JavaScript

Sasha Vodnik
Don Gosselin

sixth edition

CENGAGE brain .com

Buy. Rent. Access.

Access student data files and other study tools on **cengagebrain.com**.

For detailed instructions visit
http://solutions.cengage.com/ctdownloads/

Store your Data Files on a USB drive for maximum efficiency in organizing and working with the files.

Macintosh users should use a program to expand WinZip or PKZip archives. Ask your instructor or lab coordinator for assistance.

JAVASCRIPT

SIXTH EDITION

Sasha Vodnik

Don Gosselin

CENGAGE Learning®

Australia • Brazil • Mexico • Singapore • United Kingdom • United States

CENGAGE Learning®

JavaScript, Sixth Edition
Sasha Vodnik and Don Gosselin

Product Director: Kathleen McMahon
Senior Product Manager: Jim Gish
Senior Content Developer: Alyssa Pratt
Marketing Manager: Eric LaScola
Senior Content Project Manager: Catherine DiMassa
Developmental Editor: Ann Shaffer
Quality Assurance Testers: Danielle Shaw and Serge Palladino
Art Director: Jack Pendleton
Manufacturing Planner: Julio Esperas
IP Analyst: Sara Crane
IP Project Manager: Kathryn Kucharek
Compositor: Integra Software Services Pvt. Ltd.
Cover Image: © Galyna Andrushko/ Shutterstock.com

Library of Congress Control Number: 2014937483

ISBN-13: 978-1-305-07844-4

Cengage Learning
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at **www.cengage.com/global**.

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

To learn more about Cengage Learning, visit **www.cengage.com.**

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com**

Printed in the United States of America
Print Number: 01 Print Year: 2014

BRIEF CONTENTS

CONTENTS

PREFACE

JavaScript is a client-side scripting language that allows web page authors to develop interactive web pages and sites. Although JavaScript is considered a programming language, it is also a critical part of web page design and authoring. This is because the JavaScript language enables web developers to add functionality directly to a web page's elements. The language is relatively easy to learn, allowing non-programmers to quickly incorporate JavaScript functionality into a web page. In fact, because it is used extensively in the countless web pages that are available on the World Wide Web, JavaScript is arguably the most widely used programming language in the world.

JavaScript, Sixth Edition teaches web page development with JavaScript for students with little programming experience. Although it starts with an overview of the components of web page development, students using this book should have basic knowledge of web page creation, including familiarity with commonly used HTML elements and CSS properties. This book covers the basics of ECMAScript Edition 3, which is compatible with older versions of Internet Explorer, as well as some features of ECMAScript 5.1, which is supported by all modern browsers. This book also covers advanced topics including object-oriented programming, the Document Object Model (DOM), touch and mobile interfaces, and Ajax. The HTML documents in this book are written to HTML5 standards, with some XHTML-compatible element syntax. After completing this course, you will able to use JavaScript to build professional quality web applications.

The Approach

This book introduces a variety of techniques, focusing on what you need to know to start writing JavaScript programs. In each chapter, you perform tasks that let you use a particular technique to build JavaScript programs. The step-by-step tasks are guided activities that reinforce the skills you learn in the chapter and build on your learning experience by providing additional ways to apply your knowledge in new situations. In addition to

step-by-step tasks, each chapter includes objectives, short quizzes, a summary, key terms with definitions, review questions, and reinforcement exercises that highlight major concepts and let you practice the techniques you've learned.

Overview of This Book

The examples and exercises in this book will help you achieve the following objectives:

- Use JavaScript with HTML elements
- Work with JavaScript variables and data types and learn how to use the operations that can be performed on them
- Add functions and control flow within your JavaScript programs
- Trace and resolve errors in JavaScript programs
- Write JavaScript code that controls the web browser through the browser object model
- Use JavaScript to make sure data was entered properly into form fields and to perform other types of preprocessing before form data is sent to a server
- Create JavaScript applications that use object-oriented programming techniques
- Manipulate data in strings and arrays
- Save state information using hidden form fields, query strings, cookies, and Web Storage
- Incorporate touchscreen support and mobile capabilities in web applications
- Dynamically update web applications with Ajax
- Build a web application using the jQuery library

JavaScript, Sixth Edition presents twelve chapters that cover specific aspects of JavaScript programming. **Chapter 1** discusses basic concepts of the World Wide Web, reviews HTML documents, and covers the basics of how to add JavaScript to web pages. How to write basic JavaScript code, including how to use variables, data types, expressions, operators, and events, is also discussed in Chapter 1. This early introduction of key JavaScript concepts gives you a framework for better understanding more advanced concepts and techniques later in this book, and allows you to work on more comprehensive projects from the start. **Chapter 2** covers functions, data types, and how to build expressions. **Chapter 3** explains how to store data in arrays and how to use structured logic in control structures and statements. **Chapter 4** provides a thorough discussion of debugging techniques, including how to use the browser consoles integrated into all modern browsers. **Chapter 5** teaches how to use JavaScript to manipulate the web browser using the `Window`, `History`, `Location`, `Navigator`, and `Screen` objects. **Chapter 6** explains how to use JavaScript to make sure data was entered properly into form fields and how to perform other types of preprocessing before form data is sent to a server. **Chapter 7** presents object-oriented programming concepts,

including how to use JavaScript's built-in `Date`, `Number`, and `Math` classes. **Chapter 8** covers advanced techniques for manipulating data in strings, arrays, and JSON. **Chapter 9** explains how to save state information using hidden form fields, query strings, cookies, and Web Storage, and also briefly discusses JavaScript security issues. **Chapter 10** covers supporting touch and pointer events in a web application, as well as using data provided by mobile device hardware and optimizing a web app for mobile users. **Chapter 11** introduces the basics of how to use Ajax to dynamically update portions of a web page with server-side data. **Chapter 12** introduces using the jQuery library to simplify common programming tasks in JavaScript. **Appendix A** provides detailed instructions on installing the XAMPP web server on a local machine. **Appendix B** gives a brief refresher on the basics of HTML, XHTML, and CSS. **Appendix C** serves as a one-stop reference for JavaScript syntax and usage covered throughout the book. **Appendix D**, which is online, lists answers for all Short Quizzes.

What's New in This Edition?

The sixth edition includes the following important new features:

- New, professionally-designed chapter projects, including mobile layouts in most chapters.
- All new Hands-On Projects in all chapters, along with new individual and group Case Projects.
- New boxed elements in each chapter: Best Practices box highlights a guideline for real world implementation of the topic at hand; Programming Concepts box explains a principle underlying the subject of the chapter; and Skills at Work box provides guidance for navigating the world of work.
- Multicolor code samples in each chapter identifying language components visually, with numbered lines for longer code blocks.
- Full-color figures showing the state of the project after each modification.
- Non-mobile projects coded for IE8 compatibility.
- Debugging coverage moved to Chapter 4, providing you with skills for finding and correcting errors in your apps before moving past the introductory chapters.
- Updated coverage of current industry best practices for creating arrays and objects, writing equality operators, and listening for events.
- Use of `document.write()`, `window.alert()`, and similar methods limited to earliest chapters, and replaced with modern techniques in remainder of book.
- New Chapter 10 on developing for touchscreen and mobile devices.
- New Chapter 12 on introductory programming with jQuery.
- Appendix A on installing a web server simplified to use the free, open-source XAMPP GUI installer rather than the command line.

- Appendix B on HTML and CSS updated to cover HTML5 instead of XHTML, and expanded to cover CSS selectors.
- New, streamlined layout that makes locating information easier.

Features

Each chapter in *JavaScript, Sixth Edition* includes the following features:

- **Chapter Objectives:** Each chapter begins with a list of the important concepts presented in the chapter. This list provides you with a quick reference to the contents of the chapter as well as a useful study aid.
- **Figures and Tables:** Plentiful full-color screenshots allow you to check your screen after each change. Tables consolidate important material for easy reference.
- **Code Examples:** Numerous code examples throughout each chapter are presented in any easy-to-read font, with key words shown in color. Longer code blocks include line numbers for easy reference.
- **New Terms:** New terms are printed in boldface to draw your attention to new material.

Note *These elements provide additional helpful information on specific techniques and concepts.*

Caution *These cautionary notes flag steps you need to perform with care to avoid potential pitfalls.*

- **Skills at Work:** These notes provide guidance for navigating the world of work.
- **Best Practices:** These notes highlight guidelines for real world implementation of various topics.
- **Programming Concepts:** These notes explain principles underlying the subject of each chapter or section.
- **Short Quiz:** Several short quizzes are included in each chapter. These quizzes, consisting of two to five questions, help ensure you understand the major points introduced in the chapter.
- **Summary:** These brief overviews revisit the ideas covered in each chapter, providing you with a helpful study guide.
- **Key Terms:** These lists compile all new terms introduced in the chapter along with their definitions, creating a convenient reference covering a chapter's important concepts.

- **Review Questions:** At the end of each chapter, a set of twenty review questions reinforces the main ideas introduced in the chapter. These questions help you determine whether you have mastered the concepts presented in the chapter.
- **Hands-On Projects:** Although it is important to understand the concepts behind every technology, no amount of theory can improve on real-world experience. To this end, each chapter includes detailed Hands-On Projects that provide you with practice implementing technology skills in real-world situations.
- **Case Projects:** These end-of-chapter projects are designed to help you apply what you have learned to open-ended situations, both individually and as a member of a team. They give you the opportunity to independently synthesize and evaluate information, examine potential solutions, and make decisions about the best way to solve a problem.

Instructor Resources

The following supplemental materials are available when this book is used in a classroom setting. All of the instructor resources available with this book are available on the Instructor Companion Site at *sso.cengage.com*. An instructor account is required.

Instructor's Manual. The Instructor's Manual that accompanies this textbook includes additional instructional material to assist in class preparation, including items such as Sample Syllabi, Chapter Outlines, Technical Notes, Lecture Notes, Quick Quizzes, Teaching Tips, Discussion Topics, and Additional Case Projects.

Cengage Learning Testing Powered by Cognero is a flexible, online system that allows you to:

- Author, edit, and manage test bank content from multiple Cengage Learning solutions.
- Create multiple test versions in an instant.
- Deliver tests from your LMS, your classroom, or wherever you want.

PowerPoint® Presentations. This book comes with Microsoft PowerPoint slides for each chapter. These are included as a teaching aid for classroom presentation, to make available to students on the network for chapter review, or to be printed for classroom distribution. Instructors can add their own slides for additional topics they introduce to the class.

Data Files. Files that contain all of the data necessary for the Hands-on Projects and Case Projects are also provided to students on *CengageBrain.com*.

Solution Files. Solutions to end-of-chapter questions and projects are available for this text.

Read This Before You Begin

The following information will help you prepare to use this textbook.

Data Files

To complete the steps, exercises, and projects in this book, you will need data files that have been created specifically for this book. The data files are available at *CengageBrain.com*. Note that you can use a computer in your school lab or your own computer to complete the steps, exercises, and projects in this book.

Using Your Own Computer

You can use a computer in your school lab or your own computer to complete the chapters. To use your own computer, you will need the following:

- **A modern web browser**, such as the current version of Chrome, Internet Explorer, Firefox, or Safari. If possible, you should have access to all of the most popular modern browsers (Chrome, Internet Explorer, and Firefox), as well as to Internet Explorer 8 for testing backward-compatible code.
- **A code-based HTML editor**, such as Aptana Studio 3, Komodo Edit, Notepad++, TextWrangler, Adobe Dreamweaver, or Sublime Text.
- **A web server** (for Chapter 11) such as Apache HTTP Server or Microsoft Internet Information Services and PHP. Appendix A contains detailed instructions on how to install a web server and PHP.

To The Instructor

To complete all the exercises and chapters in this book, your students must work with a set of user files called data files. The data files are available on the Instructor Companion Site and on *CengageBrain.com*.

Cengage Learning Data Files License

You are granted a license to copy the data files to any computer or computer network used by individuals who have purchased this book.

Feedback and Questions

We welcome feedback and questions about this book from instructors and students. You can contact author Sasha Vodnik on Twitter at *@sashavodnik*.

Acknowledgements

Creating the Sixth Edition of JavaScript has truly been a team effort. Thanks to the many people who helped shape and strengthen what I've written. Ann Shaffer provided great suggestions and great questions in equal numbers, and helped me keep an eye on the big picture while tending to the details. Alyssa Pratt kept us focused on getting this content out the door, while giving us the support and resources to make it all top-notch. Jim Gish connected me with both this project and an amazing group of people who helped make it all happen. Cathie DiMassa at Cengage, and Ramanan Sundararajan, along with the team at Integra, enhanced simple words on a page with an engaging visual layout, and labored over the tiniest details. Danielle Shaw and Serge Palladino read all the text, tested all the steps, and provided invaluable feedback. Kenji Oshima took the list of imaginary businesses and institutions for the chapter projects and created a unique, professional logo for each one.

Many, many thanks to the reviewers who provided feedback on early drafts of these chapters, and whose suggestions made the content clearer, made the examples more relevant, and helped me create better teaching tools for instructors and better learning materials for students: Jason Fleetwood-Boldt; Mark Murtha, Metropolitan Community College; Nicky Newell, Northeast Mississippi Community College; and Kevin Parker, Idaho State University.

Finally, thanks to my husband, Jason Bucy, for his love and support.

CHAPTER 1

INTRODUCTION TO JAVASCRIPT

When you complete this chapter, you will be able to:

- Explain the history of the World Wide Web
- Describe the differences between client-side and server-side scripting
- Understand the components of a JavaScript statement
- Add basic JavaScript code to your web pages
- Structure your JavaScript programs

The original purpose of the World Wide Web (WWW) was to locate and display information. However, once the web grew beyond a small academic and scientific community, people began to recognize that greater interactivity would make the web more useful. As commercial applications of the web grew, the demand for more interactive and visually appealing websites also grew.

To respond to the demand for greater interactivity, an entirely new web programming language was needed. Netscape filled this need in the mid-1990s by developing the JavaScript programming language. Originally designed for use in the Navigator web browser, JavaScript is now supported by all of the most popular web browsers, including Internet Explorer, Firefox, Chrome, Safari, and Opera.

Although JavaScript is considered a programming language, it is also a critical part of web page design and authoring. This is because the JavaScript language enables web developers to add functionality directly to a web page's elements. JavaScript can turn static documents into applications such as games or calculators. JavaScript code can change the contents of a web page after a browser has rendered it. It can also create visual effects such as animation, and it can control the web browser window itself. None of this was possible before the creation of JavaScript.

In this chapter, you will learn the skills required to create basic JavaScript programs. To be successful in your JavaScript studies, you should already possess a strong knowledge of techniques for authoring web pages. The first part of this chapter provides a quick refresher on the history of the World Wide Web and the basics of how to create web pages. Even if you are highly experienced with HTML, you might not be familiar with the formal terminology that is used in web page authoring. For this reason, be certain to read through these sections to ensure that you understand the terminology used in this book.

Introduction to the World Wide Web

The Internet is a vast network that connects computers all over the world. The original plans for the Internet grew out of a series of memos written by J. C. R. Licklider of the Massachusetts Institute of Technology (MIT), in August 1962, discussing his concept of a "Galactic Network." Licklider envisioned a global computer network through which users could access data and programs from any site on the network. The Internet was actually developed in the 1960s by the Advanced Research Projects Agency (or ARPA) of the U.S. Department of Defense, which later changed its name to Defense Advanced Research Projects Agency (or DARPA). The goal of the early Internet was to connect the main computer systems of various universities and research institutions that were funded by this agency. This first implementation of the Internet was referred to as the ARPANET. More computers were connected to the ARPANET in the years following its initial development in the 1960s, although access to the ARPANET was still restricted by the U.S. government primarily to academic researchers, scientists, and the military.

The 1980s saw the widespread development of local area networks (LANs) and the personal computer. Although at one time restricted to academia and the military, computers and networks soon became common in business and everyday life. By the end of the 1980s, businesses and individual computer users began to recognize the global communications capabilities and potential of the Internet, and they convinced the U.S. government to allow commercial access to the Internet.

In 1990 and 1991, Tim Berners-Lee created what would become the **World Wide Web**, or the **web**, at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland, as a way

to easily access cross-referenced documents stored on the CERN computer network. When other academics and scientists saw the usefulness of Berners-Lee's system, the web as we know it today was born. In fact, this method of accessing cross-referenced documents, known as **hypertext linking**, was, in the early years, one of the most important aspects of the web because it allowed users to open other web pages quickly. A **hypertext link**, or **hyperlink** or **link**, contains a reference to a specific web page that you can click to open that web page.

A common misconception is that the words "web" and "Internet" are synonymous. The web is only one *part* of the Internet and is a means of communicating on the Internet. The Internet is also composed of other communication elements such as email systems that send and receive messages. However, because of its enormous influence on computing, communications, and the economy, the World Wide Web is arguably the most important part of the Internet today and is the primary focus of this book.

A document on the web is called a **web page** and is identified by a unique address called the **Uniform Resource Locator**, or **URL**. A URL is also commonly referred to as a **web address**. A URL is a type of **Uniform Resource Identifier (URI)**, which is a generic term for many types of names and addresses on the World Wide Web. The term **website** refers to the location on the Internet of a set of web pages and related files (such as graphic and video files) that belong to a company, organization, or individual. You display a web page on the screen of a computer, tablet, or phone by using a program called a **web browser**. A person can retrieve and open a web page in a web browser either by entering a URL in the web browser's Address box or by clicking a link. No matter which method is used, the user's web browser asks a web server for the web page in what is referred to as a **request**. A **web server** is a computer that delivers web pages. What the web server returns to the user is called the **response**.

Note

Many apps installed on smartphones and tablets are technically web documents that access web servers. They generally use the default web browsers on the devices where they're installed to connect with their servers, without requiring users to enter URLs.

Understanding Web Browsers

NCSA Mosaic was created in 1993 at the University of Illinois and was the first program to allow users to navigate the web by using a graphical user interface (GUI). In 1994, Netscape released Navigator, which soon controlled 75 percent of the market. Netscape maintained its control of the browser market until 1996, when Microsoft entered the market with the release of Internet Explorer, and the so-called browser wars began, in which Microsoft and Netscape fought for control of the browser market.

The browser wars began over dynamic HTML (DHTML), which is a combination of various technologies, including HTML and JavaScript, that allows a web page to change after it has been loaded by a browser. Examples of DHTML include the ability to reposition text and elements, change document background color, and create effects such as animation. Earlier versions of Internet Explorer and Navigator supported DHTML elements in ways that were exclusive to each browser, meaning that the DHTML code for Internet Explorer was incompatible with Navigator, and vice versa. Furthermore, Microsoft and Netscape each wanted its version of DHTML to become the industry standard.

To settle the argument, the World Wide Web Consortium set out to create a platform-independent and browser-neutral version of DHTML. The **World Wide Web Consortium**, or **W3C**, was established in 1994 at MIT to oversee the development of web technology standards. While the W3C was drafting a recommendation for DHTML, Internet Explorer version 4 and Navigator version 4 each added a number of proprietary DHTML elements that were completely incompatible with the other browser. As a result, when working with advanced DHTML techniques such as animation, a programmer had to write a different set of HTML code for each browser. Unfortunately for Netscape, the W3C adopted as the formal standard the version of DHTML found in version 4 of Internet Explorer, which prompted many loyal Netscape users to defect to Microsoft.

Note

The W3C does not actually release a version of a particular technology. Instead, it issues a formal recommendation for a technology, which essentially means that the technology is (or will be) a recognized industry standard.

One benefit of the browser wars was that they forced the web industry to rapidly develop and adopt advanced web page standards (including JavaScript, CSS, and DHTML) that are consistent across browsers. In 2004, Internet Explorer appeared to have essentially won the browser wars, as it controlled 95 percent of the browser market. However, Microsoft did not fully support web standards in subsequent versions of Internet Explorer, creating an opening for browsers that were more fully compliant with current standards. In the past decade, Internet Explorer has lost significant market share on desktop computers to Mozilla Firefox and Google Chrome. Other browsers—including Apple Safari and Opera—have also captured slices of the desktop browser market.

Note

Several companies collect and publish statistics on web browser usage. Each company uses a slightly different methodology, resulting in different conclusions. You can examine each company's findings by searching on web browser usage statistics *with a search engine.*

In the last few years, major technology companies have begun a new contest for browser market share on mobile devices. Apple Safari established a dominant position in this market as the default browser for both the iPhone and iPad. Google Chrome, the default browser on recent Android phones, also has significant market share, as does the mobile version of Opera. Unlike during the desktop browser wars of the 1990s, at the time of this writing Microsoft had yet to establish a significant market share in the mobile market.

Creating Web Pages

Originally, people created web pages using only Hypertext Markup Language. **Hypertext Markup Language**, or **HTML**, is a markup language used to create the web pages that appear on the World Wide Web. Web pages are also commonly referred to as **HTML pages** or **documents**. A **markup language** is a set of characters or symbols that defines a document's logical structure—that is, it indicates the meaning or function of each item in a document. HTML is based on an older language called **Standard Generalized Markup Language**, or **SGML**, which has a wider scope than HTML and can define the structure of documents in many different contexts.

Markup languages are designed to separate the data in a document from the way that data is formatted. Each element in an HTML document is marked according to its type, such as a paragraph or a heading. For a brief period in the early days of the web, the standards for HTML also incorporated specifications defining how elements should appear in a web browser—for example, the `b` element was used to format text in bold, and the `i` element to format text as italic. However, the people and organizations responsible for the growth of the web recognized the importance of keeping the appearance of documents independent from their structure, and restored HTML elements to their original purpose of identifying structure only. A separate, complementary language called **Cascading Style Sheets (CSS)** was developed for specifying the appearance of web page elements.

Note

This textbook uses the terms web pages *and* HTML documents *interchangeably.*

Basic HTML Syntax

An HTML document is a text document that contains codes, called **tags**, which specify how the data in the document is treated by a web browser. HTML tags can indicate a wide range of elements, from different types of text content—like headings and paragraphs—to controls that allow user input—such as option buttons and check boxes. Other HTML tags allow you to display graphic images and other objects in a document. Tags are enclosed in brackets (< >), and most consist of an opening tag and a closing tag that surround the text

or other items they format or control. The closing tag must include a forward slash (/) immediately after the opening bracket to define it as a closing tag. For example, to mark a line of text as a paragraph, you use the opening tag and the closing tag . When you open the HTML document in a web browser, the browser recognizes text enclosed in these tags as a paragraph.

A tag pair and any data it contains are referred to as an **element**. The information contained between an element's opening and closing tags is referred to as its **content**. Some elements do not require a closing tag. Elements that do not require a closing tag are called **empty elements** because they do not allow you to use a tag pair to enclose text or other elements. For instance, the br element, which inserts a line break on a web page, does not include a closing tag. You simply place the
 tag anywhere in an HTML document where you want a line break to appear.

Note | *HTML documents must have a file extension of .htm or .html.*

There are literally hundreds of HTML elements. Table 1-1 lists some commonly used elements.

HTML ELEMENT NAME	DESCRIPTION
article	Marks the main content of a web document
body	Marks the body of an HTML document
div	Marks a generic section of the web page body
head	Marks the page header and contains information about the entire page
h*n*	Marks heading level elements, where *n* represents a number from 1 to 6
html	Marks the content of an HTML document
img	Inserts an image file
nav	Marks navigation options, such as a navigation bar at the top or bottom of a page or along its side
p	Identifies the marked text as a paragraph

Table 1-1: Common HTML elements

All HTML documents must use the `html` element as the root element. A **root element** contains all the other elements in a document. This element tells a web browser to assemble any instructions between the tags into a web document. The opening `<html>` and closing `</html>` tags are required and contain all the text and other elements that make up the HTML document.

Two other important HTML elements are the `head` element and the `body` element. The `head` element contains information that is used by web browsers, and you place it at the beginning of an HTML document, after the opening `<html>` tag. You place several elements within the `head` element to help manage a document's content, including the `title` element, which contains text that appears in a browser's tab or title bar. A `head` element must contain a `title` element. With the exception of the `title` element, elements contained in the `head` element are not visible in the HTML document shown in the browser. The `head` element and the elements it contains are referred to as the **document head**.

Following the document head is the `body` element. The `body` element and the text and elements it contains are referred to as the **document body**.

HTML is not case sensitive, so you can use, for instance, `<P>` in place of `<p>`. However, an offshoot of HTML, a language called XHTML, is case sensitive, and requires the use of lowercase letters for tags. Because it can be useful to write code that works in both HTML and XHTML documents, this book uses lowercase letters for all tags. (You will learn more about XHTML shortly.)

You use various parameters, called **attributes**, to provide additional information about many HTML elements. You place an attribute before the closing bracket of the opening tag, and separate it from the tag name or other attributes with a space. You assign a value to an attribute using the syntax *`attribute`*`="`*`value`*`"`. For example, to add an image to an HTML document, you use the `img` element with a number of attributes. One of these, the `src` attribute, specifies the filename of an image file. To add the file logo.jpg to a web document, you would use the `src` attribute within the `img` element, as follows:

```
<img src="logo.jpg" />
```

When you open an HTML document in a web browser, the document is assembled and formatted according to the instructions contained in its elements. The process by which a web browser assembles and formats an HTML document is called **parsing** or **rendering**. The final document that appears in the web browser includes only recognized HTML elements and text. When a web browser renders an HTML document, it ignores nonprinting characters such as tabs and line breaks. Although you can use the Enter or Return key on your keyboard to add line breaks to text within the body of an HTML document to make the code more legible, browsers ignore these line breaks when parsing the document. In addition,

most web browsers ignore multiple contiguous spaces in a web document and replace them with a single space. The following code shows the document head and a portion of the document body for the web page shown in Figure 1-1.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Hotel Natoma - Reservations</title>
      <link type="text/css" rel="stylesheet" media="screen"
            href="natoma.css" />
      <link type="text/css" rel="stylesheet" media="screen"
            href="hnform.css" />
      <link rel="shortcut icon" href="favicon.ico" />
</head>
   <body>
      <div id="box">
         <h1>
            <img src="images/natoma.gif" width="368"
                 height="65" alt="Hotel Natoma" title="" />
         </h1>
         <nav>
            <ul id="mainnav">
               <li><a href="index.html">Home</a></li>
               <li><a href="nearby.html">
                      What&#39;s Nearby</a></li>
               <li><a href="http://bit.ly/bb3Sic"
                      target="_blank">Location</a></li>
               <li><a href="museums.html">SF Museums</a></li>
               <li><a href="greensf.html">Green SF</a></li>
               <li><a href="reserve.html">Reservations</a></li>
            </ul>
         </nav>
         <article>
            <h2 id="main">Reservations</h2>
...
```

Note

Throughout this book, each instance of four or more lines of HTML, CSS, or JavaScript code includes line numbers along the left side to make the code easier to read and discuss. These numbers are not part of the code, so take care not to type the line numbers when you type code into a text editor.

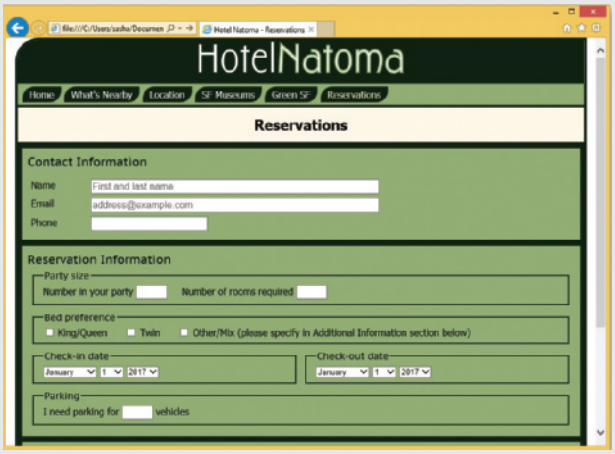

Figure 1-1: Web page in a browser

Note

The screen captures of web pages shown in this book were taken in current versions of Internet Explorer, Firefox, and Chrome running on the Windows 8 operating system. Current versions of all major browsers should render documents with very little variation across Windows, Mac, and Linux platforms. However, documents may look markedly different in older browser versions, especially Internet Explorer 8 and earlier. While older browsers can play an important role in the testing process, you should use the current version (or a recent version) of a major browser to ensure that your screen matches the figures in this book.

Creating an HTML Document

Because HTML documents are text files, you can create them in any text editor, such as Notepad or TextEdit, or any word-processing application capable of creating simple text files. If you use a text editor to create an HTML document, you cannot view the final result