUR 17 **Creating a Simple Ajax Library** **273**

HOUR 18 **Solving Ajax Problems** **287**

Part V: Using JavaScript Libraries

HOUR 19 **Making Life Easier with Libraries** **303**

HOUR 20 **A Closer Look at jQuery** **317**

HOUR 21 **The jQuery UI User Interface Library** **335**

Part VI: Using JavaScript with Other Web Technologies

HOUR 22 **JavaScript and Multimedia** **353**

HOUR 23 **HTML5 and JavaScript** **365**

HOUR 24 **JavaScript Beyond the Web Page** **383**

Part VII: Appendices

APPENDIX A **Tools for JavaScript Development** **399**

APPENDIX B **JavaScript Quick Reference** **403**

Index **411**

Table of Contents

Introduction **1**
Who This Book Is For ... 1
The Aims of This Book ... 1
Conventions Used ... 2
Q&A, Quiz, and Exercises ... 2
How the Book Is Organized ... 2
Tools You'll Need ... 3

PART I: **First Steps with JavaScript** **5**

HOUR 1: **Introducing JavaScript** **7**
Web Scripting Fundamentals ... 7
Server- Versus Client-Side Programming ... 8
JavaScript in a Nutshell ... 8
Where JavaScript Came From ... 9
The <script> Tag ... 10
Introducing the DOM ... 11
Talking to the User ... 13
Summary ... 17
Q&A ... 18
Workshop ... 18
Exercises ... 19

HOUR 2: **Writing Simple Scripts** **21**
Including JavaScript in Your Web Page ... 21
JavaScript Statements ... 23
Variables ... 24
Operators ... 26
Capturing Mouse Events ... 30
Summary ... 33
Q&A ... 34
Workshop ... 34
Exercises ... 35

HOUR 3: **Using Functions** **37**
General Syntax ... 37
Calling Functions ... 38
Arguments ... 40
Returning Values from Functions ... 43
Scope of Variables ... 44
Summary ... 46
Q&A ... 47
Workshop ... 47
Exercises ... 48

HOUR 4: **DOM Objects and Built-In Objects** **49**
Interacting with the User ... 49
Selecting Elements by Their id ... 51
Accessing Browser History ... 52
Using the location Object ... 53
Browser Information—The navigator Object ... 54
Dates and Times ... 57
Simplifying Calculation with the Math Object ... 59
Summary ... 64
Q&A ... 65
Workshop ... 65
Exercises ... 66

HOUR 5: **Different Types of Data** **67**
Numbers ... 67
Strings ... 70
Boolean Values ... 73
Arrays ... 74
Summary ... 79
Q&A ... 80
Workshop ... 80
Exercises ... 81

PART II: **More Advanced JavaScript** **83**

HOUR 6: **Scripts That Do More** **85**
Conditional Statements ... 85
Loops and Control Structures ... 90
Debugging Your Scripts ... 94
Summary ... 100
Q&A ... 101
Workshop ... 101
Exercises ... 102

HOUR 7: **Object Oriented Programming** . . **103**
What Is Object Oriented Programming (OOP)? ... 103
Object Creation ... 104
Extending and Inheriting Objects Using prototype ... 111
Encapsulation ... 115
Using Feature Detection ... 116
Summary ... 118
Q&A ... 119
Workshop ... 119
Exercises ... 120

HOUR 8: **Meet JSON** **121**
What Is JSON? 121
Accessing JSON Data 123
Data Serialization with JSON 124
JSON Data Types 126
Simulating Associative Arrays 127
Creating Objects with JSON 127
JSON Security 131
Summary 132
Q&A 133
Workshop 133
Exercises 134

HOUR 9: **Responding to Events** **135**
Understanding Event Handlers 135
Default Actions 138
The event Object 141
Cross-Browser Event Handlers 142
Advanced Event Handler Registration 146
Summary 151
Q&A 152
Workshop 152
Exercises 153

HOUR 10: **JavaScript and Cookies** **155**
What Are Cookies? 155
The document.cookie Property 156
Cookie Ingredients 157
Writing a Cookie 158
A Function to Write a Cookie 159
Reading a Cookie 161
Deleting Cookies 162
Setting Multiple Values in a Single Cookie 165
Summary 166
Q&A 166
Workshop 167
Exercises 168

PART III: **Working with the Document Object Model (DOM)** **169**

HOUR 11: **Navigating the DOM** **171**
DOM Nodes 171
Selecting Elements with getElementsByTagName() 177
Reading an Element's Attributes 179
Mozilla's DOM Inspector 180
Summary 181
Q&A 182
Workshop 182
Exercises 183

HOUR 12: **Scripting the DOM** **185**
Creating New Nodes 185
Manipulating Child Nodes 187
Editing Element Attributes 191
Dynamically Loading JavaScript Files 192
Summary 198
Q&A 199
Workshop 199
Exercises 200

HOUR 13: **JavaScript and CSS** **201**
A Ten Minute CSS Primer 201
The DOM style Property 204
Accessing Classes Using className 207
The DOM styleSheets Object 209
Summary 214
Q&A 215
Workshop 215
Exercises 216

HOUR 14: **Good Coding Practice** **217**
Don't Overuse JavaScript 217
Writing Readable and Maintainable Code 218
Graceful Degradation 221
Progressive Enhancement 222
Unobtrusive JavaScript 223
Feature Detection 226
Handling Errors Well 227
Summary 231
Q&A 232
Workshop 232
Exercises 233

HOUR 15: **Graphics and Animation** **235**
Preloading Images 235
Animating Page Elements 236
Animating Transparency 238
CSS3 Transitions, Transformations, and Animations 239
Scripting DOM Positioning 240
Optimizing Performance 242
Summary 245

Q&A ... 246
Workshop ... 246
Exercises ... 247

PART IV: **Ajax** ... **249**

HOUR 16: **Introducing Ajax** ... **251**
The Anatomy of Ajax ... 251
The XMLHttpRequest Object ... 256
Creating Instances of XMLHttpRequest ... 256
Sending the Server Request ... 260
Monitoring Server Status ... 263
Callback Functions ... 264
responseText and responseXML Properties ... 265
Summary ... 269
Q&A ... 270
Workshop ... 270
Exercises ... 271

HOUR 17: **Creating a Simple Ajax Library** ... **273**
An Ajax Library ... 273
Implementing the Library ... 274
Using the Library ... 278
Summary ... 284
Q&A ... 285
Workshop ... 285
Exercises ... 286

HOUR 18: **Solving Ajax Problems** ... **287**
Debugging Ajax Applications ... 287
Common Ajax Errors ... 292
Some Programming Gotchas ... 297
Summary ... 298
Q&A ... 299
Workshop ... 299
Exercises ... 300

PART V: **Using JavaScript Libraries** ... **301**

HOUR 19: **Making Life Easier with Libraries** ... **303**
Why Use a Library? ... 303
What Sorts of Things Can Libraries Do? ... 304
Some Popular Libraries ... 305
Introducing prototype.js ... 306
Summary ... 313
Q&A ... 314
Workshop ... 314
Exercises ... 315

HOUR 20: **A Closer Look at jQuery** ... **317**
Including jQuery in Your Pages ... 317
jQuery's $(document).ready Handler ... 318
Selecting Page Elements ... 319
Working with HTML Content ... 320
Showing and Hiding Elements ... 321
Animating Elements ... 322
Command Chaining ... 324
Handling Events ... 328
Using jQuery to Implement Ajax ... 328
Summary ... 332
Q&A ... 333
Workshop ... 333
Exercises ... 334

HOUR 21: **The jQuery UI User Interface Library** ... **335**
What jQuery UI Is All About ... 335
How to Include jQuery UI in Your Pages ... 336
Interactions ... 337
Using Widgets ... 343
Summary ... 348
Q&A ... 349
Workshop ... 349
Exercises ... 350

PART VI: **Using JavaScript with Other Web Technologies** ... **351**

HOUR 22: **JavaScript and Multimedia** ... **353**
Multimedia Formats ... 353
Browser Plug-Ins ... 355
Using an Anchor Tag ... 356
Using <embed> and <object> ... 356
Flash ... 358
Summary ... 362
Q&A ... 363
Workshop ... 363
Exercises ... 364

HOUR 23: **HTML5 and JavaScript** ... **365**
New Markup for HTML5 ... 365
Some Important New Elements ... 366
Drag and Drop ... 373
Local Storage ... 376

Working with Local Files 377
Summary 379
Q&A 380
Workshop 380
Exercises 381

HOUR 24: **JavaScript Beyond the Web Page** **383**
JavaScript Outside the Browser 383
Writing Google Chrome Extensions 384
Summary 394
Q&A 395
Workshop 395
Exercises 396

PART VII: **Appendices** **397**

APPENDIX A: **Tools for JavaScript Development** **399**
Editors 399
Validators 400
Debugging and Verifying Tools 401

APPENDIX B: **JavaScript Quick Reference** **403**

Index **411**

About the Authors

Phil Ballard, the author of *Sams Teach Yourself Ajax in 10 Minutes*, graduated in 1980 with an honors degree in electronics from the University of Leeds, England. Following an early career as a research scientist with a major multinational, he spent a few years in commercial and managerial roles within the high technology sector, later working full time as a software engineering consultant.

Operating as "The Mouse Whisperer" (www.mousewhisperer.co.uk), Ballard has spent recent years involved solely in website and intranet design and development for an international portfolio of clients.

Michael Moncur is a freelance webmaster and author. He runs a network of websites, including the Web's oldest site about famous quotations, online since 1994. He wrote *Sams Teach Yourself DHTML in 24 Hours* and has also written several bestselling books about networking, certification programs, and databases. He lives with his wife in Salt Lake City.

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.

We welcome your comments. You can email or write 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 better.

Please note that we cannot help you with technical problems related to the topic of this book.

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

Email: consumer@samspublishing.com

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

Reader Services

Visit our website and register this book at www.informit.com/title/9780672336089 for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

Who This Book Is For

If you're interested in learning JavaScript, chances are that you've already gained at least a basic understanding of HTML and web page design in general and want to move on to adding some extra interactivity to your pages. Or maybe you currently code in another programming language and want to see what additional capabilities JavaScript can add to your armory.

If you've never tinkered with HTML at all, nor done any computer programming, it would be helpful to browse through an HTML primer before getting into the book. Don't worry—HTML is very accessible, and you don't need to be an HTML expert to start experimenting with the JavaScript examples in this book.

JavaScript is an ideal language to use for your first steps in programming, and in case you get bitten by the bug, pretty much all of the fundamental concepts that you learn in JavaScript will later be applicable in a wide variety of other languages such as C, Java, or PHP.

The Aims of This Book

When JavaScript was first introduced, it was somewhat limited in what it could do. With basic features and rather haphazard browser support, it gained a reputation in some quarters as being something of a toy or gimmick. Now, due to much better browser support for W3C standards and improvement in the JavaScript implementations used in recent browsers, JavaScript is finally beginning to be treated as a serious programming language.

Many advanced programming disciplines used in other programming languages can readily be applied to JavaScript, for example, object oriented programming promotes the writing of solid, readable, maintainable, and reusable code.

So-called "unobtrusive" scripting techniques and the use of DOM scripting focus on adding interaction to web pages while keeping the HTML simple to read and well separated from the program code.

This book aims to teach the fundamental skills relevant to all of the important aspects of JavaScript as it's used today. In the course of the book, you start from basic concepts and gradually learn the best practices for writing JavaScript programs in accordance with current web standards.

Conventions Used

All of the code examples in the book are written to validate correctly as HTML5. In the main, though, the code avoids using HTML5-specific syntax because at the time of writing its support in web browsers is still not universal. The code examples should work correctly in virtually any recent web browser, regardless of the type of computer or operating system.

In addition to the main text of each lesson, you will find a number of boxes labeled as Notes, Tips, and Cautions.

NOTE

These sections provide additional comments that might help you to understand the text and examples.

TIP

These blocks give additional hints, shortcuts, or workarounds to make coding easier.

CAUTION

Avoid common pitfalls by using the information in these blocks.

TRY IT YOURSELF

Each hour contains at least one section that walks you through the process of implementing your own script. This will help you to gain confidence in writing your own JavaScript code based on the techniques you've learned.

Q&A, Quiz, and Exercises

After each hour's lesson, you'll find three final sections.

Q&A tries to answer a few of the more common questions about the hour's topic.

The Quiz tests your knowledge of what you learned in that lesson.

Exercises offer suggestions for further experimentation, based on the lesson, that you might like to try on your own.

How the Book Is Organized

The book is divided into six parts, gradually increasing in the complexity of the techniques taught.

- **Part I—First Steps with JavaScript**

  Part I is an introduction to the JavaScript language and how to write simple scripts using the language's common functions. This part of the book is aimed mainly at readers with little or no prior programming knowledge and no knowledge of the JavaScript language.

- **Part II—More Advanced JavaScript**

  Here more sophisticated programming paradigms are introduced, such as program control loops and event handling, object oriented programming, JSON notation, and cookies.

- **Part III—Working with the Document Object Model (DOM)**

  This part of the book concentrates on navigating and editing the DOM (Document Object Model) tree, using CSS stylesheets, and styling and animating page elements. There is emphasis on using good coding practice such as unobtrusive JavaScript.

- **Part IV—Ajax**

  Here you learn how to make background calls to the server using the `XMLHTTPRequest` object and handle the server responses, build a simple Ajax library, and learn about debugging Ajax applications.

- **Part V—Using JavaScript Libraries**

  In this part, you learn how to simplify cross-browser development using third-party libraries such as Prototype and jQuery.

- **Part VI—Using JavaScript with Other Web Technologies**

  In the final part examples are given of how to use JavaScript to control multimedia, exploit HTML5 capabilities, write browser add-ons, and more.

Tools You'll Need

Writing JavaScript does not require any expensive and complicated tools such as Integrated Development Environments (IDEs), compilers, or debuggers.

The examples in this book can all be created in a text editing program, such as Windows' Notepad. At least one such application ships with just about every operating system, and countless more are available for no or low cost via download from the Internet.

NOTE

Appendix A, "Tools for JavaScript Development," lists some additional easily obtainable tools and resources for use in JavaScript development.

To see your program code working, you'll need a web browser such as Internet Explorer, Mozilla Firefox, Opera, Safari, or Google Chrome. It is recommended that you upgrade your browser to the latest current stable version.

The vast majority of the book's examples do not need an Internet connection to function. Simply storing the source code file in a convenient location on your computer and opening it with your chosen browser is generally sufficient. The exceptions to this are the hour on cookies and the section of the book about Ajax; to explore all of the example code will require a web connection (or a connection to a web server on your Local Area Network) and a little web space in which to post the example code. If you've done some HTML coding, you may already have that covered; if not, a hobby-grade web hosting account costs very little and will be more than adequate for trying out the examples in this book. (Check that your web host allows you to run scripts written in the PHP language if you want to try out the Ajax examples in Part IV. Nearly all hosts do).

HOUR 3
Using Functions

WHAT YOU'LL LEARN IN THIS HOUR

- How to define functions
- How to call (execute) functions
- How functions receive data
- Returning values from functions
- About the scope of variables

Commonly, programs carry out the same or similar tasks repeatedly during the course of their execution. For you to avoid rewriting the same piece of code over and over again, JavaScript has the means to parcel up parts of your code into reusable modules, called *functions*. After you've written a function, it is available for the rest of your program to use, as if it were itself a part of the JavaScript language.

Using functions also makes your code easier to debug and maintain. Suppose you've written an application to calculate shipping costs; when the tax rates or haulage prices change, you'll need to make changes to your script. There may be 50 places in your code where such calculations are carried out. When you attempt to change every calculation, you're likely to miss some instances or introduce errors. However, if all such calculations are wrapped up in a few functions used throughout the application, then you just need to make changes to those functions. Your changes will automatically be applied all through the application.

Functions are one of the basic building blocks of JavaScript and will appear in virtually every script you write. In this hour you see how to create and use functions.

General Syntax

Creating a function is similar to creating a new JavaScript command that you can use in your script.

Here's the basic syntax for creating a function:

```
function sayHello() {
    alert("Hello");
    // ... more statements can go here ...
}
```

CAUTION

The keyword `function` must always be used in lowercase, or an error will be generated.

You begin with the keyword `function`, followed by your chosen function name with parentheses appended, then a pair of curly braces {}. Inside the braces go the JavaScript statements that make up the function. In the case of the preceding example, we simply have one line of code to pop up an `alert` dialog, but you can add as many lines of code as are necessary to make the function…well, function!

To keep things tidy, you can collect together as many functions as you like into one `<script>` element:

```
<script>
    function doThis() {
        alert("Doing This");
    }
    function doThat() {
        alert("Doing That");
    }
</script>
```

TIP

Function names, like variable names, are case-sensitive. A function called `MyFunc()` is different from another called `myFunc()`. Also, as with variable names, it's really helpful to the readability of your code to choose meaningful function names.

Calling Functions

Code wrapped up in a function definition will not be executed when the page loads. Instead, it waits quietly until the function is *called*.

To call a function, you simply use the function name (with the parentheses) wherever you want to execute the statements contained in the function:

```
sayHello();
```

TIP

You've already seen numerous examples of using the *methods* associated with JavaScript objects, such as `document.write()` and `window.alert()`.

Methods are simply functions that "belong" to a specific object. You learn much more about objects in Hour 4, "DOM Objects and Built-In Objects."

For example, you may want to add a call to your new function `sayHello()` to the `onClick` event of a button:

```
<input type="button" value="Say Hello" onclick="sayHello()" />
```

Putting JavaScript Code in the Page `<head>`

Up to now, our examples have all placed the JavaScript code into the `<body>` part of the HTML page. Using functions lets you employ the much more common, and usually preferable, practice of storing our JavaScript code in the `<head>` of the page. Functions contained within a `<script>` element in the page head, or in an external file included via the `src` attribute of a `<script>` element in the page head, are available to be called from anywhere on the page. Putting functions in the document's head section ensures that they have been defined prior to any attempt being made to execute them.

Listing 3.1 has an example.

LISTING 3.1 Functions in the Page Head

```
<!DOCTYPE html>
<html>
<head>
    <title>Calling Functions</title>
    <script>
        function sayHello() {
            alert("Hello");
        }
    </script>
</head>
<body>
    <input type="button" value="Say Hello" onclick="sayHello()" />
</body>
</html>
```

In this listing, you can see that the function definition itself has been placed inside a `<script>` element in the page head, but the call to the function has been made from a different place entirely—on this occasion, from the `onClick` event handler of a button in the body section of the page.

The result of clicking the button is shown in Figure 3.1.

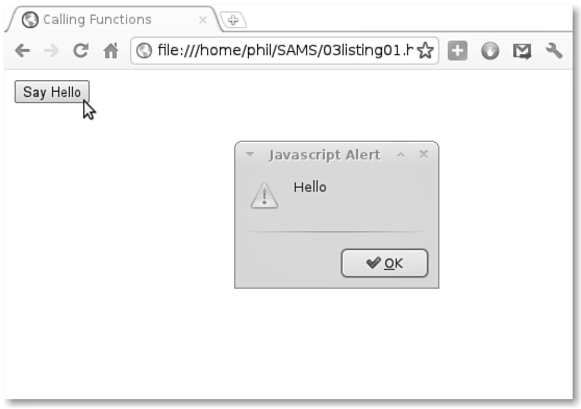


FIGURE 3.1
Calling a JavaScript function

Arguments

It would be rather limiting if your functions could only behave in an identical fashion each and every time they were called, as would be the case in the preceding example.

Fortunately, you can extend the capabilities of functions a great deal by passing data to them. You do this when the function is called, by passing to it one or more *arguments*:

```
functionName(arguments)
```

Let's write a simple function to calculate the cube of a number and display the result:

```
function cube(x) {
    alert(x * x * x);
}
```

Now we can call our function, replacing the variable *x* with a number. Calling the function like this

```
cube(3);
```

results in a dialog box being displayed that contains the result of the calculation, in this case 27.

NOTE

You'll sometimes see or hear the word *parameters* used in place of arguments, but it means exactly the same thing.

Of course, you could equally pass a variable name as an argument. The following code would also generate a dialog containing the number 27:

```
var length = 3;
cube(length);
```

Multiple Arguments

CAUTION

Make sure that your function calls contain enough argument values to match the arguments specified in the function definition. If any of the arguments in the definition are left without a value, JavaScript may issue an error, or the function may perform incorrectly. If your function call is issued with too many arguments, the extra ones will be ignored by JavaScript.

Functions are not limited to a single argument. When you want to send multiple arguments to a function, all you need to do is separate them with commas:

```
function times(a, b) {
    alert(a * b);
}
times(3, 4); // alerts '12'
```

You can use as many arguments as you want.

It's important to note that the names given to arguments in the definition of your function have nothing to do with the names of any variables whose values are passed to the function. The variable names in the argument list act like placeholders for the actual values that will be passed when the function is called. The names that you give to arguments are only used inside the function definition to specify how it works.

We talk about this in more detail later in the hour when we discuss variable *scope*.

TRY IT YOURSELF ▼

A Function to Output User Messages

Let's use what we've learned so far in this hour by creating a function that can send the user a message about a button he or she has just clicked. We place the function definition in the `<head>` section of the page and call it with multiple arguments.

Here's our function:

```
function buttonReport(buttonId, buttonName, buttonValue) {
    // information about the id of the button
    var userMessage1 = "Button id: " + buttonId + "\n";
    // then about the button name
    var userMessage2 = "Button name: " + buttonName + "\n";
    // and the button value
    var userMessage3 = "Button value: " + buttonValue;
    // alert the user
    alert(userMessage1 + userMessage2 + userMessage3);
}
```

The function `buttonReport` takes three arguments, those being the `id`, `name`, and `value` of the button element that has been clicked. With each of these three pieces of information, a short message is constructed. These three messages are then concatenated into a single string, which is passed to the `alert()` method to pop open a dialog containing the information.

To call our function, we put a button element on our HTML page, with its `id`, `name`, and `value` defined:

```
<input type="button" id="id1" name="Button 1" value="Something" />
```

We need to add an `onClick` event handler to this button from which to call our function. We're going to use the `this` keyword, as discussed in Hour 2, "Writing Simple Scripts":

```
onclick = "buttonReport(this.id, this.name, this.value)"
```

TIP

You may have noticed that the first two message strings have an element `"\n"` appended to the string; this is a "new line" character, forcing the message within the alert dialog to return to the left and begin a new line. Certain special characters like this one must be prefixed with \ if they are to be correctly interpreted when they appear in a string. Such a prefixed character is known as an *escape sequence*. You learn more about escape sequences in Hour 5, "Different Types of Data."

▼ TRY IT YOURSELF

A Function to Output User Messages

continued

The complete listing is shown in Listing 3.2.

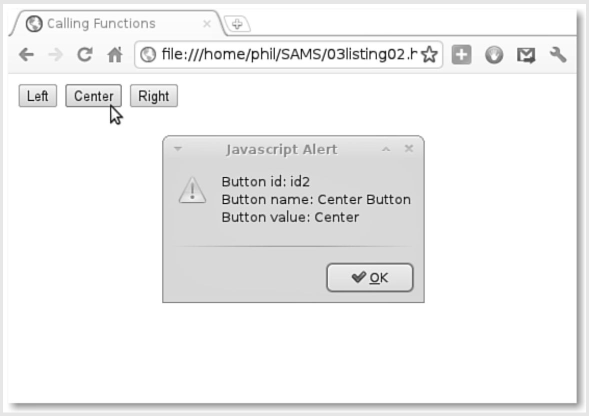
LISTING 3.2 Calling a Function with Multiple Arguments

```
<!DOCTYPE html>
<html>
<head>
    <title>Calling Functions</title>
    <script>
        function buttonReport(buttonId, buttonName, buttonValue) {
            // information about the id of the button
            var userMessage1 = "Button id: " + buttonId + "\n";
            // then about the button name
            var userMessage2 = "Button name: " + buttonName + "\n";
            // and the button value
            var userMessage3 = "Button value: " + buttonValue;
            // alert the user
            alert(userMessage1 + userMessage2 + userMessage3);
        }
    </script>
</head>
<body>
    <input type="button" id="id1" name="Left Hand Button" value="Left"
➥onclick = "buttonReport(this.id, this.name, this.value)"/>
    <input type="button" id="id2" name="Center Button" value="Center"
➥onclick = "buttonReport(this.id, this.name, this.value)"/>
    <input type="button" id="id3" name="Right Hand Button" value="Right"
➥onclick = "buttonReport(this.id, this.name, this.value)"/>
</body>
</html>
```

Use your editor to create a file buttons.html and enter the preceding code. You should find that it generates output messages like the one shown in Figure 3.2, but with different message content depending on which button has been clicked.

TRY IT YOURSELF ▼

A Function to Output User Messages
continued



```
Button id: id2
Button name: Center Button
Button value: Center
```

FIGURE 3.2 Using a function to send messages

Returning Values from Functions

Okay, now you know how to pass information to functions so that they can act on that information for you. But how can you get information back from your function? You won't always want your functions to be limited to popping open a dialog box!

Luckily, there is a mechanism to collect data from a function call—the *return value*. Let's see how it works:

```
function cube(x) {
    return x * x * x;
}
```

Instead of using an `alert()` dialog within the function, as in the previous example, this time we prefixed our required result with the `return` keyword. To access this value from outside the function, we simply assign to a variable the value *returned* by the function:

```
var answer = cube(3);
```

The variable `answer` will now contain the value 27.

NOTE

The values returned by functions are not restricted to numerical quantities as in this example. In fact, functions can return values having any of the data types supported by JavaScript. We discuss data types in Hour 5.

TIP

Where a function returns a value, we can use the function call to pass the return value directly to another statement in our code. For example, instead of

```
var answer = cube(3);
alert(answer);
```

we could simply use

```
alert(cube(3));
```

The value of 27 returned from the function call `cube(3)` immediately becomes the argument passed to the `alert()` method.

Scope of Variables

We have already seen how to declare variables with the `var` keyword. There is a golden rule to remember when using functions:

"Variables declared inside a function only exist inside that function."

This limitation is known as the *scope* of the variable. Let's see an example:

```
// Define our function addTax()
function addTax(subtotal, taxRate) {
    var total = subtotal * (1 + (taxRate/100));
    return total;
}
// now let's call the function
var invoiceValue = addTax(50, 10);
alert(invoiceValue); // works correctly
alert(total);  // doesn't work
```

If we run this code, we first see an `alert()` dialog with the value of the variable `invoiceValue` (which should be 55 but in fact will probably be something like 55.000000001 as we have not asked JavaScript to round the result).

We will not, however, then see an `alert()` dialog containing the value of the variable `total`. Instead, JavaScript simply produces an error. Whether you see this error reported depends on your browser settings—we learn more about error handling later in the book—but JavaScript will be unable to display an `alert()` dialog with the value of your variable `total`.

This is because we placed the declaration of the variable `total` *inside* the `addTax()` function. Outside the function the variable `total` simply doesn't exist (or, as JavaScript puts it, "is not defined"). We used the `return` keyword to pass back just the *value* stored in the variable `total`, and that value we then stored in another variable, `invoice`.

We refer to variables declared inside a function definition as being *local* variables, that is, *local to that function*. Variables declared outside any function are known as *global* variables. To add a little more confusion, local and global variables can have the same name, but still be different variables!

The range of situations where a variable is defined is known as the *scope* of the variable—we can refer to a variable as having *local scope* or *global scope*.

TRY IT YOURSELF ▼

Demonstrating the Scope of Variables

To illustrate the issue of a variable's scope, take a look at the following piece of code:

```
var a = 10;
var b = 10;
function showVars() {
    var a = 20; // declare a new local variable 'a'
    b = 20;     // change the value of global variable 'b'
    return "Local variable 'a' = " + a + "\nGlobal variable 'b' = " + b;
}
var message = showVars();
alert(message + "\nGlobal variable 'a' = " + a);
```

Within the `showVars()` function we manipulate two variables, `a` and `b`. The variable `a` we define inside the function; this is a local variable that only exists inside the function, quite separate from the global variable (also called `a`) that we declare at the very beginning of the script.

The variable `b` is not declared inside the function, but outside; it is a *global* variable.

Listing 3.3 shows the preceding code within an HTML page.

LISTING 3.3 Global and Local Scope

```
<!DOCTYPE html>
<html>
<head>
    <title>Variable Scope</title>
</head>
<body>
    <script>
        var a = 10;
        var b = 10;
        function showVars() {
            var a = 20; // declare a new local variable 'a'
            b = 20;     // change the value of global variable 'b'
            return "Local variable 'a' = " + a + "\nGlobal variable 'b' =
➥" + b;
        }
        var message = showVars();
        alert(message + "\nGlobal variable 'a' = " + a);
    </script>
</body>
</html>
```

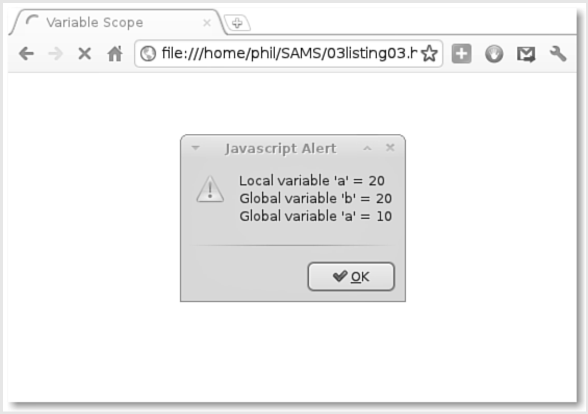
When the page is loaded, `showVars()` returns a message string containing information about the updated values of the two variables `a` and `b`, as they exist inside the function—`a` with local scope, and `b` with global scope.

▼ TRY IT YOURSELF

Demonstrating the Scope of Variables
continued

A message about the current value of the other, *global* variable `a` is then appended to the message, and the message displayed to the user.

Copy the code into a file scope.html and load it into your browser. Compare your results with Figure 3.3.



```
Local variable 'a' = 20
Global variable 'b' = 20
Global variable 'a' = 10
```

FIGURE 3.3
Local and global scope

Summary

In this hour you learned about what functions are and how to create them in JavaScript. You learned how to call functions from within your code and pass information to those functions in the form of arguments. You also found out how to return information from a function to its calling statement.

Finally, you learned about the local or global scope of a variable and how the scope of variables affects how functions work with them.

Q&A

Q. Can one function contain a call to another function?

A. Most definitely; in fact, such calls can be nested as deeply as you need them to be.

Q. What characters can I use in function names?

A. Function names must start with a letter or an underscore and can contain letters, digits, and underscores in any combination. They cannot contain spaces, punctuation, or other special characters.

Workshop

Try to answer all the questions before reading the subsequent "Answers" section.

Quiz

1. Functions are called using
   - **a.** The function keyword
   - **b.** The `call` command
   - **c.** The function name, with parentheses

2. What happens when a function executes a return statement?
   - **a.** An error message is generated.
   - **b.** A value is returned and function execution continues.
   - **c.** A value is returned and function execution stops.

3. A variable declared inside a function definition is called
   - **a.** A local variable
   - **b.** A global variable
   - **c.** An argument

Answers

1. c. A function is called using the function name.
2. c. After executing a return statement, a function returns a value and then ceases function execution.
3. a. A variable defined within a function has local scope.

Exercises

Write a function to take a temperature value in Celsius as an argument and return the equivalent temperature in Fahrenheit, basing it on the code from Hour 2.

Test your function in an HTML page.

INDEX

Symbols

&= (assignment) operator, 404

*= (assignment) operator, 404

^= (assignment) operator, 405

/= (assignment) operator, 404

-= (assignment) operator, 27, 404

%= (assignment) operator, 404

+= (assignment) operator, 27, 403-404

|= (assignment) operator, 405

<<= (assignment) operator, 405

>>= (assignment) operator, 405

>>>= (assignment) operator, 405

\ (backslash), 70

& (bitwise AND) operator, 404

| (bitwise OR) operator, 404

^ (bitwise XOR) operator, 404

~ (bitwise NOT) operator, 404

, (comma) operator, 406

/*...*/ comment syntax, 24

// comment syntax, 24

+ (concatenation) operator, 27-28, 403

?: (conditional) operator, 406

- - (decrement) operator, 26

/ (division) operator, 26, 403

= (equal) operator, 27, 404

== (equality) operator, 405

$() function (prototype.js), 309

$() operator (jQuery), 319

> (greater than) operator, 87, 405

>= (greater than or equal to) operator, 87, 405

++ (increment) operator, 26

<< (left shift) operator, 404

< (less than) operator, 87, 405

<= (less than or equal to) operator, 87, 405

&& (logical AND) operator, 89, 403

|| (logical OR) operator, 89, 403

! (logical NOT) operator, 74, 403

% (modulus) operator, 26, 403

* (multiplication) operator, 26, 403

<!——> notation, 23

!= (not equal) operator, 86, 405

>> (right shift) operator, 404

=== (strict equality) operator, 86, 405

!== (strict not equal) operator, 405

- (unary negation) operator, 26

>>> (zero-fill right shift) operator, 404

A

<a> element, 356

abort() method, 258

abs method, 407

abstraction, 220

accessing

    browser history, 52-53

    classes with className, 207-209

    JSON data, 123-124

accordian widget, 343-344

accordion() method, 343-344

acos method, 407

ActionScript, JavaScript support in, 384

addEventListen() method, 146

Adobe Flash, 358-361

controlling with JavaScript, 359-361

methods, 359

Adobe tools, JavaScript support in, 383

advanced event handler registration

cross-browser implementation, 147-148

Microsoft method, 147

W3C method, 146-147

advantages of JavaScript, 8-9

Ajax

Ajax library

callback function, 276

goals, 274

HTTP GET and POST requests, 275-276

including in HTML page, 278-280

making Ajax calls, 276-277

myAjaxLib.js source code, 277-278

overview, 273

returning keyword META information from remote sites, 280-283

XMLHttpRequest instances, 274-275

application structure, 251-253

flow diagram, 255

server requests, 253-254

server responses, 254

XMLHttpRequest object, 253

callback functions, 264-265

canceling Ajax calls, 270

clock application, 267-269

debugging

Back button, 292-293

bookmarks, 293

browser caching of GET requests, 297

degradation, 294

escaping content, 297

feedback for users, 293-294

with Firebug, 287-290, 299

inappropriate use of Ajax, 295-296

with Internet Explorer, 290-292

links, 293

Permission Denied errors, 297

pop-up information, 295

search engine spiders, 294, 299

security, 296

testing across multiple platforms, 296

implementing with jQuery, 328

ajax() method, 330

get() method, 329

load() method, 329

post() method, 330

sample Ajax form with jQuery, 330-332

server requests

dealing with browser cache, 261-262

overview, 253-254

sending, 260-261

server status, monitoring

readyState property, 263

server response status codes, 264

XMLHttpRequest object

creating instances of, 256-258

methods, 258-260

overview, 253-256

properties, 258-259, 265-267

ajax() method, 330

ajax.js, 267-269

Ajax.PeriodicalUpdater class, 310-311

Ajax.Request class, 309

Ajax.Updater class, 309-310

alert dialogs, displaying, 49

alert() method, 14, 49

allowDrop() function, 375

altKey property (events), 143

ampersand (&), 404

anchor elements, 356

AND operators

& (bitwise AND), 404

&& (logical AND), 403

logical AND (&&), 89

animate() method, 324, 371

animated shooting game example, 242-245

animation

animated shooting game example, 242-245

animating elements with jQuery, 322

animate() method, 324

sample program, 325-327

sliding page elements, 323-324
transitions, 323
<canvas> element, 370-372
CSS3 transitions and transformations, 239-240, 246
DOM elements, 240-241
page elements, 236-237
frame rate, 236, 242
setInterval() method, 237, 246
setTimeout() method, 237, 246
performance optimization, 242
timers, 242
transparency, 238-239
anonymous functions, 107-108
appendChild() method, 187-188
appending child nodes, 187-188
arc() function, 371
arguments
explained, 40
multiple arguments, 40-42
arithmetic operators, 26-27, 403
array.html, 78-79
arrays
array manipulation sample script, 78-79
associative arrays, 80, 127
concatenating, 76
creating, 74-75
defining with JSON (JavaScript Object Notation), 128
definition of, 74
initializing, 75
length property, 75
methods
concat(), 75-76
indexOf(), 75-76
join(), 75-76
lastIndexOf(), 75-77
slice(), 75-77
sort(), 76-77
splice(), 76-77
toString(), 75-76
slicing, 77
sorting, 77
splicing, 77
<article> element, 366
<aside> element, 366
asin method, 407
assigning values to variables, 25
assignment operators, 404-405
associative arrays, 80, 127
assumptions, avoiding, 221
asterisk (*), 26, 403
atan method, 407
attachEvent function, 147
attr() method, 321
attributes (element)
editing, 191-192
reading, 179-180
attributes property (DOM nodes), 180
audio
formats, 353-354
playing with <audio> element, 369
streaming, 363
<audio> element, 369
autoplay attribute (<video> element), 367
.avi files, 354
avoiding
assumptions, 221
overuse of JavaScript, 217-218

B

Back button, debugging, 292-293
back() method, 53
backslash (\), 70
banner cycling script, 97-100
behavior layer, 223
best practices
assumptions, avoiding, 221
code reuse, 220-221
comments, 218-219
error handling, 227-228
feature detection, 226-227
graceful degradation, 221-222
naming conventions, 219-220
overuse of JavaScript, avoiding, 217-218
progressive enhancement, 222-223
unobtrusive JavaScript
converting code into, 228-231
explained, 223-226
bitwise operators, 404
bookListObject, 129-131
bookmarks, debugging, 293
Boolean values, 73-74

break command, 92-93

browser extensions

Firefox extensions, 395

Google Chrome extensions

extension to return airport information, 387-388

icon files, 388

manifest.json file, 388-389

popup.html file, 389-392

writing, 384-386

JavaScript support in, 383

packing, 393

browser plug-ins, 355-358

browser sniffing, 117

browsers

browser history, accessing, 52-53

browser extensions

Firefox extensions, 395

Google Chrome extensions, 384-392

JavaScript support in, 383

packing, 393

browser plug-ins, 355-358

browser sniffing, 117

browser wars, 9-10

cache, 261-262

feature detection, 116-118, 226-227

Firefox

DOM Inspector, 180-181

extensions, 395

Firebug, 100, 287-290, 299, 401

JavaScript support in, 384

Internet Explorer, 9

debugging Ajax with, 290-292

F12 Developer Tools, 100

JSON (JavaScript Object Notation) support, 123, 133

buttonReport() function, 41-42

C

cache (browser), 261-262

callback functions, 264-265, 276

calling functions, 38

CamelCase naming convention, 25, 220

canceling Ajax calls, 270

canPlayType() method, 368

<canvas> element, 370-372

capturing mouse events

onClick event handler, 30-31

onMouseOut event handler, 31-33

onMouseOver event handler, 31-33

caret (^), 404

Cascading Style Sheets. *See* CSS styles

case sensitivity, 25, 31

catch statement, 227-228

CDNs (Content Delivery Networks), 318

ceil() method, 59-60, 407

Celsius, converting to Fahrenheit, 28-29

changing

classes with className, 207-209

mouse cursor, 215

character strings, assigning to variable values, 25

checkform() function, 140

child nodes (DOM)

appending, 187-188

inserting, 188

removing, 191

replacing, 188-190

childNodes property (DOM nodes), 174-175

Chome extensions

extension to return airport information, 387-388

icon files, 388

JavaScript support in, 383

manifest.json file, 388-389

packing, 393

popup.html file

basic popup.html, 389-391

complete popup.html, 391-392

writing, 384-386

classes

accessing using className, 207-209

Ajax.PeriodicalUpdater, 310-311

Ajax.Request, 309

Ajax.Updater, 309-310

className property, 207-209

clearRect() method, 371

client-side programming, 8

clientX property (events), 143

clientY property (events), 143

clock application (Ajax), 267-269

cloneNode() method, 186-187

cltrlKey property (events), 143

code reuse, 104, 220-221

collections, NodeList, 174

comma (,) operator, 406

command chaining (jQuery), 324

comments

- HTML comments, 23
- JavaScript comments, 24
- performance considerations, 24
- writing, 218-219

comparison operators, 86-87, 405

compiled languages, 10

concat() method, 71-72, 75-76

concatenating

- arrays, 76
- strings, 27-28, 72

concatenation operator, 27-28

conditional (?:) operator, 406

conditional statements

- if(), 85-86
  - testing for equality, 87-88
  - testing multiple conditions, 88
- switch, 88-89

confirm() method, 50

confirmation dialogs, displaying, 50

constants, mathematical, 61

constructor functions, 108-110

Content Delivery Networks (CDNs), 318

continue statement, 101

control structures. *See* loops

controls attribute (<video> element), 367

converting

- Celsius to Fahrenheit, 28-29
- code into unobtrusive code, 228-231
- strings
  - to numbers, 69
  - to uppercase/lowercase, 73

cookie property (document object), 156

cookieName value, 157

cookies

- components, 157-158
  - cookieName value, 157
  - cookieValue value, 157
  - domain attribute, 157-158
  - expires attribute, 158
  - path attribute, 158
  - secure attribute, 158
- definition of, 155-156
- deleting, 162
- document.cookie property, 156
- escaping and unescaping data, 156-157
- limitations of, 156
- reading, 161-162
- security, 166
- setting multiple values in, 165-166
- testing
  - cookies.js, 162-163
  - cookietest.html, 163-165
  - cookietest2.html, 164-165
- writing, 158-161

cookies.js, 162-163

cookietest.html, 163-165

cookietest2.html, 164-165

cookieValue value, 157

cos method, 407

createCookie() function, 159-162

createElement() method, 185-186

createElementTextNode() method, 186

Crockford, Douglas, 401

cross-browser event handlers, 142-144, 147-148, 152

CSS (Cascading Style Sheets) styles, 201

- advantages of, 201-202
- changing classes using className, 207-209
- CSS3 animations
  - transformations, 239-240, 246
  - transitions, 239-240, 246
- DOM style property, 204-207
- style declarations
  - placing, 204
  - syntax, 202-203
- style properties, setting, 205-206
- stylesheets
  - DOM styleSheets object, 209
  - enabling/disabling, 209-211
  - selecting, 211-214
  - switching, 209-211

cursor, changing, 215

customized widgets, 349

cycle() function, 98-99

cycling images on page, 97-100

D

data serialization, 124-126

data types

- arrays
  - associative arrays, 80
  - associative arrays, simulating, 127
  - creating, 74-75
  - definition of, 74
  - initializing, 75
  - length property, 75
  - methods, 75-77
- definition of, 67
- JSON (JavaScript Object Notation) data types, 126
- numbers
  - converting strings to, 69
  - floating-point numbers, 68
  - hexadecimal numbers, 68
  - infinity, 69-70
  - integers, 67-68
  - NaN (not a number), 69
  - natural numbers, 67
  - whole numbers, 67
- strings
  - Boolean values, 73-74
  - converting to numbers, 69
  - definition of, 70
  - escape sequences, 70-71
  - maximum length of, 80
  - methods, 71-73

Date object

- creating with a given date and time, 58
- creating with current date and time, 57
- methods, 408-410
- reading date and time, 62-64
- setting/editing dates and times, 58-59
- time zones, 65

date picker widget, 344-346

datepicker() method, 345-346

datetime.js, 230

debugging, 94-96. *See also* error handling, 227-228

- Ajax, 287
  - Back button, 292-293
  - bookmarks, 293
  - browser caching of GET requests, 297
  - degradation, 294
  - escaping content, 297
  - feedback for users, 293-294
  - with Firebug, 287-290, 299
  - inappropriate use of Ajax, 295-296
  - with Internet Explorer, 290-292
  - links, 293
  - Permission Denied errors, 297
  - pop-up information, 295
  - search engine spiders, 294, 299
  - security, 296
  - testing across multiple platforms, 296
- debugging tools
  - Firebug, 401
  - JSLint, 401

declaring

- CSS styles
  - placement of style declarations, 204
  - syntax, 202-203
- direct instances, 104-108

decrement (- -) operator, 26

default actions (event handlers)

- explained, 138
- preventing, 138-141

degradation, graceful, 221-222, 294

delete operator, 406

deleteCookie() function, 162-163

deleting cookies, 162

deprecated status, 11

deserialization (JSON)

- with eval() function, 123
- with native browser support, 123, 133
- with parse() function, 123

detachEvent function, 147

detecting browser features, 226-227

detecting features, 116-118

development

- of JavaScript, 9
- of jQuery, 333

dialogs, displaying

- alert dialogs, 49
- confirmation dialogs, 50
- prompts, 50-51

direct instances, declaring, 104-105
anonymous functions, 107-108
this keyword, 105-107
disabling stylesheets, 209-211
displayData() function, 391
displaying
alert dialogs, 49
confirmation dialogs, 50
prompts, 50-51
division (/) operator, 26
do … while loop, 91
doAjax() function, 276-277
document object, 12
cookie property, 156
getElementById() method, 51-52
reading properties of, 16
write() method, 14-15
Document Object Model. *See* DOM
$(document).ready handler, 318-319
documentation (JSON), 133
documents (HTML), including JavaScript in, 21-23
Dojo library, 305
DOM (Document Object Model), 10
animating DOM elements, 240-241
browser support for, 12
development of, 10
DOM Core, 199
DOM Inspector, 180-181
element attributes, editing, 191-192
explained, 11
JavaScript files, loading dynamically, 192-193
levels, 11
nodes
attributes property, 180
child nodes, 187-191
childNodes property, 174-175
creating, 185-187
determining whether nodes have child nodes, 182
explained, 171-173
firstChild property, 175
lastChild property, 176
names, 177
nextSibling property, 176
node lists, 174
nodeName property, 177
nodeValue property, 176-177
parentNode property, 176
previousSibling property, 176
types of, 173-174
values, 176-177
objects. *See* objects
DOM Core, 199
DOM Inspector, 180-181
domain attribute (cookies), 157-158
dot notation, 13
downloading
DOM Inspector, 180
jQuery, 317-319
drag and drop
in HTML5, 373-375
with jQuery UI, 337-340
drag() function, 375
draggable() method, 337-338
Dragonfly, 100
drawing with canvas element, 370-372
drop() function, 375
droppable() method, 339-340
dynamic menu creation, 193-197
dynamic file loading, 192-193

E

E constant, 61
E property (Math object), 407
ECMA (European Computer Manufacturers Association), 9
ECMAScript, 9
editing
dates/times, 58-59
element attributes, 191-192
editors
Geany, 400
jEdit, 400
Notepad++, 399
SciTE, 400
element nodes, 173
elements
<a>, 356
animating, 236-237, 322
animate() method, 324
sample program, 325-327
setInterval() method, 237, 246
setTimeout() method, 237, 246
sliding page elements, 323-324
transitions, 323
<article>, 366

- <aside>, 366
- attributes, editing, 191-192
- <audio>, 369
- <canvas>, 370-372
- <embed>, 357
- <figcaption>, 366
- <figure>, 366
- <footer>, 366
- getting and setting content of, 320-321
- <head>, 38-39
- <header>, 366
- <nav>, 366
- <noscript>, 232
- <object>, 357
- reading attributes of, 179-180
- <script>, 10-11, 21-22
- <section>, 366
- selecting, 319-320
  - by id, 51-52
  - with getElementsByTagName(), 177-179
- showing/hiding, 321-322
- <summary>, 366
- <video>, 366-368

<embed> element, 357

enabling stylesheets, 209-211

encapsulation, 104, 115-116

ended attribute (<video> element), 367

enhancement, progressive, 222-223

equal sign (=), 27, 404

equality, testing for, 87-88

equality (==) operator, 405

error handling, 227-228. ***See also*** **debugging, 94-96**

escape() function, 156-157

escaping data, 41, 70-71, 156-157, 297

European Computer Manufacturers Association (ECMA), 9

eval() function, 123

event handlers, 30, 135
- adding, 136-137, 148-151
- advanced event handler registration
  - cross-browser implementation, 147-148
  - Microsoft method, 147
  - W3C method, 146-147
- common event handlers, 136
- cross-browser event handlers, 142-144, 147-148, 152
- default actions, 138-141
- inline event handlers, 136
- jQuery, 328
- onClick, 30-31, 144-145, 148-151
- onMouseOut, 31-33
- onMouseOver, 31-33
- removing, 137, 148-151

event object
- explained, 141
- Microsoft approach, 142
- W3C approach, 142

events
- event handlers, 30, 135
  - adding, 136-137, 148-151
  - advanced event handler registration, 146-148
  - common event handlers, 136
  - cross-browser event handlers, 142-144, 147-148, 152
  - default actions, 138-141
  - inline event handlers, 136
  - jQuery, 328
  - onClick, 30-31, 144-145, 148-151
  - onMouseOut, 31-33
  - onMouseOver, 31-33
  - removing, 137, 148-151
- event object
  - explained, 141
  - Microsoft approach, 142
  - W3C approach, 142
- mouse events, capturing, 30
  - onClick, 30-31
  - onMouseOut, 31-33
  - onMouseOver, 31-33
- properties, 143

events.js, 149-150

exclamation point (!), 403

exp method, 407

expires attribute (cookies), 158

exponential notation, 68

extending objects, 111-115

Extensible Markup Language (XML)
- compared to JSON, 122
- parsing, 285

extensions (browser). ***See*** **browser extensions**

F

$F() function, 307

F12 Developer Tools, 100

fadeIn() method, 323

fadeOut() method, 323

fadeTo() method, 323

fading

page elements, 323

between transparency and opacity, 238-239

Fahrenheit, converting Celsius to, 28-29

"falsy" values, 74

feature detection, 57, 116-118, 226-227

feedback for users, debugging, 293-294

figcaption element, 366

<figure> element, 366

File API (HTML5), 377-379

File interface, 377

FileList interface, 377

files. ***See also specific files***

audio formats, 353-354

loading dynamically, 192-193

local files, interacting with, 377-379

naming, 219-220

video formats, 354-355

finding maximum and minimum, 60

Firebug, 100, 287-290, 299, 401

Firefox

DOM Inspector, 180-181

extensions, 395

Firebug, 100, 287-290, 299, 401

JavaScript support in, 384

firstChild property (DOM nodes), 175

Flash, 358-361

controlling with JavaScript, 359-361

methods, 359

flashLoaded() function, 359-361

Flickr, support for JSON, 122

floating-point numbers, 68

floor method, 407

floor() method, 59-60

.flv files, 355

footer element, 366

for loop, 91-92

for...in loop, 93

Form object, 308

forms

Form object, 308

sample Ajax form with jQuery, 330-332

forward slash (/), 26

forward() method, 53

fps (frames per second), 236

frame rate, 236, 242

frames per second (fps), 236

function operator, 38, 406

functions. ***See also*** **methods**

$(), 306-307

adding to page head, 38-39

Ajax callback functions, 264-265

allowDrop(), 375

animate(), 371

anonymous functions, 107-108

arc(), 371

arguments, 40-42

attachEvent(), 147

buttonReport(), 41-42

callback functions, 276

calling, 38

checkform(), 140

compared to methods, 38

constructor functions, 108-110

createCookie(), 159-162

creating, 37-38

cycle(), 98-99

deleteCookie(), 162-163

detachEvent, 147

displayData(), 391

doAjax(), 276-277

drag(), 375

drop(), 375

escape(), 156-157

eval(), 123

explained, 22, 37

$F(), 307

flashLoaded(), 359-361

getCookie(), 162-163

getElementArea(), 221

isFinite(), 70

jsonParse(), 125

lineTo(), 370

moveItRight(), 240-241

moveTo(), 370

naming, 38, 47

nesting, 47

outputting user messages with, 41-42

parse(), 123
parseFloat(), 69
parseInt(), 69
responseAjax(), 264-265
returning values from, 43
setInterval(), 97
showVars(), 45
syntax, 37-38
telltime(), 63, 228-230
toggleClass(), 208
Try.these, 308
unescape(), 156-157
variable scope, 44-46

G

games, animated shooting game, 242-245
Geany, 400
generating random numbers, 60-61
GET requests, 270, 275-276, 297
get() method, 329
getAllResponseHeaders() method, 258
getAttribute() method, 180, 191
getCookie() function, 162-163
getDate() method, 408
getDay() method, 408
getElementArea() function, 221
getElementById() method, 51-52, 117
getElements() method, 308
getElementsByTagName() method, 177-179
getFullYear() method, 65, 408
getHours() method, 408
getMilliseconds() method, 408
getMinutes() method, 408
getMonth() method, 408
getResponseHeader() method, 258
getSeconds() method, 408
getTime() method, 408
getTimezoneOffset() method, 65, 408
getUTCDate() method, 408
getUTCDay() method, 408
getUTCFullYear() method, 408
getUTCHours() method, 408
getUTCMilliseconds() method, 408
getUTCMinutes() method, 408
getUTCMonth() method, 408
getUTCSeconds() method, 408
getYear() method, 65
global objects, 14
global variables, 44-46
Google Ajax API CDN, 318
Google Apps Script, JavaScript support in, 384
Google Chrome extensions. *See* Chrome extensions, 383
GotoFrame() method, 359
graceful degradation, 221-222
graphics. *See* images
greater than (>) operator, 87, 405
greater than or equal to (>=) operator, 87, 405

H

handling
errors, 227-228
events. *See* event handlers
hasChildNodes() method, 182
<head> element, 38-39
<header> element, 366
Hello World example, 15-16
hexadecimal numbers, 68
hide() method, 322
hiding page elements, 321-322
history of jQuery, 333
history (browser), accessing, 52-53
history object, 12, 52-53
history of JavaScript, 9
HTML (HyperText Markup Language)
comment syntax, 23
elements. *See* elements
explained, 7
HTML5
<audio> element, 369
<canvas> element, 370-372
drag and drop, 373-375
File API, 377-379
learning, 380
local storage, 376-377
overview, 365-366
<video> element, 366-368
web site examples, 380
metatags, 280
white space in, 175-176
html() method, 320, 390

HTML5
- <audio> element, 369
- <canvas> element, 370-372
- drag and drop, 373-375
- File API, 377-379
- learning, 380
- local storage, 376-377
- overview, 365-366
- <video> element, 366-368
- website examples, 380

HTTP
- GET requests, 275-276, 297
- POST requests, 275-276

HyperText Markup Language. ***See* HTML**

hyphen (-), 26

I

icon files for Google Chrome extensions, 388

IDs, selecting elements by, 51-52

if() statement, 85-86
- testing for equality, 87-88
- testing multiple conditions, 88

image rollovers, creating, 32-33

images
- animation
  - animated shooting game example, 242-245
  - CSS3 transitions and transformations, 239-240, 246
  - DOM elements, 240-241
  - frame rate, 236, 242
  - page elements, 236-237, 246
  - performance optimization, 242
  - timers, 242
  - transparency, 238-239
- cycling on web page, 97-100
- drawing with canvas element, 370-372
- preloading, 235-236
- rollovers, creating, 32-33

in operator, 406

inappropriate use of Ajax, 295-296

including
- jQuery from CDNs (Content Delivery Networks), 318
- jQuery UI in web pages, 336-337

increment (++) operator, 26

indexOf() method, 71-72, 75-76, 406

infinity, 69-70

inheritance, 104, 112-113

initializing arrays, 75

inline event handlers, 136

innerHTML property, 52, 199

insertBefore() method, 188

inserting child nodes, 188

instanceof operator, 406

instantiating objects, 108-110, 256-258

integers, 67-68

interacting with user
- alert dialogs, 49
- confirmation dialogs, 50
- prompts, 50-51

Internet Explorer, 9
- debugging Ajax with, 290-292
- F12 Developer Tools, 100

interpreted languages, 10

isFinite() function, 70

IsPlaying() method, 359

J

Java, 8

JavaScript Object Notation. ***See* JSON**

jEdit, 400

join() method, 75-76

jQuery, 306
- command chaining, 324
- compatibility with other libraries, 333
- development of, 333
- $(document).ready handler, 318-319
- downloading, 317-318
- event handling, 328
- implementing Ajax with, 328
  - ajax() method, 330
  - get() method, 329
  - load() method, 329
  - post() method, 330
  - sample Ajax form with jQuery, 330-332
- including from CDNs (Content Delivery Networks), 318
- jQuery CDN, 318

jQuery UI
- accordian widget, 343-344
- advantages of, 335-336
- customized widgets, 349
- date picker widget, 344-346
- drag and drop, 337-340
- including in web pages, 336-337
- resizing handles, 341
- sortable lists, 341-342
- tabbed interfaces, 346-348
- ThemeRoller, 336-337

methods
- ajax(), 330
- attr(), 321
- get(), 329
- hide(), 322
- html(), 320
- load(), 329
- post(), 330
- show(), 321-322
- text(), 320-321
- toggle(), 322

page elements
- animating, 322-327
- getting and setting content of, 320-321
- selecting, 319-320
- showing/hiding, 321-322

jQuery CDN, 318

.js file extension, 22

jScript, 9

JSLint, 401

JSON (JavaScript Object Notation), 121
- accessing JSON data, 123-124
  - with eval() function, 123
  - with native browser support, 123
  - with parse() function, 123
- associative arrays, simulating, 127
- compared to XML, 122
- data serialization, 124-126
- data types, 126
- documentation, 133
- explained, 121
- Flickr support for, 122
- objects, creating, 127
  - arrays, 128-129
  - methods, 128
  - multilevel objects, 129-131
  - objects within objects, 129
  - properties, 128
- parsing, 125-126
- security, 131
- syntax, 121-122
- web page, 121

jsonParse() function, 125

junctions, 90

K

keyCode property (events), 143

keyword META information, returning from remote sites, 280-283

keywords. *See* functions; methods; statements

L

language attribute (<script> element), 10

lastChild property (DOM nodes), 176

lastIndexOf() method, 71-72, 75-77, 406

layers
- behavior layer, 223
- presentation layer, 223
- semantic layer, 223

leaving loops
- with break, 92-93
- with continue, 101

left shift (>>) operator, 404

length property
- arrays, 75
- history object, 52

less than (<) operator, 87, 405

less than or equal to (<=) operator, 87, 405

libraries, 303
- advantages of, 303-304
- Ajax library
  - callback function, 276
  - goals, 274
  - HTTP GET and POST requests, 275-276
  - including in HTML page, 278-280
  - making Ajax calls, 276-277
  - myAjaxLib.js source code, 277-278
  - overview, 273

returning keyword META information from remote sites, 280-283
XMLHttpRequest instances, 274-275
Dojo, 305
including in web pages, 314
jQuery. *See* jQuery
MooTools, 305
Prototype Framework, 305
prototype.js
$() function, 306-307
Ajax.PeriodicalUpdater class, 310-311
Ajax.Request class, 309
Ajax.Updater class, 309-310
$F() function, 307
Form object, 308
stock price reader, 311-312
Try.these() function, 308
Yahoo! UI Library, 305

LibreOffice.org, JavaScript support in, 383
lineTo() function, 370
links, debugging, 293
lists, sorting, 341-342
LN2, 61, 407
LN10, 61, 407
load() method, 329
loading
JavaScript files dynamically, 192-193
multimedia
with <a> element, 356
with <embed> element, 357
with <object> element, 357
with plug-ins, 357-358

local files, interacting with, 377-379
local storage with HTML5, 376-377
local variables, 44-46
localStorage object, 376
location object
navigating with, 54
properties, 53
reloading pages, 54

log method, 407
LOG10E, 61, 407
LOG2E, 61, 407
logical AND (&&) operator, 89
logical operators, 89-90, 403
logical OR (||) operator, 89
loop attribute (<video> element), 367
loops, 90. *See also* statements
choosing, 101
do … while, 91
for, 91-92
for...in, 93
leaving with break, 92-93
leaving with continue, 101
while, 90-91

loosely typed languages, 67
lowercase, converting strings to, 73

M

manifest.json file, 388-389
Math object, 59
constants, 407
finding maximum and minimum, 60
generating random numbers, 60-61
mathematical constants, 61
methods, 59-61, 407
rounding, 60
with keyword, 61-62

mathematical constants, 61
max() method, 60, 407
maximum numbers, finding, 60
menu.js, 196-197
menus, creating dynamically, 193-197
messages, outputting with functions, 41-42
META information, returning from remote sites, 280-283
metatags, 280
metatags.html, 282
methods, 12. *See also* functions
abort(), 258
accordion(), 343-344
addEventListener, 146
adding with prototype, 111-112
ajax(), 330
alert(), 49
animate(), 324
appendChild(), 187-188
attr(), 321
back(), 53

canPlayType(), 368
ceil(), 59-60
clearRect(), 371
cloneNode(), 186-187
compared to functions, 38
concat(), 71-72, 75-76
confirm(), 50
createElement(), 185-186
createElementTextNode(), 186
datepicker(), 345-346
defining with JSON (JavaScript Object Notation), 128
draggable(), 337-338
droppable(), 339-340
fadeIn(), 323
fadeOut(), 323
fadeTo(), 323
flashLoaded(), 359
floor(), 59-60
forward(), 53
get(), 329
getAllResponseHeaders(), 258
getAttribute(), 180, 191
getElementById(), 51-52, 117
getElements(), 308
getElementsByTagName(), 177-179
getFullYear(), 65
getResponseHeader(), 258
getTimezoneOffset(), 65
getYear(), 65
GotoFrame(), 359
hasChildNodes(), 182
hide(), 322
html(), 320, 390
indexOf(), 71-72, 75-76
insertBefore(), 188
IsPlaying(), 359
join(), 75-76
lastIndexOf(), 71-72, 75-77
load(), 329
max(), 59-60
min(), 59-60
naming, 219-220
open(), 259
pause(), 368
PercentLoaded(), 359
Play(), 359, 368
post(), 330
prompt(), 50-51
random(), 59-60
reload, 54
reload(), 54
removeEventListener, 146
replace(), 71-72
resizable(), 341
Rewind(), 359
round(), 59-60
send(), 258-260
serialize(), 308
setAttribute(), 191
setFullYear(), 65
setInterval(), 237, 246, 371
setRequestHeader(), 258
setTimeout(), 237, 246
setYear(), 65
show(), 321-322
slice(), 75-77
slideDown(), 323
slideUp(), 323
sort(), 76-77
sortable(), 341-342
splice(), 76-77
split(), 71-73
StopPlay(), 359
stringify(), 124
substr(), 72-73
tabs(), 347
text(), 320-321
toDateString(), 59
toggle(), 322
toLowerCase(), 72-73
toString(), 75-76
TotalFrames(), 359
toTimeString(), 59
toUpperCase(), 72-73
Zoom(), 359

Microsoft CDN, 318

min() method, 59-60, 407

minimum numbers, finding, 60

modulus (%) operator, 26, 403

monitoring server status, 263
readyState property, 263
server response status codes, 264

MooTools library, 305

mouse cursor, changing, 215

mouse events, capturing
onClick event handler, 30-31
onMouseOut event handler, 31-33
onMouseOver event handler, 31-33

.mov files, 354

moveItRight() function, 240-241

moveTo() function, 370

movies. ***See* video**

moving ball animation, 370-372

Mozilla Firefox

- DOM Inspector, 180-181
- extensions, 395
- Firebug, 100, 287-290, 299, 401
- JavaScript support in, 384

.mp3 files, 354

.mp4 files, 355

.mpeg files, 354

.mpg files, 354

multilevel objects (JSON), 129-131

multiline comments, 24

multimedia

- audio
  - formats, 353-354
  - playing with <audio> element, 369
  - streaming, 363
- browser plug-ins, 355-358
- disadvantages of, 363
- Flash, 358-361
  - controlling with JavaScript, 359-361
  - methods, 359
- loading and playing
  - with <a> element, 356
  - with <embed> element, 357
  - with <object> element, 357
- video
  - formats, 354-355
  - playing with <video> element, 366-368

multiple arguments, 40-42

multiple conditions, testing for, 88

multiple platforms, testing across, 296

multiple values, setting in cookies, 165-166

multiplication (*) operator, 26

myAjaxLib

- callback function, 276
- goals, 274
- HTTP GET and POST requests, 275-276
- including in HTML page, 278-280
- making Ajax calls, 276-277
- myAjaxLib.js source code, 277-278
- overview, 273
- returning keyword META information from remote sites, 280-283
- XMLHttpRequest instances, 274-275

N

\n (new line) character, 41

naming conventions, 219-220

- files, 219-220
- functions, 38, 47
- methods, 219-220
- nodes, 177
- properties, 219-220
- variables, 25

NaN (not a number), 69

natural numbers, 67

<nav> element, 366

navigating

- DOM (Document Object Model). *See* DOM (Document Object Model), 171
- with location object, 54

navigator object, 54-57

navigator.html file, 55

negation operator (!), 74

negative infinity, 70

nesting functions, 47

Netscape Navigator 4, 9

new line (\n) character, 41

new operator, 406

nextSibling property (DOM nodes), 176

Node js, 395

NodeList, 174

nodeName property (DOM nodes), 177

nodes (DOM)

- child nodes
  - appending, 187-188
  - inserting, 188
  - removing, 191
  - replacing, 188-190
- creating
  - cloneNode() method, 186-187
  - createElement() method, 185-186
  - createElementTextNode() method, 186
- determining whether nodes have child nodes, 182
- explained, 171-173

names, 177
node lists, 174
properties
attributes, 180
childNodes, 174-175
firstChild, 175
lastChild, 176
nextSibling, 176
nodeName, 177
nodeValue, 176-177
parentNode, 176
previousSibling, 176
types of, 173-174
values, 176-177
nodeValue property (DOM nodes), 176-177
<noscript> element, 232
not a number (NaN), 69
not equal (!=) operator, 405
NOT operators
! (logical NOT), 403
~ (bitwise NOT), 404
Notepad++, 399
null values, 51, 74
numbers, 67
converting strings to, 69
finding maximum and minimum, 60
floating-point numbers, 68
hexadecimal numbers, 68
infinity, 69-70
integers, 67-68
NaN (not a number), 69
natural numbers, 67
random numbers, generating, 60-61
rounding, 60
whole numbers, 67

O

object literals, 74
Object object, 104-105
Object Oriented languages, 104
<object> element, 357
object-oriented programming. *See* OOP
objects. *See also* OOP (object-oriented programming)
bookListObject, 129-131
creating direct instances of, 104-105
anonymous functions, 107-108
this keyword, 105-107
creating with constructor functions, 108-110
creating with JSON, 127
arrays, 128-129
methods, 128
multilevel objects, 129-131
objects within objects, 129
properties, 128
Date, 57
creating with a given date and time, 58
creating with current date and time, 57
methods, 408-410
reading date and time, 62-64
setting/editing dates and times, 58-59
time zones, 65
document, 12
cookie property, 156
getElementById() method, 51-52
reading properties of, 16
write() method, 14-15
encapsulation, 115-116
event
explained, 141
Microsoft approach, 142
W3C approach, 142
explained, 12
extending, 111-115
feature detection, 116-118
Form, 308
global objects, 14
history, 12, 52-53
inheritance, 112-113
instantiating, 108-110
JSON (JavaScript Object Notation), 13
accessing JSON data, 123-124
associative arrays, simulating, 127
compared to XML, 122
data serialization, 124-126
data types, 126
explained, 121
Flickr support for, 122

multilevel objects, 129-131
object creation, 127-129
parsing, 125-126
syntax, 121-122
web page, 121
localStorage, 376
location, 53-54
navigating with, 54
properties, 53
reloading pages, 54
Math
constants, 407
finding maximum and minimum, 60
generating random numbers, 60-61
mathematical constants, 61
methods, 59-61, 407
rounding, 60
with keyword, 61-62
navigator, 54-57
Object, 104-105
portability, 119
sessionStorage, 376
styleSheets, 209
window, 12-13
alert() method, 14, 49
confirm() method, 50
prompt() method, 50-51
XMLHttpRequest
creating instances of, 256-258
methods, 258-260
overview, 253-256
properties, 258-259

obtrusive code, converting into unobtrusive code, 228-231
onBlur event handler, 136
onChange event handler, 136
onClick event handler, 30-31, 136
adding/removing, 148-151
listing onClick event properties, 144-145
properties, listing, 144
onDblClick event handler, 136
onFocus event handler, 136
onKeydown event handler, 136
onKeypress event handler, 136
onKeyup event handler, 136
onLoad event handler, 136
onMousedown event handler, 136
onMousemove event handler, 136
onMouseOut event handler, 31-33, 136
onMouseOver event handler, 31-33, 136
onMouseup event handler, 136
onreadystatechange property (XMLHttpRequest object), 258
onReset event handler, 136
onSelect event handler, 136
onSubmit event handler, 136, 139-140
onUnload event handler, 136
OOP (object-oriented programming)
advantages of, 104
encapsulation, 115-116
feature detection, 116-118
inheritance, 112-113
object creation
constructor functions, 108-110
direct instances, 105-108
object extension, 114-115
overview, 103-104
when to use, 119
opacity, fading between transparency and opacity, 238-239
open() method, 259
opening DOM Inspector, 180
OpenOffice.org, JavaScript support in, 383
Opera Dragonfly, 100
operators
arithmetic operators, 26-27, 403
assignment operators, 404-405
bitwise operators, 404
comparison operators, 86-87, 405
concatenation operator, 27-28
logical operators, 89-90, 403
negation (!) operator, 74
precedence, 27
special operators, 406
string operators, 403
typeof, 117-118, 226
optimizing performance. *See* performance considerations, 18, 24, 242
OR operators
| (bitwise OR), 404
|| (logical OR), 89, 403
overuse of JavaScript, avoiding, 217-218

P

packing extensions, 393

page elements. *See* elements

page head, adding functions to, 38-39

parentNode property (DOM nodes), 176

parse() method, 123, 408

parseFloat() function, 69

parseInt() function, 69

parsing

- JSON strings, 125-126
- text, 285
- XML, 285

path attribute (cookies), 158

pause() method, 368

paused attribute (<video> element), 367

PDF files, JavaScript support in, 383

percent sign (%), 26

PercentLoaded() method, 359

performance considerations

- animation, 242
- comments, 24
- JavaScript, 18

Permission Denied errors, 297

PHP-enabled web servers, 282

PI, 61, 407

placing style declarations, 204

Play() method, 359, 368

playing multimedia

- with <audio> element, 369
- with <a> element, 356
- with <embed> element, 357
- with <object> element, 357
- with plug-ins, 357-358
- with <video> element, 366-368

plug-ins (browser), 355-358

plus sign (+), 403

pop-up information, 295

popup.html Google Chrome extension

- basic popup.html file, 389-391
- complete popup.html file, 391-392
- icon files, 388
- manifest.json file, 388-389
- returning airport information, 387-388
- writing, 384-386

POST requests, 270, 275-276

post() method, 330

poster attribute (<video> element), 367

pow method, 407

precedence of operators, 27

preloading images, 235-236

presentation layer, 223

preventing default actions, 138-141

previousSibling property (DOM nodes), 176

procedural programming, 103

programs, 21

- Ajax clock application, 267-269
- banner cycling script, 97-100
- debugging, 94-96
- definition of, 8
- Hello World example, 15-16
- including in web pages, 21-23
- stock price reader, 311-312

progressive enhancement, 222-223

prompt() method, 50-51

prompts, displaying, 50-51

properties. ***See also specific properties***

- defining with JSON (JavaScript Object Notation), 128
- explained, 12
- naming, 219-220
- reading, 16
- stringlike properties, 156
- style properties, 205-206

protocols, stateless, 155

Prototype Framework library, 305

prototype keyword, extending objects with, 111-115

prototype.js library

- $() function, 306-307
- Ajax.PeriodicalUpdater class, 310-311
- Ajax.Request class, 309
- Ajax.Updater class, 309-310
- $F() function, 307
- Form object, 308
- stock price reader, 311-312
- Try.these function, 308

Q-R

QuickTime files, 354

.ram files, 354

random method, 408

random numbers, generating, 60-61

random() method, 59-60

reading

cookies, 161-162

date and time, 62-64

document object properties, 16

element attributes, 179-180

readystate property (XMLHttpRequest object), 258, 263-264

Real Audio files, 354

RealVideo files, 354

reload() method, 54

reloading web pages, 54

remote sites, returning keyword META information from, 280-283

removeEventListen method, 146

removing

child nodes, 191

event handlers, 137, 148-151

rendering context, 370

replace() method, 71-72, 406

replacing

child nodes, 188-190

substrings within strings, 72

requests

Ajax server requests

dealing with browser cache, 261-262

overview, 253-254

sending, 260-261

HTTP GET and POST requests, 270, 275-276, 297

XMLHttpRequest

creating instances of, 256-258

methods, 258-260

overview, 256

properties, 258-259

Resig, John, 333

resizable() method, 341

resizing handles, adding, 341

responding to events. *See* event handlers

responseAjax() function, 264-265

responses (server), 254

responseText property (XMLHttpRequest object), 258, 265-266, 285

responseXML property (XMLHttpRequest object), 258, 265-267

return values, 43

reusing code, 104, 220-221

Rewind() method, 359

right shift (<<) operator, 404

.rm files, 354

round() method, 59-60, 407

rounding numbers, 60

S

SciTE, 400

scope of variables, 44-46

screenX property (events), 143

screenY property (events), 143

<script> element, 10-11, 21-22

scripts. *See* programs

search engine spiders, 294, 299

<section> element, 366

secure attribute (cookies), 158

security

Ajax, 296

cookies, 166

JSON, 131

selecting

elements, 319-320

by id, 51-52

with getElementsByTagName(), 177-179

stylesheets, 211-214

semantic layer, 223

send() method, 258-260

sending Ajax server requests, 260-261

serialization with JSON (JavaScript Object Notation), 124-126

serialize() method, 308

server requests (Ajax)

dealing with browser cache, 261-262

overview, 253-254

sending, 260-261

server responses, 254

server status, monitoring, 263

readyState property, 263

server response status codes, 264

server-side programming, 8

sessionStorage object, 376

setAttribute() method, 191

setDate() method, 408

setFullYear() method, 65, 409

setHours() method, 409

setInterval() method, 97, 237, 246, 371

setMilliseconds() method, 409

setMinutes() method, 409

setMonth() method, 409

setRequestHeader() method, 258

setSeconds() method, 409

setTime() method, 409

setTimeout() method, 237, 246

setting

- dates/times, 58-59
- page element content, 320-321

setUTCDate() method, 409

setUTCFullYear() method, 409

setUTCHours() method, 409

setUTCMilliseconds() method, 409

setUTCMinutes() method, 409

setUTCMonth() method, 409

setUTCSeconds() method, 409

setYear() method, 65

shiftKey property (events), 143

ShockWave Flash. *See* Flash, 358-361

shooting game (animated), 242-245

show() method, 321-322

showVars() function, 45

simulating associative arrays, 127

sin method, 407

slash (/), 403

slice() method, 75-77

slicing arrays, 77

slideDown() method, 323

slideUp() method, 323

sliding page elements, 323-324

software. *See* tools, 399-401

sort() method, 76-77

sortable() method, 341-342

sorting

- arrays, 77
- lists, 341-342

sound

- formats, 353-354
- playing with audio element, 369
- streaming, 363

special operators, 406

Sphere, JavaScript support in, 384

spiders, 294, 299

splice() method, 76-77

splicing arrays, 77

split method, 406

split() method, 71-73

splitting strings, 73

sqrt method, 407

SQRT1_2, 61, 407

SQRT2, 61, 407

stateless protocols, 155

statements. *See also* loops

- break, 92-93
- catch, 227-228
- comments, 24
- continue, 101
- do … while, 91
- explained, 23
- for, 91-92
- for...in, 93
- if(), 85-86
  - testing for equality, 87-88
  - testing multiple conditions, 88
- function, 38
- null, 74
- prototype, 111-115
- switch, 88-89
- this, 105-107
- try, 227-228
- while, 90-91
- with, 61-62

static web pages, 7

status of server, monitoring

- readyState property, 263
- server response status codes, 264

status property (XMLHttpRequest object), 258

statusText property (XMLHttpRequest object), 258

stock price reader program, 311-312

StopPlay() method, 359

storage, local storage with HTML5, 376-377

streaming audio/video, 363

strict equality (===) operator, 405

strict not-equal-to (!==) operator, 405

stringify() method, 124

stringlike properties, 156

strings

- assigning to variable values, 25
- Boolean values, 73-74
- concatenating, 27-28, 72
- converting
  - to numbers, 69
  - to uppercase/lowercase, 73
- definition of, 70
- escape sequences, 70-71

extending, 114
JSON strings
creating, 124
deserializing, 123, 133
parsing, 125-126
maximum length of, 80
methods, 71-72, 406
concat(), 71-72
indexOf(), 71-72
lastIndexOf(), 71-72
replace(), 71-72
split(), 71-73
substr(), 72-73
toLowerCase(), 72-73
toUpperCase(), 72-73
operators, 403
replacing substrings in, 72
splitting, 73
style property (DOM), 204-207
styles (CSS)
advantages of, 201-202
DOM style property, 204-207
style declarations
placing, 204
syntax, 202-203
style properties, setting, 205-206
stylesheets
DOM styleSheets object, 209
enabling/disabling, 209-211
selecting, 211-214
switching, 209-211
stylesheets
DOM styleSheets object, 209
enabling/disabling, 209-211
selecting, 211-214
switching, 209-211
substr() method, 72-73
substring method, 406
subtraction (-) operator, 26
<summary> element, 366
supporting users with JavaScript disabled, 232
.swf files, 355, 358
switch statement, 88-89
switching stylesheets, 209-211

T

tabbed interfaces, 346-348
tabs widget, 346-348
tabs() method, 347
tags. *See* elements
tan method, 407
telltime() function, 63, 228-230
temperature.html, 29
testing
Ajax across multiple platforms, 296
cookies
cookies.js, 162-163
cookietest.html, 163-165
cookietest2.html, 164-165
for equality, 87-88
in multiple browsers, 18
multiple conditions, 88
text nodes, 173
text parsing, 285
text() method, 320-321
ThemeRoller, 336-337
this keyword, 105-107
time
reading date and time, 62-64
setting/editing, 58-59
time zones, 65
timers, 242
title, selecting stylesheets by, 213-214
toDateString() method, 59, 409
toggle() method, 322
toggleClass() function, 208
toggling between stylesheets, 210-211
toLocaleDateString() method, 409
toLocaleString() method, 409
toLocaleTimeString() method, 409
toLowerCase method, 406
tools
Firebug, 401
jEdit, 400
JSLint, 401
Geany, 400
Notepad++, 399
SciTE, 400
WDG (Web Design Group) validation service, 401
toString() method, 75-76, 409
TotalFrames() method, 359
toTimeString() method, 59, 409
toUpperCase() method, 72-73, 406
toUTCString() method, 409

transformations, animating, 239-240, 246

transitions, animating, 239-240, 246, 323

transparency, animating, 238-239

try statement, 227-228

Try.these function, 308

turning off JavaScript, 232

type attribute

- events, 143
- <script> element, 10

typeof operator, 117-118, 226, 406

U

undefined values, 74

unescape() function, 156-157

unescaping data, 156-157

Universal Time (UTC), 65

unobtrusive JavaScript

- converting code into, 228-231
- explained, 223-226

uppercase, converting strings to, 73

user interaction

- alert dialogs, 49
- confirmation dialogs, 50
- prompts, 50-51

user messages, outputting with functions, 41-42

users with JavaScript disabled, supporting, 232

UTC (Universal Time), 65

UTC() method, 410

V

validators, 400-401

valueOf() method, 410

values

- assigning to variables, 25
- Boolean values, 73-74
- cookie values
  - cookieName, 157
  - cookieValue, 157
  - multiple values, 165-166
- node values, 176-177
- null values, 51, 74
- returning from functions, 43
- undefined values, 74

variables

- assigning values to, 25
- explained, 24-25
- naming, 25
- scope, 44-46
- undefined, 74

VBScript, 9

video

- browser plug-ins, 355-358
- formats, 354-355
- loading and playing
  - with <a> element, 356
  - with <embed> element, 357
  - with <object> element, 357
  - with <video> element, 366-368
- streaming, 363

<video> element, 366-368

void operator, 406

volume attribute (<video> element), 367

W

W3C (World Wide Web Consortium), 10, 401

.wav files, 354

WDG (Web Design Group) validation service, 401

web browsers. *See* browsers

web pages

- banner cycling script, 97-100
- drawing on, 370-372
- including Ajax library in, 278-280
- including JavaScript in, 21-23
- including jQuery UI in, 336-337
- including libraries in, 314
- menus, creating on-the-fly, 193-197
- navigating with location object, 54
- reloading, 54
- static pages, 7

web servers, PHP-enabled, 282

WebM, 367

while loop, 90-91

whitespace, 34, 175-176

whole numbers, 67

widgets

- accordian, 343-344
- customized widgets, 349
- date picker, 344-346
- tabs, 346-348

window object, 12-13

alert() method, 14, 49

confirm() method, 50

prompt() method, 50-51

with keyword, 61-62

.wma files, 354

.wmv files, 354

World Wide Web Consortium (W3C), 10, 401

write() method, 14-15

writing

comments, 218-219

cookies, 158-161

Google Chrome extensions, 384-386

X

XAMPP, 282

XHTML, 31

XML (Extensible Markup Language)

compared to JSON, 122

parsing, 285

XMLHttpRequest object

creating instances of, 256-258, 274-275

methods, 258-259

open(), 259

send(), 259-260

overview, 253-256

properties, 258-259

responseText, 265-266

responseXML, 265-267

XOR operators (^), 404

Y-Z

Yahoo! UI Library, 305

zero fill right shift (>>>) operator, 404

Zoom() method, 359