Contents at a Glance

Part I: Getting Started on the Web
LESSON 1 Understanding How the Web Works .................................................. 1
2 Structuring an HTML Document ..................................................... 23
3 Understanding Cascading Style Sheets ............................................. 55
4 Understanding JavaScript .............................................................. 79
5 Validating and Debugging Your Code .............................................. 97

Part II: Building Blocks of Practical Web Design
LESSON 6 Working with Fonts, Text Blocks, Lists, and Tables ................... 121
7 Using External and Internal Links ................................................. 167
8 Working with Colors, Images, and Multimedia .................................. 191

Part III: Advanced Web Page Design with CSS
LESSON 9 Working with Margins, Padding, Alignment, and Floating .......... 249
10 Understanding the CSS Box Model and Positioning ............................. 271
11 Using CSS to Do More with Lists, Text, and Navigation .................... 289
12 Creating Layouts Using Modern CSS Techniques .................................. 317
13 Taking Control of Backgrounds and Borders ...................................... 353
14 Using CSS Transformations and Transitions ..................................... 383
15 Animating with CSS and the Canvas ............................................. 401

Part IV: Responsive Web Design
LESSON 16 Understanding the Importance of Responsive Web Design .......... 427
17 Designing for Mobile Devices ..................................................... 443
18 Using Media Queries and Breakpoints ............................................. 471
**Part V: Getting Started with Dynamic Sites**

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Understanding Dynamic Websites and HTML5 Applications</td>
<td>487</td>
</tr>
<tr>
<td>20</td>
<td>Getting Started with JavaScript Programming</td>
<td>507</td>
</tr>
<tr>
<td>21</td>
<td>Working with the Document Object Model (DOM)</td>
<td>523</td>
</tr>
<tr>
<td>22</td>
<td>Using JavaScript Variables, Strings, and Arrays</td>
<td>551</td>
</tr>
<tr>
<td>23</td>
<td>Controlling Flow with Conditions and Loops</td>
<td>595</td>
</tr>
<tr>
<td>24</td>
<td>Responding to Events and Using Windows</td>
<td>617</td>
</tr>
<tr>
<td>25</td>
<td>JavaScript Best Practices</td>
<td>655</td>
</tr>
<tr>
<td>26</td>
<td>Using Third-Party JavaScript Libraries and Frameworks</td>
<td>681</td>
</tr>
</tbody>
</table>

**Part VI: Advanced Website Functionality and Management**

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Working with Web-Based Forms</td>
<td>695</td>
</tr>
<tr>
<td>28</td>
<td>Organizing and Managing a Website</td>
<td>729</td>
</tr>
</tbody>
</table>

Index .................................................................................................................. 751
# Table of Contents

## Part I: Getting Started on the Web

### LESSON 1: Understanding How the Web Works  

- A Brief History of HTML and the World Wide Web .......................... 2
- Creating Web Content ....................................................................... 2
- Understanding Web Content Delivery ............................................. 3
- Selecting a Web Hosting Provider .................................................... 6
- Testing with Multiple Web Browsers and Devices .......................... 8
- Creating a Sample File ..................................................................... 9
- Using FTP to Transfer Files .............................................................. 10
- Understanding Where to Place Files on the Web Server .................. 14
- Distributing Content Without a Web Server ................................... 17
- Tips for Testing Web Content ......................................................... 18
- Summary ......................................................................................... 19
- Q&A ................................................................................................. 20
- Workshop ......................................................................................... 20
- Exercises ........................................................................................ 22

### LESSON 2: Structuring an HTML Document  

- Getting Prepared ............................................................................. 24
- Getting Started with a Simple Web Page ......................................... 25
- HTML Tags Every Web Page Must Have ......................................... 28
- Organizing a Page with Paragraphs and Line Breaks ..................... 31
- Organizing Your Content with Headings ....................................... 33
- Understanding Semantic Elements ................................................. 36
- Using `<header>` in Multiple Ways ............................................... 42
- Understanding the `<section>` Element .......................................... 44
- Using `<article>` ............................................................................. 45
Implementing the `<nav>` Element ............................................................. 45
When to Use `<aside>` ............................................................................. 47
Using `<footer>` Effectively .............................................................. 48
Summary ............................................................................................... 49
Q&A ..................................................................................................... 50
Workshop .............................................................................................. 51
Exercises ................................................................................................ 53

LESSON 3: Understanding Cascading Style Sheets 55
How CSS Works ....................................................................................... 56
A Basic Style Sheet .................................................................................. 57
A CSS Style Primer ................................................................................... 63
Using Style Classes .................................................................................. 68
Using Style IDs ....................................................................................... 70
Internal Style Sheets and Inline Styles ......................................................... 71
Summary ............................................................................................... 74
Q&A ..................................................................................................... 75
Workshop .............................................................................................. 75
Exercises ................................................................................................ 77

LESSON 4: Understanding JavaScript 79
Learning Web Scripting Basics .................................................................. 80
How JavaScript Fits into a Web Page .......................................................... 81
Exploring JavaScript’s Capabilities ............................................................ 84
Displaying Time with JavaScript ............................................................... 85
Testing the Script .................................................................................... 89
Summary ............................................................................................... 93
Q&A ..................................................................................................... 93
Workshop .............................................................................................. 94
Exercises ................................................................................................ 96

LESSON 5: Validating and Debugging Your Code 97
Validating Your Web Content ................................................................... 97
Debugging HTML and CSS Using Developer Tools ....................................... 99
Debugging JavaScript Using Developer Tools ............................................ 112
Summary ................................................................. 118
Q&A ........................................................................... 118
Workshop ..................................................................... 118
Exercises ...................................................................... 120

Part II: Building Blocks of Practical Web Design

LESSON 6: Working with Fonts, Text Blocks, Lists, and Tables 121
  Working with Special Characters .................................. 122
  Boldface, Italic, and Special Text Formatting .................. 126
  Tweaking the Font .......................................................... 129
  Using Web Fonts ............................................................... 133
  Aligning Text on a Page .................................................. 136
  The Three Types of HTML Lists ..................................... 139
  Placing Lists Within Lists .............................................. 142
  Creating a Simple Table .................................................. 147
  Controlling Table Sizes .................................................. 151
  Alignment and Spanning Within Tables ......................... 154
  Page Layout with Tables ................................................ 157
  Using CSS Columns ....................................................... 158
  Summary ..................................................................... 162
  Q&A ........................................................................... 164
  Workshop ..................................................................... 165
  Exercises ..................................................................... 166

LESSON 7: Using External and Internal Links 167
  Using Web Addresses ..................................................... 167
  Linking Within a Page Using Anchors ......................... 170
  Linking Between Your Own Web Content ..................... 174
  Linking to Non-HTML Files .......................................... 177
  Linking to External Web Content .................................. 178
  Linking to an Email Address ......................................... 179
  Opening a Link in a New Browser Window .................... 180
  Giving Titles to Links ..................................................... 181
Using CSS to Style Hyperlinks ................................................................. 182
Using Links Effectively ........................................................................... 186
Summary ............................................................................................. 187
Q&A ................................................................................................... 188
Workshop ............................................................................................ 189
Exercises .............................................................................................. 190

LESSON 8: Working with Colors, Images, and Multimedia 191
Best Practices for Choosing Colors ........................................................... 192
Understanding Web Colors ..................................................................... 194
Using Hexadecimal Values for Colors ...................................................... 196
Using RGB and RGBA Values for Colors .................................................... 197
Using CSS to Set Background, Text, and Border Colors ............................... 199
Choosing Graphics Software .................................................................. 201
The Least You Need to Know About Graphics ............................................ 202
Preparing Photographic Images .............................................................. 203
Creating Banners and Buttons ................................................................. 210
Optimizing Images by Reducing or Removing Colors ................................. 211
Creating Tiled Background Images .......................................................... 212
Placing Images on a Web Page .................................................................. 214
Describing Images with Text ................................................................... 217
Specifying Image Height and Width ........................................................ 218
Aligning Images ................................................................................... 219
Turning Images into Links ....................................................................... 223
Using Background Images ...................................................................... 226
Using Image Maps ................................................................................ 227
Linking to Multimedia Files .................................................................... 233
Embedding Multimedia Files .................................................................. 237
Additional Tips for Using Multimedia ...................................................... 242
Summary ............................................................................................. 242
Q&A ................................................................................................... 245
Workshop ............................................................................................ 246
Exercises .............................................................................................. 248
Part III: Advanced Web Page Design with CSS

LESSON 9: Working with Margins, Padding, Alignment, and Floating 249

Using Margins ................................................................. 249
Padding Elements ............................................................ 257
Keeping Everything Aligned ........................................... 261
Centering Blocks of Content .......................................... 262
Understanding the float Property ...................................... 263
Summary ............................................................................. 267
Q&A ..................................................................................... 267
Workshop ............................................................................. 267
Exercises ............................................................................ 270

LESSON 10: Understanding the CSS Box Model and Positioning 271

The CSS Box Model .......................................................... 271
Changing the Box Model ................................................... 275
The Whole Scoop on Positioning ........................................ 276
Controlling the Way Things Stack Up .............................. 281
Managing the Flow of Text ............................................... 284
Summary ............................................................................. 285
Q&A ..................................................................................... 285
Workshop ............................................................................. 286
Exercises ............................................................................ 288

LESSON 11: Using CSS to Do More with Lists, Text, and Navigation 289

HTML List Refresher .......................................................... 290
How the CSS Box Model Affects Lists .............................. 290
Placing List Item Indicators .............................................. 294
Creating Image Maps with List Items and CSS .................. 296
How Navigation Lists Differ from Regular Lists ............... 299
Creating Vertical Navigation with CSS .............................. 300
Creating Horizontal Navigation with CSS .......................... 310
Summary ............................................................................. 314
Q&A ..................................................................................... 314
Workshop ............................................................................. 315
Exercises ............................................................................ 316
## LESSON 15: Animating with CSS and the Canvas

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding CSS Animation</td>
<td>401</td>
</tr>
<tr>
<td>Using the CSS Canvas</td>
<td>410</td>
</tr>
<tr>
<td>Choosing Between CSS Animation and Canvas Animation</td>
<td>423</td>
</tr>
<tr>
<td>Summary</td>
<td>424</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>424</td>
</tr>
<tr>
<td>Workshop</td>
<td>424</td>
</tr>
<tr>
<td>Exercises</td>
<td>426</td>
</tr>
</tbody>
</table>

## Part IV: Responsive Web Design

### LESSON 16: Understanding the Importance of Responsive Web Design

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is Responsive Web Design?</td>
<td>427</td>
</tr>
<tr>
<td>Writing HTML for Responsive Web Design</td>
<td>435</td>
</tr>
<tr>
<td>Validating HTML, CSS, and JavaScript</td>
<td>438</td>
</tr>
<tr>
<td>Summary</td>
<td>439</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>440</td>
</tr>
<tr>
<td>Workshop</td>
<td>440</td>
</tr>
<tr>
<td>Exercises</td>
<td>442</td>
</tr>
</tbody>
</table>

### LESSON 17: Designing for Mobile Devices

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designing for Mobile Devices</td>
<td>443</td>
</tr>
<tr>
<td>Understanding Mobile First Design</td>
<td>451</td>
</tr>
<tr>
<td>Using Responsive Tables and Images</td>
<td>455</td>
</tr>
<tr>
<td>Creating Responsive Layouts Without Media Queries</td>
<td>464</td>
</tr>
<tr>
<td>Alternatives for Mobile Design Besides RWD</td>
<td>466</td>
</tr>
<tr>
<td>Summary</td>
<td>468</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>469</td>
</tr>
<tr>
<td>Workshop</td>
<td>469</td>
</tr>
<tr>
<td>Exercise</td>
<td>470</td>
</tr>
</tbody>
</table>

### LESSON 18: Using Media Queries and Breakpoints

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is a Media Query?</td>
<td>471</td>
</tr>
<tr>
<td>Using Media Query Expressions</td>
<td>476</td>
</tr>
<tr>
<td>What Is a Breakpoint?</td>
<td>477</td>
</tr>
</tbody>
</table>
### Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing Browser History</td>
<td>528</td>
</tr>
<tr>
<td>Working with the <code>location</code> Object</td>
<td>530</td>
</tr>
<tr>
<td>More About the DOM Structure</td>
<td>531</td>
</tr>
<tr>
<td>Working with DOM Nodes</td>
<td>534</td>
</tr>
<tr>
<td>Creating Positionable Elements (Layers)</td>
<td>536</td>
</tr>
<tr>
<td>Hiding and Showing Objects</td>
<td>541</td>
</tr>
<tr>
<td>Modifying Text in a Page</td>
<td>543</td>
</tr>
<tr>
<td>Adding Text to a Page</td>
<td>545</td>
</tr>
<tr>
<td>Summary</td>
<td>547</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>547</td>
</tr>
<tr>
<td>Workshop</td>
<td>548</td>
</tr>
<tr>
<td>Exercises</td>
<td>550</td>
</tr>
<tr>
<td><strong>LESSON 22: Using JavaScript Variables, Strings, and Arrays</strong></td>
<td>551</td>
</tr>
<tr>
<td>Using Variables</td>
<td>552</td>
</tr>
<tr>
<td>Understanding Expressions and Operators</td>
<td>555</td>
</tr>
<tr>
<td>Data Types in JavaScript</td>
<td>556</td>
</tr>
<tr>
<td>Converting Between Data Types</td>
<td>557</td>
</tr>
<tr>
<td>Using <code>String</code> Objects</td>
<td>558</td>
</tr>
<tr>
<td>Working with Substrings</td>
<td>562</td>
</tr>
<tr>
<td>Using Numeric Arrays</td>
<td>564</td>
</tr>
<tr>
<td>Using String Arrays</td>
<td>565</td>
</tr>
<tr>
<td>Sorting a Numeric Array</td>
<td>567</td>
</tr>
<tr>
<td>Using Functions</td>
<td>570</td>
</tr>
<tr>
<td>Introducing Objects</td>
<td>575</td>
</tr>
<tr>
<td>Using Objects to Simplify Scripting</td>
<td>577</td>
</tr>
<tr>
<td>Extending Built-in Objects</td>
<td>582</td>
</tr>
<tr>
<td>Using the <code>Math</code> Object</td>
<td>583</td>
</tr>
<tr>
<td>Working with <code>Math</code> Methods</td>
<td>585</td>
</tr>
<tr>
<td>Working with Dates</td>
<td>587</td>
</tr>
<tr>
<td>Summary</td>
<td>590</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>590</td>
</tr>
<tr>
<td>Workshop</td>
<td>591</td>
</tr>
<tr>
<td>Exercises</td>
<td>594</td>
</tr>
</tbody>
</table>
LESSON 23: Controlling Flow with Conditions and Loops 595
- The if Statement .......................................................... 595
- Using Shorthand Conditional Expressions .................... 599
- Testing Multiple Conditions with if and else ............... 600
- Using Multiple Conditions with switch ...................... 602
- Using for Loops ............................................................ 604
- Using while Loops ....................................................... 606
- Using do...while Loops ................................................. 607
- Working with Loops ..................................................... 607
- Looping Through Object Properties ............................. 609
- Summary ........................................................................... 612
- Q&A ................................................................................ 612
- Workshop ........................................................................ 613
- Exercises ........................................................................ 615

LESSON 24: Responding to Events and Using Windows 617
- Understanding Event Handlers ....................................... 618
- Using Mouse Events ..................................................... 623
- Using Keyboard Events ................................................ 627
- Using the load and unload Events ................................. 630
- Using click to Change the Appearance of a <div> ........... 631
- Controlling Windows with Objects ............................ 638
- Moving and Resizing Windows ..................................... 643
- Using Timeouts ............................................................. 645
- Displaying Dialog Boxes .............................................. 648
- Summary ........................................................................... 650
- Q&A ................................................................................ 650
- Workshop ........................................................................ 651
- Exercises ........................................................................ 654

LESSON 25: JavaScript Best Practices 655
- Scripting Best Practices ................................................. 655
- Reading Browser Information ...................................... 666
- Cross-Browser Scripting ............................................... 669
# Table of Contents

Supporting Non-JavaScript-Enabled Browsers ........................................... 671  
Creating an Unobtrusive Script ............................................................... 674  
Summary ............................................................................................. 677  
Q&A ................................................................................................... 677  
Workshop ............................................................................................ 677  
Exercises .............................................................................................. 680

**LESSON 26: Using Third-Party JavaScript Libraries and Frameworks** 681  
Using Third-Party JavaScript Libraries ...................................................... 681  
Adding JavaScript Effects by Using a Third-Party Library ............................ 686  
Using JavaScript Frameworks .................................................................. 689  
Summary ............................................................................................. 691  
Q&A ................................................................................................... 691  
Workshop ............................................................................................ 692  
Exercises .............................................................................................. 694

**Part VI: Advanced Website Functionality and Management**

**LESSON 27: Working with Web-Based Forms** 695  
How HTML Forms Work ......................................................................... 695  
Creating a Form ................................................................................... 696  
Accepting Text Input ............................................................................. 702  
Naming Each Piece of Form Data ............................................................ 703  
Labeling Each Piece of Form Data ........................................................... 703  
Grouping Form Elements ....................................................................... 705  
Exploring Form Input Controls ............................................................... 706  
Using HTML5 Form Validation ................................................................ 716  
Submitting Form Data ........................................................................... 718  
Accessing Form Elements with JavaScript ................................................. 720  
Summary ............................................................................................. 723  
Q&A ................................................................................................... 725  
Workshop ............................................................................................ 725  
Exercises .............................................................................................. 728
About the Authors

Jennifer Kyrnin consults professionally on web design and web development. She has built and maintained websites of all sizes, from small single-page sites to large million-page database-driven sites for international audiences. She has taught HTML, XML, and web design online since 1997 and is the author of *Sams Teach Yourself HTML5 Mobile Application Development*, *Sams Teach Yourself Responsive Web Design*, and *Sams Teach Yourself Bootstrap*.

Julie Meloni is a software development manager and technical consultant living in Washington, DC. She has written several books and articles on web-based programming languages and database topics, including the bestselling *Sams Teach Yourself PHP, MySQL, and Apache All in One*. 
Dedication

To Mark and Jaryth. As usual, you inspired me and kept me writing.

Acknowledgments

I would like to thank Mark Taber for thinking of me when this new edition came out. I couldn't have had a good-looking book without Kitty Wilson, my extraordinary copy editor. Any errors are not her fault. And thanks also go to Julie Meloni for believing that I could take over her book effectively. I hope I live up to her reputation.
What You’ll Learn in This Lesson:

- What web scripting is and what it’s good for
- How scripting and programming are different (and similar)
- What JavaScript is and where it came from
- How to include JavaScript commands in a web page
- What JavaScript can do for your web pages
- Beginning and ending scripts
- Formatting JavaScript statements
- How a script can display a result
- Including a script within a web document
- Testing a script in a browser
- Modifying a script
- Dealing with errors in scripts
- Moving scripts into separate files

The World Wide Web (WWW) began as a text-only medium; the first browsers didn’t even support images within web pages. The Web has come a long way since those early days. Today’s websites include a wealth of visual and interactive features in addition to useful content: graphics, sounds, animation, and video. Using web scripting languages, such as JavaScript, is one of the easiest ways to spice up a web page and to interact with users in new ways.

The first part of this lesson introduces the concept of web scripting and the JavaScript language. As the lesson moves ahead, you’ll learn how to include JavaScript commands directly in your HTML documents and how your scripts will be executed when the page is viewed in a browser. You will work with a simple script, edit it, and test it in your browser, all the while learning the basic tasks involved in creating and using JavaScript scripts.
LESSON 4: Understanding JavaScript

Learning Web Scripting Basics

You already know how to use two types of computer languages: HTML and CSS. You use HTML tags to describe how you want your document formatted. Then you use CSS to describe how you want the document displayed, and the browser shows the decorated content to the user. But because HTML and CSS are simple text-based languages, they can’t respond to the user, make decisions, or automate repetitive tasks. Interactive tasks such as these require a more sophisticated language: a programming language or a scripting language.

Although many programming languages are complex, scripting languages are generally simple. They have a simple syntax, can perform tasks with a minimum of commands, and are easy to learn. JavaScript is a web scripting language that enables you to combine scripting with HTML and CSS to create interactive web pages.

Scripts and Programs

A movie or a play follows a script—a list of actions (or lines) for the actors to perform. A web script provides the same type of instructions for the web browser. A script in JavaScript can range from a single line to a full-scale application. (In either case, JavaScript scripts usually run within a browser.)

Some programming languages must be compiled, or translated, into machine code before they can be executed. JavaScript, on the other hand, is an interpreted language: The browser executes each line of script as it comes to it.

There is one main advantage to interpreted languages: Writing or changing a script is very simple. Changing a JavaScript script is as easy as changing a typical HTML document, and the change is enacted as soon as you reload the document in the browser.

NOTE

Interpreted languages have disadvantages, too: They can’t execute really quickly, so they’re not ideally suited for complicated work, such as graphics, and they require the interpreter (in JavaScript’s case, usually a browser) in order to work.

Introducing JavaScript

JavaScript was developed in 1995 by Netscape Communications Corporation, the maker of the long-defunct Netscape web browser. JavaScript was the first web scripting language to be supported by browsers, and it is still far the most popular.
How JavaScript Fits into a Web Page

**NOTE**
A bit of history: JavaScript was originally called LiveScript, and it was first introduced in Netscape Navigator 2.0 in 1995. It was soon renamed JavaScript to indicate a marketing relationship with Sun’s Java language, although there is no other relationship, structurally or otherwise, between Java and JavaScript.

JavaScript is almost as easy to learn as HTML, and it can be included directly in HTML documents. Here are a few of the things you can do with JavaScript:

- Display messages to the user as part of a web page, in the browser’s status line, or in alert boxes
- Validate the contents of a form and make calculations (for example, by having an order form automatically display a running total as you enter item quantities)
- Animate images or create images that change when you move the mouse over them
- Create ad banners that interact with the user rather than simply displaying a graphic
- Detect what browser is in use or what features the browser has and perform advanced functions only on browsers that support them
- Detect installed plug-ins and notify the user if a plug-in is required
- Modify all or part of a web page without requiring the user to reload it
- Display or interact with data retrieved from a remote server

You can do all this and more with JavaScript, including creating entire applications. We’ll explore the uses of JavaScript throughout these lessons.

How JavaScript Fits into a Web Page

By using the `<script>` tag, as shown in Listing 4.1, you can add a short script (in this case, just one line) to a web document. The `<script>` tag tells the browser to start treating the text as a script, and the closing `</script>` tag tells the browser to return to HTML mode. In most cases, you can’t use JavaScript statements in an HTML document except within `<script>` tags. The exception is event handlers, described later in this lesson.

**LISTING 4.1**  A Simple HTML Document with a Simple Script

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>The American Eggplant Society</title>
</head>
```

JavaScript's `document.write` statement, which you'll learn more about later, sends output as part of the web document. In this case, it displays the modification date of the document, as shown in Figure 4.1.

In this example, we placed the script within the body of the HTML document. There are actually four places where you might use scripts:

- **In the body of the page**—In this case, the script's output is displayed as part of the HTML document when the browser loads the page.
In the header of the page, between the `<head>` tags—Scripts in the header should not be used to create output within the `<head>` section of an HTML document, since that would likely result in poorly formed and invalid HTML documents, but these scripts can be referred to by other scripts here and elsewhere. The `<head>` section is often used for functions—groups of JavaScript statements that can be used as a single unit. You will learn more about functions in Lesson 20, “Getting Started with JavaScript Programming.”

Within an HTML tag, such as `<body>` or `<form>`—This is called an event handler, and it enables the script to work with HTML elements. When using JavaScript in event handlers, you don’t need to use the `<script>` tag. You’ll learn more about event handlers in Lesson 20.

In a separate file entirely—JavaScript supports the use of files with the `.js` extension containing scripts; these can be included by specifying a file in the `<script>` tag. While using the `.js` extension is a convention, scripts can actually have any file extension, or none.

As you’ll learn in Lesson 25, “JavaScript Best Practices,” the best place to put JavaScript is inside the `<body>` tag, just before the closing `</body>` tag. This ensures that JavaScript is the last thing to load and so doesn’t disrupt the speed of the rest of the page displaying.

Using Separate JavaScript Files

When you create more complicated scripts, you’ll quickly find that your HTML documents become large and confusing. To avoid this problem, you can use one or more external JavaScript files. These are files with the `.js` extension that contain JavaScript statements.

External scripts are supported by all modern browsers. To use an external script, you specify its filename in the `<script>` tag, as shown here:

```html
<script src="filename.js"></script>
```

**NOTE**
The `type` attribute used to be required. But in HTML5 it can be left out if the script referenced is JavaScript.

Because in this case you’ll be placing the JavaScript statements in a separate file, you don’t need anything between the opening and closing `<script>` tags; in fact, anything between them will be ignored by the browser.

You can create the `.js` file by using the same text editor you use to write HTML and CSS. This file should contain one or more JavaScript commands and only JavaScript; it should not include `<script>` tags, other HTML tags, CSS, or HTML comments. Save the `.js` file in the same directory as the HTML documents that refer to it.
NOTE
External JavaScript files have a distinct advantage: You can link to the same .js file from two or more HTML documents. The browser stores this file in its cache, which can reduce the time it takes your web pages to display. But remember that every linked file requires an additional request to the server. So try to keep all your scripts in as few files as you can.

Understanding JavaScript Events

Many of the useful things you can do with JavaScript involve interacting with the user, and that means responding to events—for example, a link or a button being clicked. You can define event handlers within HTML tags to tell the browser how to respond to an event. For example, Listing 4.2 defines a button that displays a message when clicked.

LISTING 4.2  A Simple Event Handler

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Event Test</title>
  </head>
  <body>
    <h1>Event Test</h1>
    <button type="button"
      onclick="alert('You clicked the button. ')">
      Click Me!</button>
  </body>
</html>
```

In various places throughout these lessons, you’ll learn more about JavaScript’s event model and how to create simple and complex event handlers.

Exploring JavaScript’s Capabilities

If you’ve spent any time browsing the Web, you’ve undoubtedly seen lots of examples of JavaScript in action. The following sections provide some brief descriptions of typical applications for JavaScript, all of which you’ll explore in later lessons.

Improving Navigation

Some of the most common uses of JavaScript are in navigation systems for websites. You can use JavaScript to create a navigation tool—for example, a drop-down menu to select the next page to read or a submenu that pops up when you hover over a navigation link.
When it’s done right, this kind of JavaScript interactivity can make a site easier to use, even for browsers that don’t support JavaScript.

**Validating Forms**
Form validation is another common use of JavaScript, although the form validation features of HTML5 have stolen a lot of JavaScript’s thunder here as well. A simple script can read values the user types into a form and make sure they’re in the right format, such as with zip codes, phone numbers, and email addresses. This type of client-side validation enables users to fix common errors without waiting for a response from the web server, telling them that their form submission was invalid. You’ll learn how to work with form data in Lesson 27, “Working with Web-Based Forms.”

**Special Effects**
One of the earliest and most annoying uses of JavaScript was to create attention-getting special effects—for example, scrolling a message in the browser’s status line or flashing the background color of a page.

These techniques have fortunately fallen out of style, but thanks to the W3C DOM and the latest browsers, some more impressive effects are possible with JavaScript—for example, creating objects that can be dragged and dropped on a page or creating fading transitions between images in a slideshow. Some developers have HTML5, CSS3, and JavaScript working in tandem to create fully functioning interactive games.

**Remote Scripting (AJAX)**
For a long time, the biggest limitation of JavaScript was that there was no way for it to communicate with a web server. For example, you could use JavaScript to verify that a phone number had the right number of digits but not to look up the user’s location in a database based on the number.

Now that most browsers support some of JavaScript’s advanced features, this is no longer the case. Your scripts can get data from a server without loading a page, or they can send data back to be saved. These features are collectively known as AJAX (Asynchronous JavaScript and XML), or remote scripting.

**Displaying Time with JavaScript**
One common use of JavaScript is to display dates and times in the browser, and that’s where we’ll start putting some scripting pieces together. Because JavaScript runs on the browser, the times it displays will be in the user’s current time zone. However, you can also use JavaScript to calculate “universal” (UTC) time.
NOTE
UTC, which stands for Universal Time (Coordinated), is the atomic time standard based on the old GMT (Greenwich Mean Time) standard. This is the time at the prime meridian, which runs through Greenwich, London, England.

Your script, like most other JavaScript programs, begins with the HTML `<script>` tag. As you learned earlier in this lesson, you use the `<script>` and `</script>` tags to enclose a script within the HTML document.

CAUTION
Remember to include only valid JavaScript statements between the starting and ending `<script>` tags. If the browser finds anything except valid JavaScript statements within the `<script>` tags, it will display a JavaScript error message. You should use the comment indicator (`//`) in front of any lines that are not JavaScript.

To begin creating the script, open your favorite text editor and type the beginning and ending `<script>` tags, as shown here:

```html
<script>

In this script, you'll use JavaScript to determine the local and UTC times and then display them in the browser. Fortunately, all the hard parts, such as converting between date formats, are built in to the JavaScript interpreter; this is one of the reasons that displaying dates and times is a good starting place for beginners.

Storing Data in Variables
To begin the script, you will use a variable to store the current date. You will learn more about variables in Lesson 22, “Using JavaScript Variables, Strings, and Arrays,” but for now just understand that a variable is a container that can hold a value—a number, some text, or, in this case, a date.

To start writing the script, add the following line after the first `<script>` tag, making sure to use the same combination of uppercase and lowercase letters in your version because JavaScript commands and variable names are case sensitive:

```javascript
now = new Date();
```

This statement creates a variable called `now` and stores the current date and time in it. This statement and the others you will use in this script use JavaScript’s built-in `Date` object, which enables you to conveniently handle dates and times. You’ll learn more about working with dates in Lesson 22.
Displaying Time with JavaScript

NOTE
Notice the semicolon at the end of the code snippet creating a variable called now. This semicolon tells the browser that it has reached the end of a statement. Semicolons are optional, but using them helps you avoid some common errors. We’ll use them throughout these lessons for clarity.

Calculating the Results
Internally, JavaScript stores dates as the number of milliseconds since January 1, 1970. Fortunately, JavaScript includes a number of functions to convert dates and times in various ways, so you don’t have to figure out how to convert milliseconds to days, dates, or times.

To continue your script, add the following two statements before the final </script> tag:
localtime = now.toString();
utctime = now.toGMTString();

These statements create two new variables: localtime, containing the current time and date in a nice readable format, and utctime, containing the UTC equivalent.

NOTE
The localtime and utctime variables store a piece of text, such as January 1, 2001 12:00 PM. In programming parlance, a piece of text is called a string.

Creating Output
You now have two variables—localtime and utctime—which contain the results you want from your script. Of course, these variables don’t do you much good unless you can see them. JavaScript includes several ways to display information, and one of the simplest is by using the document.write statement.

The document.write statement displays a text string, a number, or anything else you throw at it. Because your JavaScript program will be used within a web page, the output will be displayed as part of the page. To display the result, add these statements before the final </script> tag:
document.write("<p><strong>Local time:</strong> " + localtime + "</p>");
document.write("<p><strong>UTC time:</strong> " + utctime + "</p>");

These statements tell the browser to add some text to the web page containing your script. The output will include some brief strings introducing the results and the contents of the localtime and utctime variables.

Notice the HTML elements, such as <p> and <strong>, within the quotation marks; because JavaScript’s output appears within a web page, it needs to be formatted using HTML.
NOTE
Notice the plus signs (+) used between the text and variables in the `document.write()` code snippets. In this case, each plus sign tells the browser to combine the values into one string of text. If you use the plus sign between two numbers that aren’t in quotes, they are added together.

Adding the Script to a Web Page
You should now have a complete script that calculates a result and displays it. Your script should match Listing 4.3.

LISTING 4.3 The Complete Date and Time Script
```html
<script>
  now = new Date();
  localtime = now.toString();
  utctime = now.toGMTString();
  document.write("<p><strong>Local time:</strong> " + localtime + "</p>");
  document.write("<p><strong>UTC time:</strong> " + utctime + "</p>");
</script>
```

To use your script, you need to add it to an HTML document. If you use the general template you’ve seen in the lessons so far, you should end up with something like Listing 4.4.

LISTING 4.4 The Date and Time Script in an HTML Document
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Displaying Times and Dates</title>
  </head>
  <body>
    <h1>Current Date and Time</h1>
    <script>
      now = new Date();
      localtime = now.toString();
      utctime = now.toGMTString();
      document.write("<p><strong>Local time:</strong> " + localtime + "</p>");
      document.write("<p><strong>UTC time:</strong> " + utctime + "</p>");
    </script>
  </body>
</html>
```

Now that you have a complete HTML document, save it with an .html extension.
Testing the Script

To test your script, you simply need to load the HTML document you created in a web browser. If you typed the script correctly, your browser should display the result of the script, as shown in Figure 4.2. (Of course, your result won’t be the same as mine, but it should be the same as the setting of your computer’s clock.)

![Current Date and Time](http://www.example.com/datetime.html)

**FIGURE 4.2**
Using JavaScript to display the date and time.

**NOTE**
With Internet Explorer, depending on your security settings, the script might not execute, and your browser might display a security warning. In this case, follow your browser’s instructions to allow your script to run. (This happens because the default security settings allow JavaScript in online documents but not in local files.)

Modifying the Script

Although the current script does indeed display the current date and time, its display isn’t nearly as attractive as the clock on your wall or desk. To remedy that situation, you can use some additional JavaScript features and a bit of HTML to display a large clock.

To display a large clock, you need the hours, minutes, and seconds in separate variables. Once again, JavaScript has built-in functions to do most of the work:

```javascript
hours = now.getHours();
mins = now.getMinutes();
secs = now.getSeconds();
```

These statements load the `hours`, `mins`, and `secs` variables with the components of the time using JavaScript’s built-in date functions.

After the hours, minutes, and seconds are in separate variables, you can create `document.write` statements to display them:

```javascript
document.write("<p><strong>");
document.write(hours + ":" + mins + ":" + secs);
document.write("</p></strong>");
```
The first statement displays an HTML `<h2>` header tag to display the clock as a second-level header element. The second statement displays the `hours`, `mins`, and `secs` variables, separated by colons, and the third adds the closing `</h2>` tag.

You can add the preceding statements to the original date and time script to add the large clock display. Listing 4.5 shows the complete modified version of the script.

**LISTING 4.5  The Date and Time Script with a Large Clock Display**

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Displaying Times and Dates</title>
</head>
<body>
<h1>Current Date and Time</h1>
<script>
  now = new Date();
  localtime = now.toString();
  utctime = now.toGMTString();
  document.write("<p><strong>Local time:</strong> 
                  " + localtime + "</p>);
  document.write("<p><strong>UTC time:</strong> 
                  " + utctime + "</p>);
  hours = now.getHours();
  mins = now.getMinutes();
  secs = now.getSeconds();
  document.write("<h2>
                 " + hours + ":" + mins + ":" + secs + 
                 "</h2>" );
</script>
</body>
</html>
```

Now that you have modified the script, save the HTML file and open the modified file in your browser. If you left the browser running, you can simply use the Reload button to load the new version of the script. Try it and verify that the same time is displayed in both the upper portion of the window and the new large clock. Figure 4.3 shows the results.
Dealing with JavaScript Errors

As you develop more complex JavaScript applications, you’re going to run into errors from time to time. JavaScript errors are usually caused by mistyped JavaScript statements.

To see an example of a JavaScript error message, you can modify the statement you added in the preceding section. In this example, we use a common error: omitting one of the parentheses. Change the last `document.write` statement in Listing 4.5 to read

```
document.write("</h2>");
```

Save your HTML document again and load the document into the browser. Depending on the browser version you’re using, one of two things will happen: Either an error message will be displayed, or the script will simply fail to execute.

If an error message is displayed, you’re halfway to fixing the problem by adding the missing parenthesis. If no error was displayed, you should configure your browser to display error messages so that you can diagnose future problems:

- In Firefox, you can select Tools, Web Developer, Web Console. The console displays the error message you created in this example, as shown in Figure 4.4.

- In Chrome, from the options menu (three horizontal dots on the right side of the browser bar) select More Tools, Developer Tools. A console displays in the bottom of the browser window. Choose the console tab if it’s not selected.
FIGURE 4.4
Showing an error in the JavaScript console in Firefox.

The error you get in this case, `SyntaxError: missing ) after argument list`, points to line 22. In this case, clicking the name of the document takes you directly to the highlighted line containing the error, as shown in Figure 4.5.

FIGURE 4.5
Firefox helpfully points out the offending line.

Most modern browsers contain JavaScript debugging tools such as the one you just witnessed. You’ll learn more about this in the next lesson.
Summary

During this lesson, you’ve learned what web scripting is and what JavaScript is. You’ve also learned how to insert a script into an HTML document or refer to an external JavaScript file, what sorts of things JavaScript can do, and how JavaScript differs from other web languages. You also wrote a simple JavaScript program and tested it using a web browser. You also learned how to modify and test scripts, and you saw what happens when a JavaScript program runs into an error.

In the process of writing this script, you have used some of JavaScript’s basic features: variables, the document.write statement, and functions for working with dates and times.

Now that you’ve learned a bit of JavaScript syntax, you’re ready to continue on to learn all sorts of things about web development before settling in to writing interactive websites using client-side scripting.

Q&A

Q. Do I need to test my JavaScript on more than one browser?

A. In an ideal world, any script you write that follows the standards for JavaScript will work in all browsers, and 98% of the time (give or take) that’s true in the real world. But browsers do have their quirks, and you should test your scripts in Chrome, Internet Explorer, and Firefox, as well as mobile devices running iOS and Android—at a minimum.

Q. If I plan to learn PHP, Ruby, or some other server-side programming language anyway, will I have any use for JavaScript?

A. Certainly. JavaScript is the ideal language for many parts of a web-based application, such as basic interactivity. Although PHP, Ruby, and other server-side languages have their uses, they can’t interact directly with the user on the client side.

Q. When I try to run my script, the browser displays the actual script in the browser window instead of executing it. What did I do wrong?

A. This is most likely caused by one of three errors. First, you might be missing the beginning or ending <script> tags. Check them and, if you use the type attribute, verify that it reads type="text/javascript". Second, your file might have been saved with a .txt extension, causing the browser to treat it as a text file. Rename it to have the extension .htm or .html to fix the problem. Third, make sure your browser supports JavaScript and ensure that it is not disabled in the preferences.

Q. Why are the <strong> and <p> tags allowed in the statements to print the time? I thought HTML tags aren’t allowed within the <script> tags.

A. Because these tags are inside quotation marks, they are considered a valid part of the script. The script’s output, including any HTML tags, is interpreted and displayed by the browser. You can use other HTML tags within quotation marks to add formatting, such as the <h2> tags you added for the large clock display.
Workshop

The workshop contains quiz questions and exercises to help you solidify your understanding of the material covered.

Quiz

1. When a user views a page containing a JavaScript program, which machine actually executes the script?
   a. The user’s machine running a web browser
   b. The web server
   c. A central machine deep within Netscape’s corporate offices
   d. A dedicated JavaScript server

2. What software do you use to create and edit JavaScript programs?
   a. A browser
   b. A text editor
   c. A pencil and a piece of paper
   d. A JavaScript editor

3. What are variables used for in JavaScript programs?
   a. Storing numbers, dates, or other values
   b. Varying randomly
   c. Causing high-school algebra flashbacks
   d. Changing the output of the script

4. What should appear at the very end of a JavaScript script embedded in an HTML file?
   a. The `<script>` tag
   b. The `</javascript>` tag
   c. The END statement
   d. The `</script>` tag

5. Which of these is not something you can do with JavaScript?
   a. Detect the features of the browser in use
   b. Modify part of a page without requiring a page refresh
   c. Write data to the remote server
   d. Interact with data from a remote server
6. Where can you place scripts?
   a. In the body of a page
   b. Within an HTML tag
   c. In a separate file
   d. All of the above

7. What do you use to include a separate script file in a page?
   a. `<link src="filename.js">`
   b. `<script src="filename.js"></script>`
   c. `<javascript src="filename.js">`
   d. `<include src="filename.js"></include>`

8. Which of these is an event handler?
   a. `<button>`
   b. `type="button"`
   c. `onclick="alert('You clicked the button.')"`
   d. `</button>`

9. What does the line `now = new Date();` do in JavaScript?
   a. It creates a variable called `now` and stores the current date in it.
   b. It creates a variable called `new` and stores the current date in it.
   c. It displays the current time.
   d. Nothing; it's not valid JavaScript.

10. Correct this line of JavaScript:
    ```javascript
        document.write("<p><strong>Dinner Time:</strong> " + localtime + "</p>);
    ```
    a. `document.write("<p><strong>Dinner Time:</strong> " + localtime + "</p>);`
    b. `document.write("<p><strong>Dinner Time:</strong> " + localtime + "</p>);`
    c. `document.write("<p><strong>Dinner Time:</strong> " + localtime + "</p>);`
    d. The line is correct as written.`
NOTE
Just a reminder for those of you reading these words in the print or e-book edition of this book: If you go to www.informit.com/register and register this book (using ISBN 9780672338083), you can receive free access to an online Web Edition that not only contains the complete text of this book but also features an interactive version of this quiz.

Answers
1. a. JavaScript programs execute in the web browser. (There is actually a server-side version of JavaScript, but that’s another story.)
2. b. You can use any text editor to create scripts.
3. a. Variables are used to store numbers, dates, or other values.
4. d. Your script should end with the <script> tag.
5. c. JavaScript can never write data to the remote server due to security concerns.
6. d. JavaScript can be added in the body, a tag, or in a separate file.
7. b. Use the line <script src="filename.js"></script>.
8. c. The onclick attribute and its value are the click event handler.
9. a. It creates a variable called now and stores the current date in it.
10. a. The final quotation mark is missing.

Exercises
- Add a millisecond field to the large clock. You can use the getMilliseconds function, which works just like getSeconds but returns milliseconds.
- Modify the script to display the time, including milliseconds, twice. Notice whether any time passes between the two time displays when you load the page.
analogous color schemes, 193
anchor HTML tags, linking
to anchor locations, 171–174
to email addresses, 179–180
to external web content, 178–179
identifying anchor locations
within web pages, 170–171
to non-HTML files, 177–178
between web content, 174–177
within web pages, 170
anchor objects (DOM), 528
Angular frameworks (JavaScript), 690
animations, 401–402
canvas, 420–424
keyframes, defining, 402–404
naming, 410
pausing, 410
repeating, 408–410
timing, 405–408
anonymous functions, event
handlers (JavaScript) and, 630
applications (HTML5), developing, 501
architectures (site), sample
build, 170
arithmetic mean, 587
arrays (JavaScript), 551
accessing elements of, 565
length of, 565
numeric arrays, 564–565, 567–569
string arrays, 565–566
sorting, 567
splitting, 566–567
<article> HTML tags, 37, 40, 45, 437
<aside> HTML tags, 37, 47–48, 437
attributes, HTML tags, 136, 168
audio, playing in web pages, 240–241
<audio> element, 240–241
<audio> HTML tags, 436
B
<b> HTML tags, 122, 126
Backbone.js frameworks
(JavaScript), 690
backgrounds (cells)
colors, 157
images, 157
backgrounds (lists), color and, 296
backgrounds (web pages), 353–354
color, 194
alternating colors, 364–365
changing with CSS, 199–201
hexadecimal color codes, 194, 195, 196–197
RGB color values, 194, 197–198
gradients, 365
linear gradients, 366
radial gradients, 367–371
images, 226–227
in multiple backgrounds, 355–359
positioning images, 361
multiple backgrounds, 355–359
placing, 359–360
scrolling, 361–364
sizing, 360
tiled background images,
creating, 212–214
bad website examples, 195
banners, creating, 210–211
BAWSI.org website, 737
Berners-Lee, Sir Tim, 2, 431
block-level elements, aligning text
in, 136–139
blogs, publishing web content
to, 18
<body> HTML tags, 29, 30, 436
body (tables), wrapping, 151
boldface text, 126, 127, 157
Boolean (logical) operators
(JavaScript), 557, 597–598
Bootstrap framework, 745
borders (tables)
collapsing, 149–150
creating, 149
spacing, 157
borders (web pages), 354
color, changing with CSS, 199–201
images as, 373
clipping border images, 373–375
defining image width, 375–376
extending border images
beyond border edge, 376
fitting to borders, 376–377
multiple borders, 355
box model (CSS), 271–275, 318
box-sizing property, changing,
275–276
image maps, creating,
296–299
lists, 290–291
creating, 291–292
image maps, creating,
296–299
margins, 293–294, 295
navigation lists, 299–300
navigation lists, horizontal
navigation, 310–314
navigation lists, multi-
level vertical navigation,
305–310
navigation lists, primary
navigation, 300
navigation lists, single-
level vertical navigation,
303–305
navigation lists, vertical
navigation, 300–303
padding, 292–294, 295
placing list item indicators,
294–296
styling, 291–292
codecs (video) 753

outline properties, 275
positioning elements in layout, 277–281
absolute positioning, 276–277
relative positioning, 276
web page elements, changing height/width, 272–274
box-sizing property (CSS), changing, 275–276
<b> HTML tags, 32, 129, 436
breakpoints
best practices, 483
debugging, 115, 116–118
defined, 471, 477
defining with media queries, 477–479
large screen-specific styles, adding with media queries, 482–483
optimal breakpoints, 483
scripting and, 115, 116–118
browsers
dishonest browsers, JavaScript and, 668–669
distributing, 18
history, accessing, 528–530
history objects (DOM), 528–530
JavaScript
attaching events across browsers, 660–661
avoiding browser specificity, 661
browser quirks, 671
cross-browser scripting, 669–672
dishonest browsers, 668–669
displaying browser information, 667–668
feature sensing, 670
non-JavaScript-enabled browsers, 671–672
<noscript> tags, 672
reading browser information, 666–667
non-JavaScript-enabled browsers, 671–672
opening links in new browser windows, 180–181
prefixes, 434
search engines, 672
server interaction, 3–6
testing, 8–10
timeouts, 645–647
web content, displaying, 141
window objects (DOM), 524, 531
windows
creating, 640
dialog boxes, displaying, 648–649
moving, 643–645
opening/closing, 640–643
timeouts, 645–647
built-in objects (JavaScript), 510
<button> element, web forms, 718–719
buttons
creating, 210–211
radio buttons, web forms, 708–709
canvas, 410–411
animations, 420–424
drawing
circles, 411–413
lines/polylines, 414–416
rectangles/squares, 411
triangles, 415–416
images, adding to, 417–419
cascading, 434
case sensitivity (text)
color names, 195
JavaScript, 514
web servers, 170
cells (tables)
aligning data, 154–156
backgrounds
colors, 157
images, 157
boldface text, 157
creating, 147
resizing in responsive tables, 456–457
spanning, 156
styling, 147
centering web page elements, 262–263
character entities. See also special characters, formatting, 123–125
check boxes, web forms, 706–708
child (nested) tags, HTML, 142
children (DOM objects), 533
Chrome (Google), Developer Tools, 101, 114–118
circles, drawing on canvas, 411–413
clearing, indenting code for, 742–743
class attributes, HTML tags, 136
click events, event handlers (JavaScript) and, 623–627, 631–638
client-side scripting, 488
<closing> HTML tags, 29
closing/opening, browser windows, 640–643
code (maintainable), writing
documenting code with comments, 740–742
indenting for clarity, 742–743
version control, 743–745
codecs (video), 237
collapsing borders (tables), 149–150

color
  analogous color schemes, 193
  backgrounds (lists), 296
  backgrounds (web pages), 194
  alternating colors, 364–365
  changing color with CSS, 199–201
  hexadecimal color codes, 194, 195, 196–197
  RGB color values, 194, 197–198
  borders, changing color with CSS, 199–201
  choosing (best practices), 192–194
  color wheels, 193
  complementary color schemes, 194
  font color, changing, 129, 130, 131–133
  graphics software, choosing, 201–202
  gray, spelling in CSS, 194
  hexadecimal color codes, 194, 195, 196–197
  highlighted text, 438
  images, tweaking in, 207–208
  monochromatic color schemes, 194
  names, case sensitivity, 195
  RGB color values, 195, 197–198
  screen resolution, 210
  text, changing color with CSS, 199–201
  text links, changing color, 195
  triadic color schemes, 194
  websites, bad website examples, 195

columns
  CSS, 158–162
    tables, rearranging in responsive tables, 457–460
    comma-separated lists, media query expressions, 476
    comments (JavaScript), 515–516, 740–742
    conditional expressions (JavaScript), 596–597, 599–600
    conditional operators (JavaScript), 597
    conditional statements (JavaScript), 511
    content/presentation/behavior, separating (JavaScript), 657
    contents, tables of, 730–731
    continuing loops (JavaScript), 609
    converting
      between data types (JavaScript), 557–558
      date formats with Date object (JavaScript), 590
    copyrights, images and, 202
    cropping images, 204–206
    CSS (Cascading Style Sheets), 55, 57, 62
      aligning web page elements, 261–262
      animations, 401–402
        canvas, 420–424
        keyframes, defining, 402–404
        naming, 410
        pausing, 410
        repeating, 408–410
        timing, 405–408
      backgrounds (web pages), 194–201
      changing color, 199–201
      basic style sheets, 57–60
      selectors, 60
    style rules, 60–61
    borders, changing color, 199–201
    box model, 271–275
    box-sizing property, changing, 275–276
    changing web page element height/width, 272–274
    outline properties, 275
    positioning elements in layout, 276–281

breakpoints
  best practices, 483
  defined, 471, 477
  defining with media queries, 477–479
  large screen-specific styles, adding with media queries, 482–483
  optimal breakpoints, 483

browser prefixes, 434

canvas, 410–411
  animations, 420–424
  drawing circles, 411–413
  drawing lines/polygons, 414–416
  drawing rectangles/squares, 411
  drawing triangles, 415–416
  images, adding to, 417–419

cascading, 434

centering web page elements, 262–263

color
  backgrounds (web pages), changing, 199–201
  gray, spelling, 194
  text color, changing, 199–201

columns, 158–162
CSS (Cascading Style Sheets) 755

creating, 63
debugging with Developer Tools, 107–112
defined, 56
display: table; property, 335–338
external style sheets, 56
float property and web page elements, 263–266
font sizes, 60, 61
frameworks, 745–746
HTML documents, linking to, 61–62
hyperlink styles, 182–186
inline styles, 72–73
internal style sheets, 56, 71–73
margins, adding to web page elements, 249–257
media queries
adding, 473
baseline styles, defining, 479–480
breakpoints, best practices, 483
breakpoints, defined, 471, 477
breakpoints, defining with media queries, 477–479
breakpoints, large screen-specific styles adding with media queries, 482–483
breakpoints, optimal breakpoints, 483
defined, 471–472
defining media type styles, 473–474
expressions, 476–477
handheld media type, 472
large screen-specific styles, adding, 482–483
media features, 474–476
print media type, 472–473
requesting multiple CSS documents, 474
retina devices and, 484
screen media type, 472
small screen-specific styles, adding, 481–482
types of, 472–474
ordering elements in layout, 281–284
outlines, 378
padding, adding to web page elements, 257–261
positioning elements in layout, 63–65
absolute positioning, 276–277
flowing text, 284–285
ordering elements, 281–284
relative positioning, 276
print style sheets, 472–473
requesting multiple CSS documents with media queries, 474
RWD, writing for, 438–439
selectors, 60
semicolons (;), 61, 63
style classes, 68–70
style ID, 70–71
style primer, 63
formatting properties, 63, 65–68
layout properties, 63–65
style rules, 56–57, 60–61
tables
accessibility, 158
borders, collapsing, 149–150
cells, aligning data, 154–156
cells, background colors, 157
cells, background images, 157
cells, boldface text, 157
cells, spanning, 156
cells, styling, 147
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
sizing, 151–153
unnecessary redesigns, 158
text
aligning, 67
changing color, 199–201
font properties, 67–68
indenting, 67
indenting text, 67
transformations
2D transformations, 383
2D transformations, moving elements, 386–388
2D transformations, multiple transformations, 391–392
2D transformations, rotating elements, 384
2D transformations, scaling elements, 385–386
2D transformations, slanting (skewing) elements, 388–391
3D transformations, 392–393
transitions, 393–396
timing, 396–397
triggering with JavaScript, 397–398
validating, 438–439
style sheets, 73
web content, 99, 109–111
z-index property, ordering elements in layout, 281–284
CSS box model, 318
- image maps, creating, 296–299
- lists, 290–291
  - creating, 291–292
  - image maps, creating, 296–299
- margins, 293–294, 295
- navigation lists, 299–300
- navigation lists, horizontal navigation, 310–314
- navigation lists, multi-level vertical navigation, 305–310
- navigation lists, primary navigation, 300
- navigation lists, single-level vertical navigation, 303–305
- padding, 292–294, 295
- placing list item indicators, 294–296
- styling, 291–292

CSS Flexible Box Layout module, 339–345
CSS Grid Layout module, 345–348
custom JavaScript objects, 510

data types (JavaScript), 556, 557
- Booleans, 557
- converting between, 557–558
- null data types, 557
- number data types, 556
- strings, 557
  - assigning values to, 559–560
  - calculating length of, 560–561
- converting case of, 561
- getting single characters, 563
- splitting, 566–567
- string objects, 558, 559
- substrings, 562
- substrings, finding, 563–564
- substrings, getting single characters, 563
- substrings, using parts of strings, 562–563
- using parts of strings, 562–563

Date object (JavaScript), 587–588
- converting date formats, 590
- creating, 588
- date values
  - reading, 588–589
  - setting, 588
- time zones, 589

<dd> HTML tags, 139, 141, 290
Debug panel (Firefox), 114–118
Debugger (Safari), 114–118
directories, 168
- absolute links, 169
- double dot (..), 169
- forward slash (/), 168
- levels of, 168
- relative addresses, 168, 169
- relative-root addresses, 168
- site architectures, sample build, 170
- subdirectories, 168
dishonest browsers, JavaScript and, 668–669
display: table; property (CSS), 335–338
displaying
  - web content, 141
  - web form data, 721–722

<dt> elements, changing appearance of with click events, 631–638
<dt> HTML tags, 136, 137, 436
<dd> HTML tags, 139, 141, 290
</ddtype> HTML tags, 435
document objects (DOM), 525, 531–532
- getting information about a document, 525–526
- writing text in a document, 527
documenting JavaScript code, 662–663, 740–742
Dojo, 686

DOM (Document Object Model), 495
- children, 533
- DOM objects (JavaScript), 510
- event handlers, mouse events, 623–627
- jQuery, 683–684
- nodes, 533, 534
  - basic properties, 534
  - document properties, 535
  - methods, 535–536
  - relationship properties, 534–535
objects
anchor objects, 528
do children, 533
document objects (DOM),
525–527, 531–532
event object, 621–623
events and, 618
hiding/showing, 541–543
history objects, 528–530
link objects, 527–528
location objects, 530–531
methods of, 524
notation, 524
parents, 533
properties of, 524
siblings, 533
window objects, 524, 531
parents, 533
positionable elements (layers),
536, 537–541
siblings, 533
structure of, 524, 531–533

<dt> HTML tags, 139, 141, 290
dynamic serving. See also RWD,
adaptive design, 467–468
dynamic websites
HTML5 applications,
developing, 501
images, changing based on
user interaction, 498–500
JavaScript, in HTML, 488–490
scripting
client-side scripting, 488
displaying random content,
491–495
DOM, 495, 496–498
hiding scripts, 490
placement of scripts, 489
server-side scripting, 488
types of, 487–488
unobtrusive JavaScript, 496
unobtrusive JavaScript and
DOM, 496–498

domains, mobile devices and
RWD, 468
double dot (..), directories, 169
do.while loops (JavaScript), 607
download speeds, mobile interfaces
and RWD, 449–450
downs/ups, mouse events,
623–627
drawing (canvas)
circles, 411–413
lines/polygons, 414–416
rectangles/squares, 411
triangles, 415–416
<dt> HTML tags, 139, 141, 290
dynamic serving. See also RWD,
adaptive design, 467–468
dynamic websites
HTML5 applications,
developing, 501
images, changing based on
user interaction, 498–500
JavaScript, in HTML, 488–490
scripting
client-side scripting, 488
displaying random content,
491–495
DOM, 495, 496–498
hiding scripts, 490
placement of scripts, 489
server-side scripting, 488
types of, 487–488
unobtrusive JavaScript, 496
unobtrusive JavaScript and
DOM, 496–498

<em> HTML tags, 126, 127, 436
e-mail addresses, linking web
content, 179–180

<dt> HTML tags, 139, 141, 290
dynamic serving. See also RWD,
adaptive design, 467–468
dynamic websites
HTML5 applications,
developing, 501
images, changing based on
user interaction, 498–500
JavaScript, in HTML, 488–490
scripting
client-side scripting, 488
displaying random content,
491–495
DOM, 495, 496–498
hiding scripts, 490
placement of scripts, 489
server-side scripting, 488
types of, 487–488
unobtrusive JavaScript, 496
unobtrusive JavaScript and
DOM, 496–498

<em> HTML tags, 126, 127, 436
e-mail addresses, linking web
content, 179–180

Ember frameworks (JavaScript),
691
<empty> HTML tags, 29
epochs, 588
error handling, JavaScript,
91–92, 662
escaping loops (JavaScript),
608–609
ESPN.com, 733–734
event handlers
HTML, 511–513
JavaScript, 511–513, 618
adding, 659–661
anonymous functions, 630
attaching events across
browsers, 660–661
click events, 623–627,
631–638
creating, 618–619
defining, 619–620
event object, 621–623
keyboard events,
627–630
load/unload events,
630–631
mouse events, 623–627
multiple event handlers,
supporting, 620–621
objects and events, 618
W3C event model,
659–660
web forms, text fields/text
areas, 713–714

event object (JavaScript),
621–623
events (JavaScript), 84
attaching events across
browsers, 660–661
W3C event model, 659–660
web forms, 719–720

expressions (JavaScript),
555
external style sheets, 56
feature sensing, JavaScript cross-browser scripting, 670
file management, 14–17
finding, substrings (JavaScript), 563–564
Firefox
  Debug panel, 114–118
  Developer Tools, 100
  JavaScript, error handling, 92
FireFTP, 10-11
fixed web page layouts, 319–322
fixed/liquid hybrid web page layouts
  creating, 324–326
  defining two columns in, 326–328
  height, setting, 329–335
  width, setting, 328–329
Flash, mobile interfaces and RWD, 445
Flexible Box Layout module (CSS), 339–345
flexible-width images, responsive images, mobile design, 461–462
float property, web page elements and, 263–266
flow control (JavaScript)
  conditional expressions, 596–597, 599–600
  conditional operators, 597
  if statements, 595–596
  conditional expressions, 596–597
  conditional operators, 597
  else keyword, 598–599
  logical (Boolean) operators, 597–598
  if.else statements
  testing multiple conditions, HTML file, 600–601
  testing multiple conditions, JavaScript file, 601–602
  logical (Boolean) operators, 597–598
  loops
    continuing, 609
    do.while loops, 607
    escaping, 608–609
    infinite loops, 608
    looping through object properties, 609–612
    for loops, 604–606
    while loops, 606–607
  shorthand conditional expressions, 599–600
  switch statements, multiple conditions and, 602–604
flowing text in web pages, 284–285
  <font> HTML tags, 122, 129
fonts
  color, changing, 129, 130, 131–133
  CSS
    properties, 67–68
    sizing, 60, 61
  font families (typefaces), changing, 130, 131–133
  Google Fonts, 134–135
  mobile interfaces and RWD, 447–449
  sizing, 60, 61, 129, 130, 131–133
  typefaces (font families), changing, 130, 131–133
  web fonts, 133–135
  italic text, 126–127
  monospaced text, 127, 128–129
  paragraph breaks, 136
  sample text, 122
  simple web pages, building with HTML, 33
  special characters, 122–125
  subscript text, 127
  superscript text, 127
  typefaces (font families), changing, 130, 131–133
  underlined text, 127–128
forms (web-based), 695–696
  accepting text input, 702–703
  accessing elements with JavaScript, 720–721
  <button> element, 718–719
  check boxes, 706–708
  creating, 696–702
  displaying data, 721–722
  events, 719–720
  form-processing scripts, 703
  grouping elements, 705
  hidden data, 705
  input controls, 706–712
  labeling data, 703–704
naming data, 703
<output> element, 719
pull-down pick lists, 710–712
radio buttons, 708–709
selection lists, 710–712
submitting data, 718
text fields/text areas, 713–715
validating, 85, 716–717
forward slash (/), HTML directories, 168
Foundation framework, 745
frameworks
Bootstrap framework, 745
CSS frameworks, 745–746
Foundation framework, 745
HTML frameworks, 745–746
HTML5 Boilerplate framework, 745
JavaScript, 690, 691
AJAX (remote scripting), 690
Angular frameworks, 690
Backbone.js frameworks, 690
Ember frameworks, 691
Knockout frameworks, 691
MVC pattern, 689–690
React frameworks, 691
FTP clients, 10
FireFTP, 10–11
selecting, 10–11
transferring files, 12–14
function calls (JavaScript), 508–509
functions (JavaScript), 570
calling, 571–573
defining, 570–571
naming, 515
returning values, 573–575

G

GIF format (images), 211–212
GIMP graphics software, 201–202
cropping images, 204–205, 206
JPEG compression, 209
resizing images, 206
global variables (JavaScript), 553
Gmail, 673
Google Chrome, Developer Tools, 101, 114–118
Google Developers, 665
Google Docs, version control, 743
Google Fonts, 134–135
graceful degradation, theory of, 428, 431, 658
gradients, backgrounds (web pages), 365
linear gradients, 366
radial gradients, 367–371
gradients, backgrounds (web pages), 365
grayscale images, Adobe Photoshop
choosing, 201–202
GIMP 201–202
cropping images, 204–205, 206
JPEG compression, 209
resizing images, 206
gray (color), spelling in CSS, 194
Grid Layout module (CSS), 345–348
grouping web form elements, 705

H

<h1> HTML tags, 436
<h2> HTML tags, 436
<h3> HTML tags, 436
handheld media type (media queries), 472
<h1> HTML tags, 30, 435, 489, 514
<header> HTML tags, 37, 39–43, 437
headers (tables), wrapping, 151
headings, simple web pages, building with HTML, 33–36
height/width
fixed/liquid hybrid web page layouts, setting in, 329–335
images, specifying in, 218–219
web page elements, changing in, 272–274
“Hello World!” sample file, creating, 9–10
helper applications, defined, 234
hexadecimal color codes, 194, 195, 196–197
highlighted text, 438
history objects (DOM), 528–530
horizontal image alignment, 219–221
horizontal navigation in navigation lists, creating, 310–314
HTML (Hypertext Markup Language)
backgrounds (web pages), changing color, 199–201
boldface text, 126, 127
borders, changing color, 199–201
<button> element, web forms, 718–719
color
backgrounds (web pages), changing, 199–201
text color, changing, 199–201
CSS, linking to, 61–62
debugging with Developer Tools, 102–107
directories, 168
absolute links, 169
double dot (..), 169
forward slash (/), 168
relative addresses, 168, 169
relative-root addresses, 168
site architectures, sample build, 170
subdirectories, 168
event handlers, 511–513
formatting text
aligning text, 136–139
attributes, HTML tags, 136, 168
boldface text, 126, 127
character entities, 123–125
font color, changing, 129, 130, 131–133
font sizes, 129, 130, 131–133
Google Fonts, 134–135
italic text, 126–127
monospaced text, 127, 128–129
sample text, formatting example, 122
special characters, 122–125
subscript text, 127
superscript text, 127
typefaces (font families), changing, 130, 131–133
underlined text, 127–128
web fonts, 133–135
forms, 695–696
accepting text input, 702–703
accessing elements with JavaScript, 720–721
<button> element, 718–719
check boxes, 706–708
creating, 696–702
displaying data, 721–722
events, 719–720
form-processing scripts, 703
grouping elements, 705
input controls, 706–712
labeling data, 703–704
naming data, 703
<output> element, 719
pull-down pick lists, 710–712
radio buttons, 708–709
selection lists, 710–712
submitting data, 718
text fields/text areas, 713–715
validating, 716–717
frameworks, 745–746
history of, 2
if.else statements (JavaScript), testing multiple conditions, 600–601
images
creating HTML for image maps, 230–233
place in web pages, 214–217
italic text, 126–127
JavaScript and, 81–83, 488–490
linking
to anchor locations, 171–174
CSS styles, 182–186
to email addresses, 179–180
to external web content, 178–179
identifying anchor locations within web pages, 170–171
naming links, 181–182
to non-HTML files, 177–178
opening links in new browser windows, 180–181
between web content, 174–177
within web pages, 170
lists, 139–140
definition HTML lists, 139, 141, 290
nested HTML lists, 142–146, 290
ordered HTML lists, 139, 290
unordered HTML lists, 139, 290
monospaced text, 127, 128–129
<output> element, web forms, 719
pseudo-classes, 182
rounding elements, 371–373
RWD, writing for, 435
basic attributes, 437–438
tags every web page should contain, 435–436
validating HTML, 438–439
web content tags, 436–437
semantic HTML, 432, 433
simple web pages, building, 25–26
creating a basic page, 27–28
formatting text, 33
headings, 33–36
HTML tags, 26–30
line breaks, 31–33
paragraph breaks, 31–33
preparing for, 24–25
saving HTML files, 33
semantic elements, 36–42
text editors, choosing, 25
viewing a basic page, 27–28
skeleton pages/templates, 30
subdirectories, 168
subscript text, 127
superscript text, 127
tables
accessibility, 158
body, wrapping, 151
borders, collapsing, 149–150
borders, creating, 149
cells, aligning data, 154–156
cells, background colors, 157
cells, background images, 157
cells, boldface text, 157
cells, creating, 147
cells, spanning, 156
cells, styling, 147
creating, 147–151
footers, wrapping, 151
headers, wrapping, 151
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
rows, wrapping, 151
sizing, 151–153
unnecessary redesigns, 158
tags, 26–28, 30
<a> HTML tags, 170–174, 436
anchor HTML tags, 170–174
<article> HTML tags, 37, 40, 45, 437
<aside> HTML tags, 37, 47–48, 437
attributes, 136, 168
<audio> HTML tags, 436
<b> HTML tags, 122, 126
<body> HTML tags, 29, 30, 436
<br> HTML tags, 32, 129, 436
<closing> HTML tags, 29
<dd> HTML tags, 139, 141, 290
<div> HTML tags, 136, 137, 436
<dl> HTML tags, 139, 141, 290
<dt> HTML tags, 139, 141, 290
<em> HTML tags, 126, 127, 436
<empty> HTML tags, 29
<font> HTML tags, 122, 129
<footer> HTML tags, 37, 39, 48–49, 437
<h1> HTML tags, 436
<h2> HTML tags, 436
<h3> HTML tags, 436
<head> HTML tags, 30, 435, 489, 514
<header> HTML tags, 37, 39–43, 437
<html> HTML tags, 30, 435
<i> HTML tags, 122, 126
<img> HTML tag, 215–218, 221, 222, 436, 462–463
</i> HTML tags, 139, 290
<link> HTML tag, 473
<map> HTML tag, 233
<meta charset> HTML tags, 436
<meta> HTML tags, 29
<nav> HTML tags, 37, 40, 45–46, 437
<ol> HTML tags, 30, 32, 136, 436, 490
<pre> HTML tags, 127, 128–129
<pseudo-classes, 182
required tags, 28–30
<script> HTML tags, 489, 490, 513–514
<section> HTML tags, 37, 39, 44–45, 177, 437
<span> HTML tags, 436
<strong> HTML tags, 126, 127, 436
</strong> HTML tags, 127
</sup> HTML tags, 127
</sub> HTML tags, 127
</sup> HTML tags, 127
</sub> HTML tags, 151
</td> HTML tags, 147, 149, 150, 154
</tfoot> HTML tags, 151
</th> HTML tags, 147, 149, 150, 154
</thead> HTML tags, 151
</title> HTML tags, 29, 30, 436
</tr> HTML tags, 147, 149, 149, 154
</u> HTML tags, 127–128
</ul> HTML tags, 139, 290
</video> HTML tags, 237, 238–239, 436
viewing in other web pages, 36
templates/skeleton pages, 30

text, changing color, 199–201
underlined text, 127–128

valid HTML, 432
validating, 97–98, 106–107, 109–111, 438–439
well-formed HTML, 432–433

<html> HTML tags, 435

HTML5, applications, developing, 501

HTML5 Boilerplate framework, 745

hyperlinks

absolute links, 169
anchor HTML tags
identifying anchor locations within web pages, 170–171
linking between web content, 174–177
linking to anchor locations, 171–174
linking to email addresses, 179–180
linking to external web content, 178–179
linking to non-HTML files, 177–178
linking within web pages, 170
color, changing, 195

CSS styles, 182–186
effective use of, 186–187
images, turning into links, 223–226
images and, 187
multimedia files, 233–234, 235–236
naming, 181–182
opening in new browser windows, 180–181
styling, 182–187
tappable links, mobile interfaces and RWD, 449

hypertext, defined, 2

<i> HTML tags, 122, 126

ID attributes, HTML tags, 136

if statements (JavaScript), 595–596
conditional expressions, 596–597
conditional operators, 597
else keyword, 598–599
logical (Boolean) operators, 597–598

if.else statements (JavaScript), testing multiple conditions, 600

HTML file, 600–601

JavaScript file, 601–602

image maps, list items and CSS box model, creating with, 296–299

images
aligning in web pages, 219
horizontal image alignment, 219–221
vertical image alignment, 221–223
backgrounds (web pages)
multiple images in, 355–359
positioning images in, 361
backgrounds (web pages), adding to, 226–227

borders (web pages), 373
clipping border images, 373–375
defining image width, 375–376
extending border images beyond border edge, 376
fitting images to borders, 376–377
canvas, adding to, 417–419
changing based on user interaction, 498–500
color

reducing/removing, 211–212
tweaking, 207–208
copyrights, 202
cropping, 204–206
flexible-width images, responsive images, mobile design, 461–462

GIF format, 211–212

graphics software
Adobe Photoshop, 201
choosing, 201–202
GIMP 201–202
height/width, specifying, 218–219
hyperlinks and, 187
image maps, 227–228
creating, 230

HTML for image maps, creating, 230–233
mapping regions within images, 229–230
needs for, 228–229
JPEG compression, 209
links, turning images into, 223–226
multiple images in backgrounds (web pages), 355–359
ownership of, 202
placement in web pages, 214–217
PNG format, 211–212
positioning in web page backgrounds, 361
resizing, 206–207
resolution, 202–203
responsive images, mobile design, 461

flexible-width images, 461–462
<picture> element, 463
sizes attribute, 462–463
<img> HTML tags, 215–218, 221, 222, 436
<picture> element, 463
<srcset> element, 462–463
sizes attribute, 462–463
accessibility, 664
AJAX (remote scripting), 690
Angular frameworks, 690
arrays, 551
accessing elements of, 565
length of, 565
numeric arrays, 564–565
numeric arrays, sorting, 567–569
string arrays, 565–566
string arrays, sorting, 567
string arrays, splitting, 566–567
Backbone.js frameworks, 690
best practices, 516–517, 655–656
accessibility, 664
avoiding browser specificity, 661
avoiding errors, 673
browser quirks, 671
content/presentation/behavior, separating, 657
cross-browser scripting, 669–672
design patterns, 664–665
dishonest browsers, 668–669
displaying browser information, 667–668
documenting code, 662–663
error handling, 662
event handlers, 659–661
feature sensing, 670
graceful degradation, theory of, 658
non-JavaScript-enabled browsers, 671–672
<noscript> tags, 672
optionality, 672–673
overusing JavaScript, 656–657
progressive enhancement, 658–659
reading browser information, 666–667
reusing code, 665–666

JavaScript, 80, 488, 507, 523
abstraction, 665
accessibility, 664
AJAX (remote scripting), 690
Angular frameworks, 690
arrays, 551
accessing elements of, 565

<srcset> element,
462–463
using different images, 463–464
rights management, 202
screen resolution, 210
text descriptions, 217–218
tiled background images, creating, 212–214
web pages, adding to, 203–204
backgrounds, 226–227
cropping images, 204–206
GIF format, 211–212
height/width, specifying, 218–219
JPEG compression, 209
placement in web pages, 214–217
PNG format, 211–212
reducing/removing color, 211–212
resizing images, 206–207
text descriptions, 217–218
tiled background images, creating, 212–214
turning images into links, 223–226
tweaking color, 207–208
web pages, aligning in, 219
horizontal image alignment, 219–221
vertical image alignment, 221–223

<code class="language-html">&lt;img&gt; HTML tags, 215–218, 221, 222, 436
&lt;picture&gt; element, 463
&lt;source&gt; attribute, 462–463
indenter, 742–743
index pages, file management, 16–17

JavaScript, 80
abstraction, 665
accessibility, 664
AJAX (remote scripting), 690
Angular frameworks, 690
arrays, 551
accessing elements of, 565
length of, 565
numeric arrays, 564–565
numeric arrays, sorting, 567–569
string arrays, 565–566
string arrays, sorting, 567
string arrays, splitting, 566–567
Backbone.js frameworks, 690
best practices, 516–517, 655–656
accessibility, 664
avoiding browser specificity, 661
avoiding errors, 673
browser quirks, 671
content/presentation/behavior, separating, 657
cross-browser scripting, 669–672
design patterns, 664–665
dishonest browsers, 668–669
displaying browser information, 667–668
documenting code, 662–663
error handling, 662
event handlers, 659–661
feature sensing, 670
graceful degradation, theory of, 658
non-JavaScript-enabled browsers, 671–672
<noscript> tags, 672
optionality, 672–673
overusing JavaScript, 656–657
progressive enhancement, 658–659
reading browser information, 666–667
reusing code, 665–666

<code class="language-html">&lt;img&gt; HTML tags, 215–218, 221, 222, 436
&lt;picture&gt; element, 463
&lt;source&gt; attribute, 462–463
indenter, 742–743
index pages, file management, 16–17

JavaScript, 80
abstraction, 665
accessibility, 664
AJAX (remote scripting), 690
Angular frameworks, 690
arrays, 551
accessing elements of, 565
length of, 565
numeric arrays, 564–565
numeric arrays, sorting, 567–569
string arrays, 565–566
string arrays, sorting, 567
string arrays, splitting, 566–567
Backbone.js frameworks, 690
best practices, 516–517, 655–656
accessibility, 664
avoiding browser specificity, 661
avoiding errors, 673
browser quirks, 671
content/presentation/behavior, separating, 657
cross-browser scripting, 669–672
design patterns, 664–665
dishonest browsers, 668–669
displaying browser information, 667–668
documenting code, 662–663
error handling, 662
event handlers, 659–661
feature sensing, 670
graceful degradation, theory of, 658
non-JavaScript-enabled browsers, 671–672
<noscript> tags, 672
optionality, 672–673
overusing JavaScript, 656–657
progressive enhancement, 658–659
reading browser information, 666–667
reusing code, 665–666

<code class="language-html">&lt;img&gt; HTML tags, 215–218, 221, 222, 436
&lt;picture&gt; element, 463
&lt;source&gt; attribute, 462–463
indenter, 742–743
index pages, file management, 16–17

JavaScript, 80
abstraction, 665
accessibility, 664
AJAX (remote scripting), 690
Angular frameworks, 690
arrays, 551
accessing elements of, 565
length of, 565
numeric arrays, 564–565
numeric arrays, sorting, 567–569
string arrays, 565–566
string arrays, sorting, 567
string arrays, splitting, 566–567
Backbone.js frameworks, 690
best practices, 516–517, 655–656
accessibility, 664
avoiding browser specificity, 661
avoiding errors, 673
browser quirks, 671
content/presentation/behavior, separating, 657
cross-browser scripting, 669–672
design patterns, 664–665
dishonest browsers, 668–669
displaying browser information, 667–668
documenting code, 662–663
error handling, 662
event handlers, 659–661
feature sensing, 670
graceful degradation, theory of, 658
non-JavaScript-enabled browsers, 671–672
<noscript> tags, 672
optionality, 672–673
overusing JavaScript, 656–657
progressive enhancement, 658–659
reading browser information, 666–667
reusing code, 665–666
conditional operators, 597
if statements, 595–596
shorthand conditional expressions, 599–600
form validation, 85
frameworks, 690, 691
AJAX (remote scripting), 690
Angular frameworks, 690
Backbone.js frameworks, 690
Ember frameworks, 691
Knockout frameworks, 691
MVC pattern, 689–690
React frameworks, 691
functions, 570
calling, 508–509, 571–573
combining with tasks, 508–509
defining, 570–571
naming, 515
returning values, 573–575
graceful degradation, theory of, 658
hiding scripts, 490
history of, 81
HTML and, 81–83, 488–490
HTML5 applications, developing, 501
if statements, 595–596
conditional expressions, 596–597
conditional operators, 597
else keyword, 598–599
logical (Boolean) operators, 597–598
if..else statements, testing multiple conditions, 600
HTML file, 600–601
JavaScript file, 601–602
jQuery, 683–684
JSON, 517–518
Knockout frameworks, 691
libraries (third-party), 681–683
Dojo, 686
effects, adding, 686–689
jQuery, 683–684
MooTools, 686
Prototype, 685, 687
script.aculo.us, 685–686, 687, 687–689
loading speeds (web pages), 657–658
logical (Boolean) operators, 557, 597–598
loops, 511
continuing, 609
do..while loops, 607
escaping, 608–609
infinite loops, 608
looping through object properties, 609–612
for loops, 604–606
while loops, 606–607
methods, text fields/text areas, 713
modifying scripts, 89–91
MooTools, 686
<noscript> tags, 672
null data types, 557
number data types, 556
objects, 510, 515, 575
built-in objects, 510, 582–583
creating, 575
custom objects, 510
Date object, 587–588
Date object, converting date formats, 590
Date object, creating, 588
Date object, reading date values, 588–589
Date object, setting date values, 588
Date object, time zones, 589
defining, 577–578
DOM objects, 510
event object, 621–623
instances, creating, 579–581
looping through object properties, 609–612
Math object, 583
Math object, generating random numbers, 584
Math object, methods, 584–587
Math object, rounding, 584
Math object, truncating, 584
methods, 576, 578–579
properties, 576
simplifying scripting, 577
operators, 555
common operators, 555–556
precedence, 556
output, creating, 87–88
overusing, 656–657
prioritizing scripts, 513–514
progressive enhancement, 434, 658–659
Prototype, 685, 687
React frameworks, 691
remote scripting (AJAX), 85
reserved words, 515
reusing code, 665–666
RWD, writing for, validating JavaScript, 438–439
script.aculo.us, 685–686, 687, 687–689
semicolons (;), 516
shorthand conditional expressions, 599–600
spacing, 515
special effects, 85
statements, 507–508
strings, 551, 557
  assigning values to, 559–560
  calculating length of, 560–561
  converting case of, 561
  getting single characters, 563
  splitting, 566–567
  string objects, 558
  string objects, creating, 559
  using parts of strings, 562–563
substrings, 562
  finding, 563–564
  getting single characters, 563
  using parts of strings, 562–563
switch statements, multiple conditions and, 602–604
syntax rules, 514–515
tasks, combining with functions, 508–509
testing scripts, 89
third-party libraries, 681–683
  Dojo, 686
  effects, adding, 686–689
  jQuery, 683–684
  MooTools, 686
  Prototype, 685, 687
script.aculo.us, 685–686, 687, 687–689
time, displaying, 85–91
timeouts, 645–647
transitions, triggering, 397–398
unobtrusive scripting, 496–498, 674–677
usability, 663–664
validating, 438–439
variables, 509, 515, 551, 552
  assigning values to, 554–555
  global variables, 553
  local variables, 553
  naming, 552
  semicolons (;) and, 63
  storing data in, 86–87
W3C event model, 659–660
web forms
  accessing elements, 720–721
  events, 719–720
web pages
  adding scripts to, 88
  fitting in, 81–83
  loading speeds, 657–658
  modifying text, 543–545
  website navigation, 84–85
window objects (DOM), 638–639
  properties of, 639
  windows, creating, 640
  windows, displaying dialog boxes, 648–649
  windows, moving, 643–645
  windows, opening/closing, 640–643
  windows, timeouts, 645–647
JPEG compression, 209
jQuery, 683–684
JSON (JavaScript Object Notation), 517–518
K-L
keyboard events, event handlers (JavaScript) and, 627–630
keyframes (animations), defining, 402–404
Knockout frameworks (JavaScript), 691
Koch, Peter-Paul, 671
labeling web form data, 703–704
large screen-specific styles, adding with media queries, 482–483
large web pages
  navigating, 735–738
  organizing, 735–738
layers (positionable elements), DOM, 536, 537–541
layout properties (CSS), 63–65
layouts
  CSS display: table; property, 335–338
  CSS Flexible Box Layout module, 339–345
  CSS Grid Layout module, 345–348
  examples of, 318
  fixed layouts, 319–322
  fixed/liquid hybrid web page layouts
    creating, 324–326
    defining two columns in, 326–328
    height, setting, 329–335
    width, setting, 328–329
  liquid layouts, 322–324
  mobile devices and, 319, 449
  progressive enhancement, 318
  responsive layouts, mobile design, 464–466
  separating structure from design/interactivity, 318
<li> HTML tags, 139, 290
libraries (third-party), JavaScript, 681–683
  Dojo, 686
  effects, adding, 686–689
  jQuery, 683–684
  MooTools, 686
  Prototype, 685, 687
script.aculo.us, 685–686, 687, 687–689
line breaks, simple web pages, building with HTML, 31–33
linear gradients, backgrounds (web pages), 366
lines/polygons, drawing on canvas, 414–416
link objects (DOM), 527–528
<link> HTML tag, 473
links
absolute links, 169
anchor HTML tags
identifying anchor locations within web pages, 170–171
linking between web content, 174–177
linking to anchor locations, 171–174
linking to email addresses, 179–180
linking to external web content, 178–179
linking to non-HTML files, 177–178
linking within web pages, 170
color, changing, 195
CSS styles, 182–186
effective use of, 186–187
images, turning into links, 223–226
images and, 187
multimedia files, 233–234, 235–236
naming, 181–182
opening in new browser windows, 180–181
styling, 182–187
tappable links, mobile interfaces and RWD, 449
liquid web page layouts, 322–324
liquid/fixed hybrid web page layouts
creating, 324–326
defining two columns in, 326–328
height, setting, 329–335
width, setting, 328–329
lists
comma-separated lists, media query expressions, 476
pull-down pick lists, web forms, 710–712
selection lists, web forms, 710–712
lists (CSS box model), 290–291
background colors, 296
creating, 291–292
image maps, creating, 296–299
margins, 293–294, 295
navigation lists, 299–300
horizontal navigation, creating, 310–314
multilevel navigation, styling, 305–310
primary navigation, 300
single-level navigation, styling, 303–305
vertical navigation, creating, 300–303
padding, 292–294, 295
placing list item indicators, 294–296
styling, 291–292
lists (HTML), 139–140
definition HTML lists, 139, 141, 290
nested HTML lists, 142–146
nested lists, 290
ordered HTML lists, 139, 290
unordered HTML lists, 139, 290
load/unload events, event handlers (JavaScript) and, 630–631
loading speeds (web pages), 657–658
local sites, creating, 17–18
local variables (JavaScript), 553
location objects (DOM), 530–531
logical (Boolean) operators (JavaScript), 557, 597–598
loops (JavaScript), 511
continuing, 609
do.while loops, 607
escaping, 606–609
for loops, 604–606
infinite loops, 608
looping through object properties, 609–612
for loops, 604–606
while loops, 606–607
maintainable code, writing, 740
documenting code with comments, 740–742
indenting for clarity, 742–743
version control, 743–745
managing files, 14–17
<map> HTML tag, 233
maps
image maps, 227–228
creating, 230
HTML for image maps, creating, 230–233
mapping regions within images, 229–230
needs for, 228–229
web pages, adding to, backgrounds, 226–227
margins
adding to web page elements, 249–257
lists, styling with CSS box model, 293–294
Math object (JavaScript)
generating random numbers, 584
methods, 584–587
rounding, 584
truncating, 584
**media queries**
adding, 473
baseline styles, defining, 479–480
breakpoints
  best practices, 483
defined, 471, 477
defining with media queries, 477–479
large screen-specific styles, adding with media queries, 482–483
optimal breakpoints, 483
defined, 471–472
defining media type styles, 473–474
expressions, 476–477
handheld media type, 472
large screen-specific styles, adding, 482–483
media features, 474–476
print media type, 472–473
requesting multiple CSS documents, 474
retina devices and, 484
screen media type, 472
small screen-specific styles, adding, 481–482
types of, 472–474
**<meta> HTML tags**, 29
**<meta charset> HTML tags**, 436
methods (JavaScript), text fields/text areas, 713
MLB section (ESPN.com), 733–734
**mobile devices**
  adaptive design. See also RWD, dynamic serving, 467
dynamic serving. See also RWD, adaptive design, 467–468
RWD
  alternatives to, 466
difficulties with, 466–467
importance of RWD in design, 430–431, 443–444
Mobile First design, 451–454
mobile interfaces, 445–451
Mobile Only design, 454–455
responsive images, 461–464
responsive layouts, 464–466
responsive tables, 455–461
separate URL/domains, 468
tables, 158
web page layouts, 319
website optimization, 739–740
Mobile First design, 451–454
**mobile interfaces, RWD, 445**
design elements, fitting, 447
download speeds, 449–450
fixed-width designs, 446–447
Flash and, 445
font sizes, legibility, 447–449
layouts, simplifying, 449
mobile web pages, testing, 450–451
navigation, simplifying, 449
tappable links, 449
viewports, configuring, 445–446
Mobile Only design, 454–455
**mobile web pages, testing, mobile interfaces and RWD, 450–451**
modifying, JavaScript scripts, 89–91
monochromatic color schemes, 194
monospaced text, 127, 128–129
MooTools, 686
mouse events, event handlers (JavaScript) and, 623
click events, 623–627
click events, changing appearance of <div> elements, 631–638
mouseover/mouseout, 623
ups/downs, 623–627
moving
  browser windows, 643–645
elements (2D transformations), 386–388
multilevel vertical navigation, styling, 305–310
multimedia files
  best practices, 242
  creating, 234
defined, 192
  embedding into web pages, 237
  linking to, 233–234, 235–236
  playing audio in web pages, 240–241
  playing video in web pages, 237–239
QuickTime, support for, 236
multiple 2D transformations, 391–392
multiple backgrounds in web pages, 355–359
multiple borders in web pages, 355
multiple images in backgrounds (web pages), 355–359
**MVC pattern, JavaScript frameworks**, 689–690
naming
  animations, 410
  HTML form data, 703
  links, 181–182
  variables (JavaScript), 552
  web form data, 703

<nav> HTML tags, 37, 40, 45–46, 437

navigating
  large web pages, 735–738
  website optimization, 739

navigation, mobile interfaces and RWD, 449

navigation lists, 299–300
  horizontal navigation, creating, 310–314
  primary navigation, 300
  vertical navigation
    creating, 300–303
    multilevel navigation, styling, 305–310
    single-level navigation, styling, 303–305

nested (child) tags, HTML, 142

nested lists, 142–146, 290

NFL section (ESPN.com), 733–734

nodes (DOM), 533, 534
  basic properties, 534
  document methods, 535
  methods, 535–536
  relationship properties, 534–535

non-JavaScript-enabled browsers, 671–672

<noscript> tags, 672

not operators, @media rules, 476–477

null data types (JavaScript), 557

number data types (JavaScript), 556

numeric arrays (JavaScript), 564–565, 567–569

objects (JavaScript), 510, 515, 575
  built-in objects, 510, 582–583
  creating, 576
  custom objects, 510
  Date object, 587–588
  defining, 577–578
  DOM objects, 510
  instances, creating, 579–581
  looping through object properties, 609–612

  Math object, 583
    generating random numbers, 584
    methods, 584–587
    rounding, 584
    truncating, 584
    methods, 576, 578–579
    properties, 576
    simplifying scripting, 577

<ol> HTML tags, 139, 290

<opening> HTML tags, 29

opening/closing, browser windows, 640–643

operators (JavaScript), 555
  common operators, 555–556
  precedence, 556

optionality of JavaScript, 672–673

ordered HTML lists, 139, 290

organizing web pages
  large web pages, 735–738
  simple web pages, 732–734

outline properties (CSS box model), 275

outlines (web pages), 378

<output> element, web forms, 719

overusing JavaScript, 656–657

<p> HTML tags, 30, 32, 136, 436, 490

padding
  adding to web page elements, 257–261
  lists, styling with CSS box model, 292–294, 295

page layouts
  CSS display: table; property, 335–338
  CSS Flexible Box Layout module, 339–345
  CSS Grid Layout module, 345–348
  examples of, 318
  fixed layouts, 319–322
  fixed/liquid hybrid web page layouts
    creating, 324–326
    defining two columns in, 326–328
  height, setting, 329–335
  width, setting, 328–329
  liquid layouts, 322–324
  mobile devices and, 319
  progressive enhancement, 318
  separating structure from design/interactivity, 318
  tables, 157–158

paragraph breaks, simple web pages, building with HTML, 31–33, 136

parents (DOM objects), 533

pausing animations, 410

Peet’s Coffee website, 736–737

photos
  backgrounds (web pages), adding to, height/width, specifying, 226–227
  screen resolution, 210
  web pages, adding to, 203–204
cropping images, 204–206
GIF format, 211–212
height/width, specifying, 218–219
JPEG compression, 209
placement in web pages, 214–217
PNG format, 211–212
reducing/removing color, 211–212
resizing images, 206–207
text descriptions, 217–218
tiled background images, creating, 212–214
turning images into links, 223–226
tweaking color, 207–208
web pages, aligning in, 219
horizontal image alignment, 219–221
vertical image alignment, 221–223
Photoshop (Adobe), 201
<picture> element, responsive images, mobile design, 463
plugins, defined, 234
PNG format (images), 211–212
polygons/lines, drawing on canvas, 414–416
pop-up windows, opening links in, 180–181
positioning elements
controlling positioning with JavaScript operators, 556
primary navigation (navigation lists), 300
print media type (media queries), 472–473
print style sheets, 472–473
prioritizing JavaScript scripts, 513–514
progressive enhancement
benefits of, 435
content adjusting the look of with CSS, 433–434
separating from presentation/functionality, 432
content layer, editing, 432–433
defined, 431
JavaScript interactivity, 434, 658–659
web page design, 318
Prototype, 685, 687
pseudo-classes, HTML tags, 182
publishing web content
blogs, 18
locally, 17–18
pull-down pick lists, web forms, 710–712
Q
QuickTime, support for, 236
quirks (browsers), 671
R
radial gradients, backgrounds (web pages), 367–371
radio buttons, web forms, 708–709
React frameworks (JavaScript), 691
rectangles/squares, drawing on canvas, 411
reducing/removing color in images, 211–212
relative addresses, 168, 169
relative positioning, 276
remote scripting (AJAX), 85
repeating, animations, 408–410
reserved words (JavaScript), reserved words, 515
resizing
cells in responsive tables, 456–457
images, 206–207
resolution
images, 202–203
screen, 210
responsive images, mobile design, 461
flexible-width images, 461–462
<picture> element, 463
sizes attribute, 462–463
<srcset> element, 462–463
using different images, 463–464
responsive layouts, mobile design, 464–466
responsive tables, mobile design, 455
hiding content, 460–461
rearranging rows/columns, 457–460
resizing cells, 456–457
retina devices and media queries, 484
reusing JavaScript code, 665–666
RGB color values, 195, 197–198
rights management, images and, 202
rotating elements
(2D transformations), 384
rows (tables)
  rearranging in responsive tables, 457–460
  wrapping, 151

RWD (Responsive Web Design).
See also adaptive design,
dynamic serving
  alternatives to, 466
  CSS, validating, 438–439
  defined, 427–428
  difficulties with, 466–467
  history of, 428
  HTML, validating, 438–439
  HTML, writing, 435
    basic attributes, 437–438
    tags every web page should contain, 435–436
    web content tags, 436–437
  JavaScript, validating, 438–439
  mobile design
    importance of RWD in design, 430–431, 443–444
    Mobile First design, 451–454
    mobile interfaces, 445–451
    Mobile Only design, 454–455
    responsive images, 461–464
    responsive layouts, 464–466
    responsive tables, 455–461
    separate URL/domains, 468
    mobile interfaces, 445
    design elements, fitting, 447
    download speeds, 449–450

fixed-width designs, 446–447
Flash and, 445
font sizes, legibility, 447–449
layouts, simplifying, 449
mobile web pages, testing, 450–451
navigation, simplifying, 449
tappable links, 449
viewports, configuring, 445–446
need for, 429–430

S

Safari, Debugger, 114–118
sample text, formatting example, 122
saving, HTML files, 33
scaling elements (2D transformations), 385–386
screen media type (media queries), 472
screen resolution, 210
<script> HTML tags, 489, 490, 513–514
script.aculo.us, 685–686, 687, 687–689
scripting, 80
  breakpoints and, 116–118
  client-side scripting, 488
  compiled scripting languages, 80
  hiding scripts, 490
  interpreted scripting languages, 80
JavaScript, 80
  adding scripts to, 88
  error handling, 91–92
  events, 84
  external scripts, 83–84
features of, 81
form validation, 85
history of, 81
HTML and, 81–83
modifying scripts, 89–91
objects, simplifying
output, creating, 87–88
remote scripting (AJAX), 85
special effects, 85
testing scripts, 89
time, displaying, 85–91
variables, semicolons (;) and, 63
variables, storing data in, 86–87
web pages, fitting in, 81–83
website navigation, 84–85
placement of scripts, 489
server-side scripting, 488
types of, 487–488
scrolling in backgrounds (web pages), 361–364
search engines, website optimization, 672, 738
  creating interesting content, 738–739
  mobile devices, 739–740
  navigation, 739
<section> HTML tags, 37, 39, 44–45, 177, 437
selection lists, web forms, 710–712
selectors (CSS), 60
semantic elements
  progressive enhancement, 433
  simple web pages, building with HTML, 36–42
semantic HTML, 432, 433
semicolons (;)
  CSS, 61, 63
small screen-specific styles, using parts of strings, 562–563
adding with media queries, 481–482
using parts of strings, 562–563
sort numeric arrays (JavaScript), 567–569
string arrays (JavaScript), 567
Sources panel (Developer Tools), 114–118
spacing in JavaScript, 515
subset HTML tags, 436
spanning, cells (tables), 156
special characters, formatting. See also character entities, 122–125
splitting, string arrays (JavaScript), 566–567
squares/rectangles, drawing on canvas, 411
srcset element, responsive images, mobile design, 462–463
Stephenson, Sam, 685
sticky web pages, 729
storing web content
absolute links, 169
attributes, HTML tags, 168
directories, 168
relative addresses, 168, 169
relative-root addresses, 168
string arrays (JavaScript), 566–566
splitting, 566–567
strings (JavaScript), 551, 557
assigning values to, 559–560
calculating length of, 560–561
converting case of, 561
getting single characters, 563
splitting, 566–567
string objects, 558, 559
splitting, 562
finding, 563–564
getting single characters, 563
using parts of strings, 562–563
<sup> HTML tags, 127
subscript text, 127
substrings (JavaScript), 562
finding, 563–564
single characters, getting, 563
using parts of strings, 562–563
<sub> HTML tags, 127
superscript text, 127
switch statements (JavaScript), multiple conditions and, 602–604
T

tables
accessibility, 158
body, wrapping, 151
borders
collapsing, 149–150
creating, 149
spacing, 157
cells
aligning data, 154–156
background colors, 157
background images, 157
boldface text, 157
creating, 147
spanning, 156
styling, 147
creating, 147–151
CSS display: table; property, 335–338
footers, wrapping, 151
headers, wrapping, 151
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
responsive tables, mobile design, 455
hiding content, 460–461
rearranging rows/columns, 457–460
resizing cells, 456–457
rows, wrapping, 151
sizing, 151–153
unnecessary redesigns, 158
tables of contents, 730–731
tappable links, mobile interfaces and RWD, 449
<tbody> HTML tags, 151
<td> HTML tags, 147, 149, 149, 150, 154
Technology Review, 431
templates/skeleton pages, HTML, 30
testing
browsers, 8–10
JavaScript scripts, 89
web content, 18–19
text
aligning, 136
attributes, HTML tags, 136
block-level elements, 136–139
boldface text, 126, 127, 157
case sensitivity
color names, 195
JavaScript, 514
web servers, 170
color entities, 123–125
color, changing, 199–201
CSS
aligning text, 67
font properties, 67–68
indenting text, 67
DOM
adding to web pages, 545–546
modifying text in web pages, 543–545
flowing text in web pages, 284–285
fonts
changing color, 129, 130, 131–133
mobile interfaces and RWD, 447–449
sizing, 129, 130, 131–133
formatting
aligning text, 136–139
attributes, HTML tags, 136, 168
boldface text, 126, 127, 157
italic text, 126–127
monospaced text, 127
paragraph breaks, 136
sample text, 122
simple web pages, building with HTML, 33
special characters, 122–125
subscript text, 127
superscript text, 127
underlined text, 127–128
highlighted text, 438
HTML forms, accepting text input, 702–703
images, describing with text, 217–218
italic text, 126–127
links, changing color, 195
monospaced text, 127, 128–129
sample text, formatting example, 122
special characters, formatting, 122–125
subscript text, 127
superscript text, 127
typefaces (font families), changing, 130, 131–133
underlined text, 127–128
web forms
accepting text input, 702–703
text fields/text areas, 713–715
web pages
adding text, 545–546
modifying text, 543–545
writing in documents, document objects (DOM), 527
text editors, choosing, 9–10
<tbody> HTML tags, 151
<th> HTML tags, 147, 149, 150, 154
theory of graceful degradation, 428, 431
third-party JavaScript libraries, 681–683
Dojo, 686
effects, adding, 686–689
jQuery, 683–684
MooTools, 686
Prototype, 685, 687
script.aculo.us, 685–686, 687, 687–689
tiled background images, creating, 212–214
time, displaying with JavaScript, 85–91
time zones, Date object (JavaScript) and, 589
timeouts, 645–647
timing
  animations, 405–408
  transitions, 396–397
<title> HTML tags, 29, 30, 436
<tr> HTML tags, 147, 149, 149, 154
transformations
  2D transformations, 383
    moving elements, 386–388
    multiple transformations, 391–392
    rotating elements, 384
    scaling elements, 385–386
    slanting (skewing) elements, 388–391
  3D transformations, 392–393
transitions, 393–396
  timing, 396–397
  triggering with JavaScript, 397–398
triadic color schemes, 194
triangles, drawing on canvas, 415–416
typefaces (font families), changing, 130, 131–133

ups/downs, mouse events, 623–627
URL, mobile devices and RWD, 468
usability, JavaScript, 663–664
UTF-8 web pages, formatting text, special characters, 123

valid HTML, 432
validating
  CSS, 438–439
  forms, JavaScript and, 85
  HTML, 438–439
  JavaScript, 438–439
  style sheets, 73
  web content
    CSS, 99, 109–111
    HTML, 97–98, 106–107, 109–111
  web forms, 716–717
variables (JavaScript), 509, 515, 551, 552
  assigning values to, 554–555
  global variables, 553
  local variables, 553
  naming, 552
  semicolons (;) and, 63
  storing data in, 86–87
version control
  Google Docs, 743
  maintainable code, writing, 743–745
vertical image alignment, 221–223
vertical navigation in navigation lists, creating, 300–303
video, 242
  codecs, 237
  hosting services, 242

playing in web pages, 237–239
<video> element, 237–239
<video> HTML tags, 237, 238–239, 436
viewports, mobile interfaces and RWD, 445–446
visual editors, 18

W
W3C event model, 659–660
web content
  absolute links, 169
  color, choosing (best practices), 192–194
  columns (CSS), 158–162
  creating, 2–3
  defined, 3
  delivery, 3–6
  displaying, 141
  interesting web content, creating, 738–739
  linking
    to anchor locations, 171–174
    CSS styles, 182–186
    to email addresses, 179–180
    to external web content, 178–179
    identifying anchor locations within web pages, 170–171
    naming links, 181–182
    to non-HTML files, 177–178
    opening links in new browser windows, 180–181
    between web content, 174–177
within web pages, 170
publishing
blogs, 18
locally, 17–18
relative addresses, 168, 169
relative-root addresses, 168
storing
absolute links, 169
attributes, HTML tags, 168
directories, 168
relative addresses, 168, 169
relative-root addresses, 168
tables
accessibility, 158
body, wrapping, 151
borders, collapsing, 149–150
borders, creating, 149
borders, spacing, 157
cells, aligning data, 154–156
cells, background colors, 157
cells, background images, 157
cells, boldface text, 157
cells, creating, 147
cells, spanning, 156
cells, styling, 147
creating, 147–151
footers, wrapping, 151
headers, wrapping, 151
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
rows, wrapping, 151
sizing, 151–153
unnecessary redesigns, 158
testing, 18–19
validating
CSS, 99, 109–111
HTML, 97–98, 106–107, 109–111
web fonts, 133–135
web forms, 695–696
accepting text input, 702–703
accessing elements with JavaScript, 720–721
<button> element, 718–719
check boxes, 706–708
creating, 696–702
displaying data, 721–722
events, 719–720
form-processing scripts, 703
grouping elements, 705
hidden data, 705
input controls, 706–712
labeling data, 703–704
naming data, 703
<output> element, 719
pull-down pick lists, 710–712
radio buttons, 708–709
selection lists, 710–712
submitting data, 718
text fields/text areas, 713–715
validating, 716–717
web hosting providers, selecting, 6–8
web pages
absolute links, 169
aligning elements in, 261–262
aligning text, 136
attributes, HTML tags, 136
block-level elements, 136–139
audio, playing in web pages, 240–241
backgrounds, 353–354
adding images to, 226–227
alternating colors, 364–365
color, 194
color, changing with CSS, 199–201
color, hexadecimal color codes, 194, 195, 196–197
color, RGB color values, 194, 197–198
gradients, 365
generators, linear gradients, 366
generators, radial gradients, 367–371
multiple backgrounds, 355–359
placing, 359–360
positioning images in, 361
scrolling, 361–364
sizing, 360
tiled background images, creating, 212–214
banners, creating, 210–211
borders, 354
color, changing with CSS, 199–201
images, 373
images, clipping, 373–375
images, defining width of, 375–376
images, extending width of, 376
images, fitting to borders, 376–377
multiple borders, 355
buttons, creating, 210–211
centering web page elements, 262–263
color, choosing (best practices), 192–194
columns (CSS), 158–162
DOM
adding text, 545–546
modifying text, 543–545
float property and web page elements, 263–266
flowing text in layout, 284–285
highlighted text, 438
HTML tags, viewing, 36
images, adding to, 203–204
cropping images, 204–206
GIF format, 211–212
height/width, specifying, 218–219
JPEG compression, 209
placement in web pages, 214–217
PNG format, 211–212
reducing/removing color, 211–212
resizing images, 206–207
text descriptions, 217–218
tiled background images, creating, 212–214
turning images into links, 223–226
tweaking color, 207–208
images, aligning in, 219
horizontal image alignment, 219–221
vertical image alignment, 221–223
JavaScript
adding scripts to web pages, 88
fitting in web pages, 81–83
large web pages
navigating, 735–738
organizing, 735–738
layouts
CSS display: table; property, 335–338
CSS Flexible Box Layout module, 339–342
CSS Flexible Box Layout module, modifying flex items, 342–345
CSS Grid Layout module, 345–348
examples of, 318
fixed layouts, 319–322
fixed/liquid hybrid web page layouts, creating, 324–326
fixed/liquid hybrid web page layouts, defining two columns in, 326–328
fixed/liquid hybrid web page layouts, setting height, 329–335
fixed/liquid hybrid web page layouts, setting width, 328–329
liquid layouts, 322–324
mobile devices and, 319
progressive enhancement, 318
separating structure from design/interactivity, 318
linking
to anchor locations, 171–174
CSS styles, 182–186
to email addresses, 179–180
to external web content, 178–179
identifying anchor locations within web pages, 170–171
naming links, 181–182
to non-HTML files, 177–178
opening links in new browser windows, 180–181
between web content, 174–177
within web pages, 170
loading speeds, 657–658
margins, adding to elements, 249–257
mobile web pages, testing, mobile interfaces and RWD, 450–451
multimedia files
embedding into web pages, 237
playing audio in web pages, 240–241
playing video in web pages, 237–239
ordering elements in layout, 281–284
outlines, 378
padding, adding to elements, 257–261
paragraph breaks, 136
positioning elements in layout, 277–281
absolute positioning, 276–277
flowing text, 284–285
ordering elements, 281–284
relative positioning, 276
relative addresses, 168, 169
relative-root addresses, 168
rounding HTML elements, 371–373
simple web pages, organizing, 732–734
simple web pages, building with HTML, 25–26
creating a basic page, 27–28
formatting text, 33
headings, 33–36
HTML tags, 26–28
HTML tags, required tags, 28–30
text, color, changing with CSS, 199–201
UTF-8 web pages, special characters, formatting, 122–125
video, playing in web pages, 237–239
web servers, case sensitivity (text)
and, 170
websites
adaptive design. See also RWD, dynamic serving, 467
bad website examples, 195
color, choosing (best practices), 192–194
dynamic serving. See also RWD, adaptive design, 467–468
dynamic websites
changing images based on user interaction, 498–500
HTML5 applications, developing, 501
JavaScript in HTML, 488–490
scripting, client-side scripting, 488
scripting, displaying random content, 491–495
scripting, DOM, 495, 496–498
scripting, hiding scripts, 490
scripting, placement of scripts, 489
scripting, server-side scripting, 488
tables
accessibility, 158
body, wrapping, 151
borders, collapsing, 149–150
borders, creating, 149
borders, spacing, 157
cells, aligning data, 154–156
cells, background colors, 157
cells, background images, 157
cells, boldface text, 157
cells, creating, 147
cells, spanning, 156
cells, styling, 147
creating, 147–149
footers, wrapping, 151
headers, wrapping, 151
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
rows, wrapping, 151
sizing, 151–153
unnecessary redesigns, 158
tables of contents, 730–731
text, color, changing with CSS, 199–201
UTF-8 web pages, special characters, formatting, 122–125
video, playing in web pages, 237–239
web servers, case sensitivity (text)
and, 170
websites
adaptive design. See also RWD, dynamic serving, 467
bad website examples, 195
color, choosing (best practices), 192–194
dynamic serving. See also RWD, adaptive design, 467–468
dynamic websites
changing images based on user interaction, 498–500
HTML5 applications, developing, 501
JavaScript in HTML, 488–490
scripting, client-side scripting, 488
scripting, displaying random content, 491–495
scripting, DOM, 495, 496–498
scripting, hiding scripts, 490
scripting, placement of scripts, 489
scripting, server-side scripting, 488
tables
accessibility, 158
body, wrapping, 151
borders, collapsing, 149–150
borders, creating, 149
borders, spacing, 157
cells, aligning data, 154–156
cells, background colors, 157
cells, background images, 157
cells, boldface text, 157
cells, creating, 147
cells, spanning, 156
cells, styling, 147
creating, 147–149
footers, wrapping, 151
headers, wrapping, 151
laying out, 156, 157–158
mixing presentation/content, 158
mobile devices, 158
page layouts, 157–158
pre-planning, 156
rows, wrapping, 151
sizing, 151–153
unnecessary redesigns, 158
tables of contents, 730–731
writing HTML, web content
tags, 436–437
search engine optimization, 738
creating interesting
content, 738–739
mobile devices, 739–740
navigation, 739
single-page interfaces, 730–731
timeouts, 645–647
World Wide Web, history of, 2
writing HTML, web content
tags, 436–437
search engine optimization, 738
creating interesting
content, 738–739
mobile devices, 739–740
navigation, 739
single-page interfaces, 730–731
timeouts, 645–647
World Wide Web, history of, 2
writing HTML, web content
tags, 436–437
search engine optimization, 738
creating interesting
content, 738–739
mobile devices, 739–740
navigation, 739
single-page interfaces, 730–731
theory of graceful degradation, 428, 431
well-formed HTML, 432–433
well-formed XHTML, 433
while loops (JavaScript), 606–607
width/height,
fixed/liquid hybrid web page
layouts, setting in, 328–329
images, specifying in, 218–219
web page elements, changing
in, 272–274
window objects (DOM), 524, 531,
638–639
properties of, 639
windows
creating, 640
dialog boxes, displaying,
648–649
moving, 643–645
opening/closing, 640–643
timeouts, 645–647
windows
creating, 640
dialog boxes, displaying,
648–649
moving, 643–645
opening/closing, 640–643
timeouts, 645–647
World Wide Web, history of, 2
writing HTML, web content
tags, 436–437
search engine optimization, 738
creating interesting
content, 738–739
mobile devices, 739–740
navigation, 739
single-page interfaces, 730–731
theory of graceful degradation, 428, 431
well-formed HTML, 432–433
well-formed XHTML, 433
while loops (JavaScript), 606–607
width/height,
fixed/liquid hybrid web page
layouts, setting in, 328–329
images, specifying in, 218–219
web page elements, changing
in, 272–274
window objects (DOM), 524, 531,
638–639
properties of, 639
windows
creating, 640
dialog boxes, displaying,
648–649
moving, 643–645
opening/closing, 640–643
timeouts, 645–647
World Wide Web, history of, 2
writing HTML, web content
tags, 436–437
search engine optimization, 738
creating interesting
content, 738–739
mobile devices, 739–740
navigation, 739
single-page interfaces, 730–731
theory of graceful degradation, 428, 431
well-formed HTML, 432–433
well-formed XHTML, 433
while loops (JavaScript), 606–607
width/height,
fixed/liquid hybrid web page
layouts, setting in, 328–329
images, specifying in, 218–219
web page elements, changing
in, 272–274
window objects (DOM), 524, 531,
638–639
properties of, 639
windows
creating, 640
dialog boxes, displaying,
648–649
moving, 643–645
opening/closing, 640–643
timeouts, 645–647
World Wide Web, history of, 2
X-Y-Z
XHTML, well-formed XHTML, 433
YUI library, 661
z-index property, ordering
elements in layout, 281–284