AJAX, Rich Internet Applications, and Web Development for Programmers

Contains 180+ Examples

Web 2.0 • AJAX • Rich Internet Applications • XML • RSS
XHTML • CSS • JavaScript • DOM • JSON
Firefox • IE7 • Dojo • Script.aculo.us • Flash • CS3 • Flex
MySQL • JavaServer Faces • ASP.NET • PHP • Ruby on Rails
IIS • Apache • Web Services • Mashups • Open Source

PAUL J. DEITEL and HARVEY M. DEITEL
... the challenges are for the designers of these applications: to forget what we think we know about the limitations of the Web, and begin to imagine a wider, richer range of possibilities. It’s going to be fun.

—Jesse James Garrett, Adaptive Path
“Ajax: A New Approach to Web Applications”
(adaptivepath.com/ideas/essays/archives/000385.php)

Introduction
Welcome to Internet and web programming and Web 2.0! And welcome to a walkthrough of the Web 2.0 phenomenon from the technical, business and social perspectives. We’ve worked hard to create what we hope you’ll find to be an informative, entertaining and challenging learning experience. At Deitel & Associates, we write programming language professional books and textbooks for Prentice Hall, deliver corporate training worldwide and develop Web 2.0 Internet businesses. This book reflects today’s Web 2.0, Ajax-based, Rich Internet Application-development methodologies.

The technologies you’ll learn here are appropriate for experienced professionals who build substantial information systems. You’ll find “industrial-strength” code examples. We have attempted to write in a clear and straightforward manner using best practices.

Perhaps most important, the book presents hundreds of working code examples and shows the outputs produced when these examples are rendered in browsers or run on computers. We present all concepts in the context of complete working programs. We call this the “live-code approach.” All of the source code is available for download from

www.deitel.com/books/ajaxfp/

under the heading “Download Code Examples and Other Premium Content for Registered Users,” once you’ve registered and logged on to our site.

The early chapters present a carefully paced introduction to “client-side” web programming, using the popular JavaScript language and the closely related technologies of XHTML (Extensible HyperText Markup Language), CSS (Cascading Style Sheets) and the DOM (Document Object Model). The material in the JavaScript chapters presents a solid foundation for the deeper treatment of scripting in the Adobe Flash, Adobe Flex, PHP and Ruby on Rails chapters later in the book. Experienced programmers will read the early chapters quickly and find the treatment of scripting in the later chapters to be rigorous and challenging.

Today’s users are accustomed to desktop applications with rich graphical user interfaces (GUIs), such as those used on Apple’s Mac OS X systems, Microsoft Windows systems, various Linux systems and more. Users want applications that employ the
multimedia capabilities of graphics, images, animation, audio and video. They want applications that can run on the Internet and the web and communicate with other applications. Users want to apply database technologies for storing and manipulating their business and personal data. They want applications that are not limited to the desktop or even to some local computer network, but that can integrate Internet and web components, and remote databases. Programmers want to use all these capabilities in a truly portable manner so that applications will run without modification on a variety of platforms.

In this book, we present a number of powerful software technologies that will enable you to build these kinds of systems. Early in the book we focus on building the portions of web-based applications that reside on the client side (i.e., the portions of applications that typically run in web browsers such as Mozilla’s Firefox or Microsoft’s Internet Explorer), using technologies such as the XHTML, JavaScript, CSS, Flash, Flex and Extensible Markup Language (XML). Later in the book we concentrate on using technologies such as web servers, databases, PHP, Ruby on Rails, ASP.NET, ASP.NET Ajax and JavaServer Faces (JSF) to build the server side of web-based applications. These portions of applications typically run on “heavy-duty” computer systems on which organizations’ business-critical websites reside. By mastering the technologies in this book, you’ll be able to build substantial web-based, client/server, database-intensive, “multitier” applications.

If you have questions as you read this book, send an e-mail to deitel@deitel.com—we’ll respond promptly. For updates on the book and the status of all supporting software, and for the latest news on Deitel publications and services, visit www.deitel.com. Sign up at www.deitel.com/newsletter/subscribe.html for the free Deitel® Buzz Online e-mail newsletter and check out www.deitel.com/ResourceCenters.html for our growing list of Internet and web programming, Internet business, Web 2.0 and related Resource Centers. Each week we announce our latest Resource Centers in the newsletter.

Key Features
Here’s some of the key features of Ajax, Rich Internet Applications and Web Development for Programmers:

- Reflects today’s Web 2.0, Ajax-based, Rich Internet Application-development methodologies.
- Coverage of the two leading web browsers—Internet Explorer and Firefox. All client-side applications in the book run correctly on both browsers.
- Focus on Web 2.0 technologies and concepts.
- Chapter on Web 2.0 and Internet Business (reviewed by leaders in the Web 2.0 community).
- Focus on building Rich Internet Applications with the interactivity of desktop applications.
- Chapter on building Ajax-enabled web applications with “raw” Ajax and with the Dojo JavaScript libraries. Applications in this chapter demonstrate partial-page updates and type-ahead capabilities.
- Chapter on Adobe Flex—a Rich Internet Application framework for creating scalable, cross-platform, multimedia-rich applications for delivery within the enterprise or across the Internet.
• Chapter on rapid applications development of database-driven web applications with Ruby on Rails; also discusses developing Ajax applications with the Prototype and Script.aculo.us libraries.

• Two chapters on Adobe Flash CS3, including building a computer game.

• Significant treatment of client-side scripting with JavaScript.

• Significant treatments of XHTML DOM manipulation and JavaScript events.

• Significant treatment of XML DOM manipulation with JavaScript.

• Chapter on building SOAP-based web services with Java and REST-based web services with ASP.NET (using Visual Basic).

• Chapter on PHP 5.

• Coverage of ASP.NET, featuring ASP.NET Ajax.

• JavaServer Faces (JSF) coverage emphasizing building Ajax-enabled JSF applications.

• Client-side case studies that enable you to interact with preimplemented server-side applications and web services that we host at test.deitel.com.

• Case studies including Deitel Cover Viewer (JavaScript/DOM), Address Book (Ajax), Cannon Game (Flash), Weather/Yahoo! Maps Mashup (Flex), Mailing List (PHP/MySQL), Message Forum and Flickr Photo Viewer (Ruby on Rails), Guest Book and Secure Books Database (ASP.NET), Address Book with Google Maps (JavaServer Faces) and Blackjack (JAX-WS web services).

All of this has been carefully reviewed by a team of 38 distinguished industry developers and academics.

AJAX, Rich Internet Applications and Web Development for Programmers

This book focuses on Web 2.0 and Rich Internet Application (RIA) development. Our goal is to develop webtop applications that have the responsiveness, look and feel of traditional desktop applications. Deitel & Associates, Inc. has evolved into a development organization, while continuing its focus on programming languages textbook and professional book authoring, and corporate training. We’re building the infrastructure for the Internet businesses we’re designing and developing as part of our Web 2.0 Internet Business Initiative. This book includes discussions of many practical issues we’ve encountered in developing that infrastructure.

Figure 1 shows the architecture of AJAX, Rich Internet Applications and Web Development for Programmers. The book is divided into several parts. Chapter 1 introduces Web 2.0 from the technical, business and social perspectives, and provides a foundation for understanding Rich Internet Application development. If you are a serious web developer, you’ll want to test your web applications across many browsers and platforms. The examples for the book execute correctly on both Microsoft’s Internet Explorer 7 (IE7) and Mozilla’s Firefox 2 (FF2) browsers. Most of the examples will also work in other browsers such as Opera and Safari, but may not work on earlier browsers. Microsoft Windows users should upgrade to IE7 and install Firefox; readers with other operating systems should install Firefox.
The second part of the book, Chapters 2–13, presents a detailed treatment of Ajax component technologies, including a comprehensive treatment of JavaScript that spans nine chapters and concludes with Chapter 13’s treatment of Ajax development. Ajax is not a new technology—we’ve been writing about all but one of its component technologies since 1999, and many of the technologies existed before that. However, Ajax is one of the key technologies of Web 2.0 and RIAs. Chapters 2–13 cover “raw” Ajax programming, where you’ll handle the details yourself—several later chapters in the book demonstrate technologies that encapsulate Ajax functionality to help you easily build Ajax-based applications that operate across a wide variety of browsers and browser versions without your having to be concerned with the low-level details.

The third part of the book, Chapters 13–24, focuses on both the client and server sides of the GUI and the graphical part of RIA development. Here we cover client-side technologies such as Adobe Flash and Adobe Flex that use, or can be combined with, Ajax or Ajax-like capabilities to develop RIAs. Each of these technologies also can consume web services. Next, we present the server side of web application development with discussions of web servers (IIS and Apache), databases, server-side scripting languages such as PHP and Ruby on Rails, and several server-side frameworks such as ASP.NET 2.0 and JavaServer Faces. We complete our server-side discussion with a chapter on building both SOAP-based and REST-based web services.

You may have noticed that Chapter 13, Ajax-Enabled Rich Internet Applications, overlaps the second and third parts of the book. Chapter 13 serves as a bridge from “raw” Ajax development to “encapsulated” Ajax development with the Dojo libraries.

Fig. 1  |  Architecture of AJAX, Rich Internet Applications and Web Development for Programmers.
Dependency Chart

Figure 2 illustrates the dependencies that exist among chapters in the book. An arrow pointing into a chapter indicates that it depends on the content of the chapter from which the arrow points. For example, Chapter 24, Web Services, depends on both Chapters 21 and 23. We recommend that you read all of a given chapter’s dependencies before reading that chapter, though other orders are possible. We’ve also commented on some additional dependencies in the diagram’s footnotes.

Teaching Approach

AJAX, Rich Internet Applications and Web Development for Programmers contains a rich collection of examples. The book concentrates on the principles of good software engineering and stresses program clarity. We are educators who teach leading-edge topics in industry classrooms worldwide. Dr. Harvey M. Deitel has 20 years of college teaching experience and 18 years of industry teaching experience. Paul Deitel has 16 years of industry teaching experience. The Deitels have taught courses at all levels to government, industry, military and academic clients of Deitel & Associates.

Live-Code Approach. AJAX, Rich Internet Applications and Web Development for Programmers is loaded with “live-code” examples—each new concept is presented in the context of a complete working web application that is immediately followed by one or more screen captures showing the application’s functionality. This style exemplifies the way we teach and write about programming; we call this the “live-code approach.”

Syntax Shading. We syntax shade all the code, similar to the way most integrated-development environments and code editors syntax color code. This improves code readability—an important goal, given that this book contains about 18,000 lines of code in complete, working programs. Our syntax-shading conventions are as follows:

comments appear in italic
keywords appear in bold italic
PHP, Ruby, ASP.NET, JSP delimiters and errors appear in bold black
constants and literal values appear in bold gray
all other code appears in black

Code Highlighting. We place white rectangles around each program’s key code segments.

Using Fonts for Emphasis. We place the key terms and the index’s page reference for each defining occurrence in bold italic text for easier reference. We emphasize on-screen components in the bold Helvetica font (e.g., the File menu) and emphasize program text in the Lucida font (e.g., int x = 5).

Web Access. All of the source-code examples for AJAX, Rich Internet Applications and Web Development for Programmers are available for download from:

www.deitel.com/books/ajaxfp/

Site registration is quick, easy and free. Download all the examples, then run each program as you read the corresponding text discussions. Making changes to the examples and seeing the effects of those changes is a great way to enhance your Internet and web programming learning experience.
Rich Internet Applications

Part 1: Introduction
1. Dive Into® Web 2.0

2. Introduction to XHTML
3. Cascading Style Sheets (CSS)
4. JavaScript: Introduction to Scripting
5. JavaScript: Control Statements I
6. JavaScript: Control Statements II
7. JavaScript: Functions
8. JavaScript: Arrays
9. JavaScript: Objects
10. Document Object Model (DOM)
11. JavaScript: Events
12. XML and RSS
13. Ajax-Enabled RIAs

Part 2: Client-Side Rich Internet Applications and Ajax

14. Adobe Flash CS3
15. Flash CS3 Interactive Game
16. Adobe Flex

Part 3: Rich Internet Application Client Technologies

17. Web Servers
18. Database
19. PHP
20. Ruby on Rails
21. ASP.NET and ASP.NET Ajax
22. Java Server Faces (JSF)\(^1\)
23. Ajax-Enabled JSF Applications\(^1\)
24. Web Services\(^2\)

Part 4: Rich Internet Application Server Technologies


Fig. 2 | AJAX Rich Internet Applications and Web Development for Programmers chapter dependency chart.
Objectives. Each chapter begins with a statement of objectives. This lets you know what to expect and gives you an opportunity to determine if you have met the objectives after reading the chapter.

Quotations. The learning objectives are followed by quotations. Some are humorous; some are philosophical; others offer interesting insights. We hope that you enjoy relating the quotations to the chapter material.

Outline. The chapter outline helps you approach the material in a top-down fashion, so you can anticipate what is to come and set a comfortable learning pace.

Illustrations/Figures. Abundant charts, tables, line drawings, programs and program output are included.

Programming Tips. We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we have gleaned from a combined six decades of programming and teaching experience. One of our readers told us that she feels this approach is like the highlighting of axioms, theorems and corollaries in mathematics books—it provides a basis on which to build good software.

Good Programming Practices
Good Programming Practices call attention to techniques that will help you produce programs that are clearer, more understandable and more maintainable.

Common Programming Errors
Programmers tend to make certain kinds of errors frequently. Pointing out these Common Programming Errors reduces the likelihood that you’ll make the same mistakes.

Error-Prevention Tip
These tips contain suggestions for exposing bugs and removing them from your programs; many describe aspects of programming that prevent bugs from getting into programs in the first place.

Performance Tip
These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.

Portability Tip
We include Portability Tips to help you write code that will run on a variety of platforms and to explain how to achieve a high degree of portability.

Software Engineering Observation
The Software Engineering Observations highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.

Thousands of Index Entries. We have included an extensive index which is especially useful when you use the book as a reference.

“Double Indexing” of Live-Code Examples. For every source-code program in the book, we index the figure caption both alphabetically and as a subindex item under “Examples.” This makes it easier to find examples using particular features.
**Preface**

**AJAX, Rich Internet Applications and Web Development for Programmers Software Downloads and Additional Resources**

Many Internet and web development tools are available. We wrote *AJAX, Rich Internet Applications and Web Development for Programmers* using Internet Explorer 7, Firefox 2 and other free-for-download software. Links to additional resources and software downloads are available in our Internet and Web programming related Resource Centers:

www.deitel.com/resourcecenters.html/

and at the website for this book:

www.deitel.com/books/ajaxfp/

**Deitel® Buzz Online Free E-mail Newsletter**

Each week, the free *Deitel® Buzz Online* newsletter announces our latest Resource Center(s) and includes commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, errata, challenges, anecdotes, information on our corporate instructor-led training courses and more. It’s also a good way for you to keep posted about issues related to *AJAX, Rich Internet Applications and Web Development for Programmers*. To subscribe, visit

www.deitel.com/newsletter/subscribe.html

**The Deitel Online Resource Centers**

Our website, www.deitel.com, provides scores of Resource Centers on various topics including programming languages, software, Web 2.0, Internet business and open source projects (Fig. 3). The Resource Centers have evolved out of the research we do to support our books and business endeavors. We’ve found many exceptional resources including tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more. Most of them are free. In the spirit of Web 2.0, we share these resources with the worldwide community. Each week we announce our latest Resource Centers in the *Deitel® Buzz Online* (www.deitel.com/newsletter/subscribe.html).

**Acknowledgments**

It is a great pleasure to acknowledge the efforts of many people whose names may not appear on the cover, but whose hard work, cooperation, friendship and understanding were crucial to the production of the book. Many people at Deitel & Associates, Inc. devoted long hours to this project—thanks especially to Abbey Deitel and Barbara Deitel.

We’d also like to thank the participants in our Honors Internship program who contributed to this publication—Ilana Segall, a mathematical and computational science major at Stanford University; Jennifer Fredholm, an English and computer science major at New York University; Scott Wehrwein, a computer science major at Middlebury College; Mark Kagan, a computer science, economics and math major at Brandeis University; Andrew Faden, a computer engineering major at Northeastern University; Jessica Henkel, a psychology major and business minor at Northeastern University; and Kyle Banks, a computer science and business major at Northeastern University.
### Deitel Resource Centers

#### Programming
- ASP.NET 3.5
- Adobe Flex
- Ajax
- Apex
- ASP.NET Ajax
- ASP.NET
- C
- C++
- C++ Boost Libraries
- C++ Game Programming
- C#
- Code Search Engines and Code Sites
- Computer Game Programming
- CSS 2.1
- Dojo
- Facebook Developer Platform
- Flash 9
- Java
- Java Certification and Assessment Testing
- Java Design Patterns
- Java EE 5
- Java SE 6
- Java SE 7 (Dolphin) Resource Center
- JavaFX
- JavaScript
- JSON
- Microsoft LINQ
- Microsoft Popfly
- .NET
- .NET 3.0
- .NET 3.5
- OpenGL
- Perl
- PHP
- Programming Projects
- Python
- Regular Expressions
- Ruby
- Ruby on Rails
- Silverlight
- Visual Basic
- Visual C++
- Visual Studio Team System
- Web 3D Technologies
- Web Services
- Windows Presentation Foundation
- XHTML
- XML

#### Computer Science
- Regular Expressions

#### Games and Game Programming
- Computer Game Programming
- Computer Games
- Mobile Gaming
- Sudoku

#### Internet Business
- Affiliate Programs
- Competitive Analysis
- Facebook Social Ads
- Google AdSense
- Google Analytics
- Google Services
- Internet Advertising
- Internet Business Initiative
- Internet Public Relations
- Link Building
- Location-Based Services
- Online Lead Generation
- Podcasting
- Search Engine Optimization
- Selling Digital Content
- Sitemaps
- Web Analytics
- Website Monetization
- YouTube and AdSense
- Java
- Java Certification and Assessment Testing
- Java Design Patterns
- Java EE 5
- Java SE 6
- Java SE 7 (Dolphin) Resource Center
- JavaFX

#### Microsoft
- ASP.NET
- ASP.NET 3.5
- ASP.NET Ajax
- C#
- DotNetNuke (DNN)
- Internet Explorer 7 (IE7)
- Microsoft LINQ
- .NET
- .NET 3.0
- .NET 3.5
- SharePoint
- Silverlight
- Visual Basic
- Visual C++
- Visual Studio Team System
- Windows Presentation Foundation
- Windows Vista

#### Open Source & LAMP Stack
- Apache
- DotNetNuke (DNN)
- Eclipse
- Firefox
- Linux
- MySQL
- Open Source Perl
- PHP
- Python
- Ruby

#### Software
- Apache
- DotNetNuke (DNN)
- Eclipse
- Firefox
- Internet Explorer 7 (IE7)
- Linux
- MySQL
- Open Source Perl
- PHP
- Python
- Ruby

#### Other Topics
- Computer Games
- Computing Jobs
- Gadgets and Gizmos
- Ring Tones
- Sudoku

---

**Fig. 3** | Deitel Resource Centers at www.deitel.com/resourcecenters.html.
We are fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We appreciate the extraordinary efforts of Marcia Horton, Editorial Director of Prentice Hall’s Engineering and Computer Science Division, Mark Taub, Editor-in-Chief of Prentice Hall Professional, and John Fuller, Managing Editor of Prentice Hall Professional. Carole Snyder and Dolores Mars did a remarkable job recruiting the book’s large review team and managing the review process. Sandra Schroeder and Chuti Prasertsith did a wonderful job designing the book’s cover. Bob Engelhardt and Marta Samsel did a marvelous job managing the book’s production.

This book was adapted from our book *Internet & World Wide Web How to Program*, 4/e. We wish to acknowledge the efforts of our reviewers on that book. Adhering to a tight time schedule, they scrutinized the text and the programs, providing countless suggestions for improving the accuracy and completeness of the presentation.

**Reviewers**
Roland Bouman (MySQL AB), Peter Brandano (KoolConnect Technologies, Inc.), Matt Chotin (Adobe), Chris Cornutt (PHPDeveloper.org), Phil Costa (Adobe), Umachitta Damodaran (Sun Microsystems), Vadiraj Deshpande (Sun Microsystems), Justin Erenkrantz (The Apache Software Foundation), Christopher Finke (Netscape), Jesse James Garrett (Adaptive Path), Kevin Henrikson (Zimbra.com), Tim Heuer (Microsoft), Molly E. Holtzschlag (W3C), Ralph Hooper (University of Alabama, Tuscaloosa), Johnvey Hwang (Splunk, Inc.), Joe Kromer (New Perspective and the Pittsburgh Adobe Flash Users Group), Eric Lawrence (Microsoft), Billy B. L. Lim (Illinois State University), Shobana Mahadevan (Sun Microsystems), Patrick Mineault (Freelance Flash Programmer), Anand Narayanaswamy (Microsoft), John Peterson (Insync and V.I.O., Inc.), Jennifer Powers (University of Albany), Robin Schumacher (MySQL AB), José Antonio González Seco (Parlamento de Andalucia), Dr. George Semeczko (Royal & SunAlliance Insurance Canada), Steven Shaffer (Penn State University), Tim Sneath (Microsoft), Karen Tegtmeyer (Model Technologies, Inc.), Paul Vencill (MITRE), Raymond Wen (Microsoft), Eric M. Wendelin (Auto-trol Technology Corporation), Raymond F. Wisman (Indiana University) and Daniel Zappala (Brigham Young University).

Well, there you have it! We hope you enjoy this look at the exciting world of Ajax, Rich Internet Applications and web development in a Web 2.0 world. As you read the book, we would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

deitel@deitel.com

We'll respond promptly, and post corrections and clarifications at:

www.deitel.com/books/ajaxfp/

We hope you enjoy reading *Ajax, Rich Internet Applications and Web Development for Programmers* as much as we enjoyed writing it!

Paul J. Deitel
Dr. Harvey M. Deitel
Maynard, Massachusetts
About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT’s Sloan School of Management, where he studied Information Technology. He holds the Java Certified Programmer and Java Certified Developer certifications, and has been designated by Sun Microsystems as a Java Champion. Through Deitel & Associates, Inc., he has delivered Internet and web programming, Java, C, C++, C# and Visual Basic courses to industry clients, including IBM, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, Stratus, Cambridge Technology Partners, Open Environment Corporation, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys and many more. He has also lectured on Java and C++ for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world’s best-selling programming language textbook authors.

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 46 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees from MIT and a Ph.D. from Boston University. He has 20 years of college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. He and Paul are the co-authors of several dozen books and multimedia packages and they are writing many more. With translations published in Japanese, German, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish, the Deitels’ texts have earned international recognition. Dr. Deitel has delivered hundreds of professional seminars to major corporations, academic institutions, government organizations and the military.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in computer programming languages, Internet and web software technology, object technology education and Internet business development through its Web 2.0 Internet Business Initiative. The company provides instructor-led courses on major programming languages and platforms, such as C++, Java, C, C#, Visual C++, Visual Basic, XML, object technology and Internet and web programming. The founders of Deitel & Associates, Inc. are Paul J. Deitel and Dr. Harvey M. Deitel. The company’s clients include many of the world’s largest companies, government agencies, branches of the military, and academic institutions. Through its 32-year publishing partnership with Prentice Hall, Deitel & Associates, Inc. publishes leading-edge programming professional books, textbooks, interactive multimedia Cyber Classrooms, Complete Training Courses, web-based training courses, online and offline LiveLessons video courses, and e-content for the popular course management systems WebCT, Blackboard and Pearson’s CourseCompass. Deitel & Associates, Inc., and the authors can be reached via e-mail at:

deitel@deitel.com

To learn more about Deitel & Associates, Inc., its publications and its worldwide Dive Into® Series Corporate Training curriculum, visit:
www.deiteln.com

and subscribe to the free Deitel® Buzz Online e-mail newsletter at:

www.deiteln.com/newsletter/subscribe.html

Check out the growing list of online Resource Centers at:

www.deiteln.com/ResourceCenters.html

Individuals wishing to purchase Deitel publications can do so through:

www.deiteln.com/books/index.html

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

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

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

International Sales
international@pearsoned.com

10

Document Object Model (DOM): Objects and Collections

OBJECTIVES
In this chapter you will learn:

- How to use JavaScript and the W3C Document Object Model to create dynamic web pages.
- The concepts of DOM nodes and DOM trees.
- How to traverse, edit and modify elements in an XHTML document.
- How to change CSS styles dynamically.
- To create JavaScript animations.

Our children may learn about heroes of the past. Our task is to make ourselves architects of the future.
—Jomo Mzee Kenyatta

Though leaves are many, the root is one.
—William Butler Yeats

The thing that impresses me most about America is the way parents obey their children.
—Duke of Windsor

Most of us become parents long before we have stopped being children.
—Mignon McLaughlin

To write it, it took three months; to conceive it three minutes; to collect the data in it—all my life.
—F. Scott Fitzgerald

Sibling rivalry is inevitable. The only sure way to avoid it is to have one child.
—Nancy Samalin
10.1 Introduction

In this chapter we introduce the Document Object Model (DOM). The DOM gives you access to all the elements on a web page. Inside the browser, the whole web page—paragraphs, forms, tables, etc.—is represented in an object hierarchy. Using JavaScript, you can create, modify and remove elements in the page dynamically.

Previously, both Internet Explorer and Netscape had different versions of Dynamic HTML, which provided similar functionality to the DOM. However, while they provided many of the same capabilities, these two models were incompatible with each other. In an effort to encourage cross-browser websites, the W3C created the standardized Document Object Model. Firefox 2, Internet Explorer 7, and many other major browsers implement most of the features of the W3C DOM.

This chapter begins by formally introducing the concept of DOM nodes and DOM trees. We then discuss properties and methods of DOM nodes and cover additional methods of the document object. We also discuss how to dynamically change style properties, which enables you to create many types of effects, such as user-defined background colors and animations. Then, we present a diagram of the extensive object hierarchy, with explanations of the various objects and properties, and we provide links to websites with further information on the topic.

Software Engineering Observation 10.1

With the DOM, XHTML elements can be treated as objects, and many attributes of XHTML elements can be treated as properties of those objects. Then, objects can be scripted (through their id attributes) with JavaScript to achieve dynamic effects.

10.2 Modeling a Document: DOM Nodes and Trees

As we saw in previous chapters, the document’s `getElementById` method is the simplest way to access a specific element in a page. In this section and the next, we discuss more thoroughly the objects returned by this method.

The `getElementById` method returns objects called DOM nodes. Every element in an XHTML page is modeled in the web browser by a DOM node. All the nodes in a document make up the page’s DOM tree, which describes the relationships among elements. Nodes are related to each other through child-parent relationships. An XHTML element inside another element is said to be a child of the containing element. The containing element is known as the parent. A node may have multiple children, but only one parent. Nodes with the same parent node are referred to as siblings.
Some browsers have tools that allow you to see a visual representation of the DOM tree of a document. When installing Firefox, you can choose to install a tool called the DOM Inspector, which allows you to view the DOM tree of an XHTML document. To inspect a document, Firefox users can access the DOM Inspector from the Tools menu of Firefox. If the DOM inspector is not in the menu, run the Firefox installer and choose Custom in the Setup Type screen, making sure the DOM Inspector box is checked in the Optional Components window.

Microsoft provides a Developer Toolbar for Internet Explorer that allows you to inspect the DOM tree of a document. The toolbar can be downloaded from Microsoft at go.microsoft.com/fwlink/?LinkId=92716. Once the toolbar is installed, restart the browser, then click the ➕ icon at the right of the toolbar and choose IE Developer Toolbar from the menu. Figure 10.1 shows an XHTML document and its DOM tree displayed in Firefox’s DOM Inspector and in IE’s Web Developer Toolbar.

The XHTML document contains a few simple elements. We explain the example based on the Firefox DOM Inspector—the IE Toolbar displays the document with only minor differences. A node can be expanded and collapsed using the + and - buttons next to the node’s name. Figure 10.1(b) shows all the nodes in the document fully expanded. The document node (shown as #document) at the top of the tree is called the root node, because it has no parent. Below the document node, the HTML node is indented from the document node to signify that the HTML node is a child of the #document node. The HTML node represents the html element (lines 7–24).

The HEAD and BODY nodes are siblings, since they are both children of the HTML node. The HEAD contains two #comment nodes, representing lines 5–6. The TITLE node

```xml
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Fig. 10.1: domtree.html -->
<!-- Demonstration of a document's DOM tree. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
  <title>DOM Tree Demonstration</title>
</head>
<body>
<h1>An XHTML Page</h1>
<p>This page contains some basic XHTML elements. We use the Firefox DOM Inspector and the IE Developer Toolbar to view the DOM tree of the document, which contains a DOM node for every element in the document.</p>
<ul>
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
</ul>
</body>
</html>
```

Fig. 10.1 | Demonstration of a document’s DOM tree. (Part I of 3.)
10.2 Modeling a Document: DOM Nodes and Trees

Fig. 10.1 | Demonstration of a document’s DOM tree. (Part 2 of 3.)
has a child text node (#text) containing the text **DOM Tree Demonstration**, visible in the right pane of the DOM inspector when the text node is selected. The **BODY** node contains nodes representing each of the elements in the page. Note that the **LI** nodes are children of the **UL** node, since they are nested inside it.

Also, notice that, in addition to the text nodes representing the text inside the body, paragraphs and list elements, a number of other text nodes appear in the document. These text nodes contain nothing but white space. When Firefox parses an XHTML document into a DOM tree, the white space between sibling elements is interpreted as text and placed inside text nodes. Internet Explorer ignores white space and does not convert it into empty text nodes. If you run this example on your own computer, you will notice that the **BODY** node has a **#comment** child node not present above in both the Firefox and Internet Explorer DOM trees. This is a result of the copyright line at the end of the example file that you downloaded.

This section introduced the concept of DOM nodes and DOM trees. The next section considers DOM nodes in more detail, discussing methods and properties of DOM nodes that allow you to modify the DOM tree of a document using JavaScript.

### 10.3 Traversing and Modifying a DOM Tree

The DOM gives you access to the elements of a document, allowing you to modify the contents of a page dynamically using event-driven JavaScript. This section introduces
properties and methods of all DOM nodes that enable you to traverse the DOM tree, modify nodes and create or delete content dynamically.

Figure 10.2 shows some of the functionality of DOM nodes, as well as two additional methods of the document object. The program allows you to highlight, modify, insert and remove elements.

Lines 117–132 contain basic XHTML elements and content. Each element has an id attribute, which is also displayed at the beginning of the element in square brackets. For example, the id of the h1 element in lines 117–118 is set to bigheading, and the heading text begins with [bigheading]. This allows the user to see the id of each element in the page. The body also contains an h3 heading, several p elements, and an unordered list.

A div element (lines 133–162) contains the remainder of the XHTML body. Line 134 begins a form element, assigning the empty string to the required action attribute (because we’re not submitting to a server) and returning false to the onsubmit attribute. When a form’s onsubmit handler returns false, the navigation to the address specified in the action attribute is aborted. This allows us to modify the page using JavaScript event handlers without reloading the original, unmodified XHTML.

```html
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 10.2: dom.html -->
<!-- Basic DOM functionality. -->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Basic DOM Functionality</title>
<style type="text/css">

h1, h3 { text-align: center;
  font-family: tahoma, geneva, sans-serif }
p { margin-left: 5%;
  margin-right: 5%;
  font-family: arial, helvetica, sans-serif }
ul { margin-left: 10% }
a { text-decoration: none }
a:hover { text-decoration: underline }
.nav { width: 100%;
  border-top: 3px dashed blue;
  padding-top: 10px }
.highlighted { background-color: yellow }
.submit { width: 120px }
</style>
<script type="text/javascript">

 <!--
 var currentNode; // stores the currently highlighted node
 var idcount = 0; // used to assign a unique id to new elements

 // get and highlight an element by its id attribute
 function byId()
 {


```
var id = document.getElementById("gbi").value;
var target = document.getElementById(id);

if (target)
    switchTo(target);
} // end function byId

// insert a paragraph element before the current element
// using the insertBefore method
function insert()
{
    var newNode = createNewNode(
        document.getElementById("ins").value);
    currentNode.parentNode.insertBefore(newNode, currentNode);
} // end function insert

// append a paragraph node as the child of the current node
function appendNode()
{
    var newNode = createNewNode(
        document.getElementById("append").value);
    currentNode.appendChild(newNode);
} // end function appendNode

// replace the currently selected node with a paragraph node
function replaceCurrent()
{
    var newNode = createNewNode(
        document.getElementById("replace").value);
    currentNode.parentNode.replaceChild(newNode, currentNode);
} // end function replaceCurrent

// remove the current node
function remove()
{
    if (currentNode.parentNode == document.body)
        alert("Can't remove a top-level element.");
    else
    {
        var oldNode = currentNode;
        switchTo(oldNode.parentNode);
        currentNode.removeChild(oldNode);
    } // end else
} // end function remove

// get and highlight the parent of the current node
function parent()
{
    var target = currentNode.parentNode;
if ( target != document.body )
  switchTo( target );
else
  alert( "No parent." );
} // end function parent

// helper function that returns a new paragraph node containing
// a unique id and the given text
function createNewNode( text )
{
  var newNode = document.createElement( "p" );
  nodeId = "new" + idcount;
  ++idcount;
  newNode.id = nodeId;
  text = "[" + nodeId + "] " + text;
  newNode.appendChild(document.createTextNode( text ) );
  return newNode;
} // end function createNewNode

// helper function that switches to a new currentNode
function switchTo( newNode )
{
  currentNode.className = ""; // remove old highlighting
  currentNode = newNode;
  currentNode.className = "highlighted"; // highlight new node
  document.getElementById( "gbi" ).value = currentNode.id;
} // end function switchTo

// -->
</script>
</head>
<body onload = "currentNode = document.getElementById( 'bigheading' )">
<h1 id = "bigheading" class = "highlighted">[bigheading] DHTML Object Model</h1>
<h3 id = "smallheading">[smallheading] Element Functionality</h3>
<p id = "para1">[para1] The Document Object Model (DOM) allows for
quick, dynamic access to all elements in an XHTML document for
manipulation with JavaScript.</p>
<p id = "para2">[para2] For more information, check out the
"JavaScript and the DOM" section of Deitel's
<a id = "link" href = "http://www.deitel.com/javascript">[link] JavaScript Resource Center.</a></p>
<p id = "para3">[para3] The buttons below demonstrate:(list)</p>
<ul id = "list">
  <li id = "item1">[item1] getElementById and parentNode</li>
  <li id = "item2">[item2] insertBefore and appendChild</li>
  <li id = "item3">[item3] replaceChild and removeChild</li>
</ul>
<div id = "nav" class = "nav">
  <form onsubmit = "return false" action = "">
    <table>
      <tr>
        <td><input type = "text" id = "gbi" value = "bigheading" /></td>
      </tr>
    </table>
  </form>
</div>

Fig. 10.2  Basic DOM functionality. (Part 3 of 8.)
a) This is the page when it first loads. It begins with the large heading highlighted.
b) This is the document after using the **Get By id** button to select para3.

![Image of DOM functionality](image)

- `para3` is selected.

---

c) This is the document after inserting a new paragraph before the selected one.

---

**Fig. 10.2**  |  Basic DOM functionality. (Part 5 of 8.)
d) Using the **Append Child** button, a child paragraph is created.

![Append Child button example]

e) The selected paragraph is replaced with a new one.

![Replacement paragraph example]
f) The **Get Parent** button gets the parent of the selected node.

---

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get By id</td>
<td></td>
</tr>
<tr>
<td>Insert Before</td>
<td></td>
</tr>
<tr>
<td>Append Child</td>
<td></td>
</tr>
<tr>
<td>Replace Current</td>
<td></td>
</tr>
<tr>
<td>Remove Current</td>
<td></td>
</tr>
</tbody>
</table>

---

g) Now we select the first list item.
Chapter 10  Document Object Model (DOM): Objects and Collections

A table (lines 135–160) contains the controls for modifying and manipulating the elements on the page. Each of the six buttons calls its own event-handling function to perform the action described by its value.

The JavaScript code begins by declaring two variables. The variable `currentNode` (line 27) keeps track of the currently highlighted node, because the functionality of the buttons depends on which node is currently selected. The body’s `onload` attribute (line 116) initializes `currentNode` to the `h1` element with id `bigheading`. Variable `idcount` (line 28) is used to assign a unique id to any new elements that are created. The remainder of the JavaScript code contains event handling functions for the XHTML buttons and two helper functions that are called by the event handlers. We now discuss each button and its corresponding event handler in detail.

**Finding and Highlighting an Element Using `getElementById` and `className`**

The first row of the table (lines 136-141) allows the user to enter the id of an element into the text field (lines 137–138) and click the `Get By Id` button (lines 139–140) to find and highlight the element, as shown in Fig. 10.2(b) and (g). The `onclick` attribute sets the button’s event handler to function `byId`.

The `byId` function is defined in lines 31–38. Line 33 uses `getElementById` to assign the contents of the text field to variable `id`. Line 34 uses `getElementById` again to find the element whose id attribute matches the contents of variable `id`, and assign it to variable `target`. If an element is found with the given id, `getElementById` returns an object rep-
representing that element. If no element is found, `getElementById` returns `null`. Line 36 checks whether `target` is an object—recall that any object used as a boolean expression is true, while `null` is false. If `target` evaluates to `true`, line 37 calls the `switchTo` function with `target` as its argument.

The `switchTo` function, defined in lines 106–112, is used throughout the program to highlight a new element in the page. The current element is given a yellow background using the style class `highlighted`, defined in line 22. Line 108 sets the current node’s `className` property to the empty string. The `className` property allows you to change an XHTML element’s `class` attribute. In this case, we clear the `class` attribute in order to remove the `highlighted` class from the `currentNode` before we highlight the new one.

Line 109 assigns the `newNode` object (passed into the function as a parameter) to variable `currentNode`. Line 110 adds the `highlighted` style class to the new `currentNode` using the `className` property.

Finally, line 111 uses the `id property` to assign the current node’s `id` to the input field’s value property. Just as `className` allows access to an element’s `class` attribute, the `id` property controls an element’s `id` attribute. While this isn’t necessary when `switchTo` is called by `byId`, we will see shortly that other functions call `switchTo`. This line makes sure that the text field’s value is consistent with the currently selected node’s `id`. Having found the new element, removed the highlighting from the old element, updated the `currentNode` variable and highlighted the new element, the program has finished selecting a new node by a user-entered `id`.

*Creating and Inserting Elements Using `insertBefore` and `appendChild`*

The next two table rows allow the user to create a new element and insert it before the current node or as a child of the current node. The second row (lines 141–145) allows the user to enter text into the text field and click the `Insert Before` button. The text is placed in a new paragraph element, which is then inserted into the document before the currently selected element, as in Fig. 10.2(c). The button in lines 143–144 calls the `insert` function, defined in lines 42–48.

Lines 44–45 call the function `createNewNode`, passing it the value of the input field (whose `id` is `ins`) as an argument. Function `createNewNode`, defined in lines 94–103, creates a paragraph node containing the text passed to it. Line 96 creates a `p` element using the `document` object’s `createElement` method. The `createElement` method creates a new DOM node, taking the tag name as an argument. Note that while `createElement` creates an element, it does not `insert` the element on the page.

Line 97 creates a unique `id` for the new element by concatenating "new" and the value of `idCount` before incrementing `idCount` in line 98. Line 99 assigns the `id` to the new element. Line 100 concatenates the element’s `id` in square brackets to the beginning of `text` (the parameter containing the paragraph’s text).

Line 101 introduces two new methods. The document’s `createTextNode` method creates a node that can contain only text. Given a string argument, `createTextNode` inserts the string into the text node. In line 101, we create a new text node containing the contents of variable `text`. This new node is then used (still in line 101) as the argument to the `appendChild` method, which is called on the paragraph node. Method `appendChild` is called on a parent node to insert a child node (passed as an argument) after any existing children.
After the `p` element is created, line 102 returns the node to the calling function `insert`, where it is assigned to variable `newNode` in lines 44–45. Line 46 inserts the newly created node before the currently selected node. The `parentNode` property of any DOM node contains the node's parent. In line 46, we use the `parentNode` property of `currentNode` to get its parent.

We call the `insertBefore` method (line 46) on the parent with `newNode` and `currentNode` as its arguments to insert `newNode` as a child of the parent directly before `currentNode`. The general syntax of the `insertBefore` method is

```javascript
parent.insertBefore(newChild, existingChild);
```

The method is called on a parent with the new child and an existing child as arguments. The node `newChild` is inserted as a child of `parent` directly before `existingChild`. Line 47 uses the `switchTo` function (discussed earlier in this section) to update the `currentNode` to the newly inserted node and highlight it in the XHTML page.

The third table row (lines 145–149) allows the user to append a new paragraph node as a child of the current element, demonstrated in Fig. 10.2(d). This feature uses a similar procedure to the `insertBefore` functionality. Lines 53–54 in function `appendNode` create a new node, line 55 inserts it as a child of the current node, and line 56 uses `switchTo` to update `currentNode` and highlight the new node.

### Replacing and Removing Elements Using `replaceChild` and `removeChild`

The next two table rows (lines 149–156) allow the user to replace the current element with a new `p` element or simply remove the current element. Lines 150–152 contain a text field and a button that replaces the currently highlighted element with a new paragraph node containing the text in the text field. This feature is demonstrated in Fig. 10.2(e).

The button in lines 151–152 calls function `replaceCurrent`, defined in lines 60–66. Lines 62–63 call `createNewNode`, in the same way as in `insert` and `appendNode`, getting the text from the correct input field. Line 64 gets the parent of `currentNode`, then calls the `replaceChild` method on the parent. The `replaceChild` method works as follows:

```javascript
parent.replaceChild(newChild, oldChild);
```

The `parent`'s `replaceChild` method inserts `newChild` into its list of children in place of `oldChild`.

The `Remove Current` feature, shown in Fig. 10.2(h), removes the current element entirely and highlights the parent. No text field is required because a new element is not being created. The button in lines 154-155 calls the `remove` function, defined in lines 69–79. If the node's parent is the body element, line 72 alerts an error—the program does not allow the entire body element to be selected. Otherwise, lines 75–77 remove the current element. Line 75 stores the old `currentNode` in variable `oldNode`. We do this to maintain a reference to the node to be removed after we’ve changed the value of `currentNode`. Line 76 calls `switchTo` to highlight the parent node.

Line 77 uses the `removeChild` method to remove the `oldNode` (a child of the new `currentNode`) from its place in the XHTML document. In general,

```javascript
parent.removeChild(child);
```

looks in `parent`'s list of children for `child` and removes it.
The final button (lines 157–158) selects and highlights the parent element of the currently highlighted element by calling the parent function, defined in lines 82–90. Function parent simply gets the parent node (line 84), makes sure it is not the body element, (line 86) and calls switchTo to highlight it (line 87). Line 89 alerts an error if the parent node is the body element. This feature is shown in Fig. 10.2(f).

This section introduced the basics of DOM tree traversal and manipulation. Next, we introduce the concept of collections, which give you access to multiple elements in a page.

## 10.4 DOM Collections

Included in the Document Object Model is the notion of **collections**, which are groups of related objects on a page. DOM collections are accessed as properties of DOM objects such as the document object or a DOM node. The document object has properties containing the **images collection, links collection, forms collection** and **anchors collection**. These collections contain all the elements of the corresponding type on the page. Figure 10.3 gives an example that uses the links collection to extract all of the links on a page and display them together at the bottom of the page.

```html
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 10.3: collections.html -->
<!-- Using the links collection. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <style type = "text/css">
      body { font-family: arial, helvetica, sans-serif }
      h1 { font-family: tahoma, geneva, sans-serif; text-align: center }
      p { margin: 5% }
      p a { color: #aa0000 }
      .links { font-size: 14px; text-align: justify; margin-left: 10%; margin-right: 10% }
      .link a { text-decoration: none }
      .link a:hover { text-decoration: underline }
    </style>
  </head>
  <script type = "text/javascript">
    function processlinks()
    {
      var linkslist = document.links; // get the document's links
      var contents = "Links in this page:
\n| ";
    
    // concatenate each link to contents
    for ( var i = 0; i < linkslist.length; i++ )
    {
      contents += linkslist[i].href + " |
    }

    alert ( contents );
  </script>
</html>
```

*Fig. 10.3* | Using the links collection. (Part 1 of 2.)
var currentLink = linkslist[i];
contents += "<span class = 'link'>"
        currentLink.innerHTML.link( currentLink.href ) +
"</span> | ";
    } // end for

document.getElementById( "links" ).innerHTML = contents;
} // end function processlinks
// -->
</script>
</head>
<body onload = "processlinks()"
<h1>Deitel Resource Centers</h1>
<p><a href = "http://www.deitel.com/">Deitel's website</a> contains
a rapidly growing
<a href = "http://www.deitel.com/ResourceCenters.html">list of
Resource Centers</a> on a wide range of topics. Many Resource
centers related to topics covered in this book,
<a href = "http://www.deitel.com/iw3htp4">Internet and World Wide
Web How to Program, 4th Edition</a>. We have Resource Centers on
<a href = "http://www.deitel.com/Web2.0">Web 2.0</a>,
<a href = "http://www.deitel.com/Firefox">Firefox</a> and
<a href = "http://www.deitel.com/IE7">Internet Explorer 7</a>,
<a href = "http://www.deitel.com/XHTML">XHTML</a>, and
<a href = "http://www.deitel.com/JavaScript">JavaScript</a>
Watch the list of Deitel Resource Centers for related new
Resource Centers.</p>
</body>
</html>
The XHTML body contains a paragraph (lines 46–59) with links at various places in the text and an empty div (line 60) with id "links". The body’s onload attribute specifies that the processLinks method is called when the body finishes loading.

Method processLinks declares variable linksList (line 27) to store the document’s links collection, which is accessed as the links property of the document object. Line 28 creates the string (contents) that will contain all the document’s links, to be inserted into the links div later. Line 31 begins a for statement to iterate through each link. To find the number of elements in the collection, we use the collection’s length property.

Line 33 inside the for statement creates a variable (currentLink) that stores the current link. Note that we can access the collection stored in linksList using indices in square brackets, just as we did with arrays. DOM collections are stored in objects which have only one property and two methods—the length property, the item method and the namedItem method. The item method—an alternative to the square bracketed indices—can be used to access specific elements in a collection by taking an index as an argument. The namedItem method takes a name as a parameter and finds the element in the collection, if any, whose id attribute or name attribute matches it.

Lines 34–36 add a span element to the contents string containing the current link. Recall that the link method of a string object returns the string as a link to the URL passed to the method. Line 35 uses the link method to create an a (anchor) element containing the proper text and href attribute.

Notice that variable currentLink (a DOM node representing an a element) has a specialized href property to refer to the link’s href attribute. Many types of XHTML elements are represented by special types of nodes that extend the functionality of a basic DOM node. Line 39 inserts the contents into the empty div with id "links" (line 60) in order to show all the links on the page in one location.

Collections allow easy access to all elements of a single type in a page. This is useful for gathering elements into one place and for applying changes across an entire page. For example, the forms collection could be used to disable all form inputs after a submit button has been pressed to avoid multiple submissions while the next page loads. The next section discusses how to dynamically modify CSS styles using JavaScript and DOM nodes.

10.5 Dynamic Styles

An element’s style can be changed dynamically. Often such a change is made in response to user events, which we discuss in Chapter 11. Such style changes can create many effects, including mouse hover effects, interactive menus, and animations. Figure 10.4 is a simple example that changes the background-color style property in response to user input.
Function `start` (lines 12–17) prompts the user to enter a color name, then sets the background color to that value. [Note: An error occurs if the value entered is not a valid color.] We refer to the background color as `document.body.style.backgroundColor`—the `body` property of the document object refers to the body element. We then use the `style` property (a property of most XHTML elements) to set the background-color CSS property. This is referred to as `backgroundColor` in JavaScript—the hyphen is removed to avoid confusion with the subtraction (−) operator. This naming convention is consistent for most CSS properties. For example, `borderWidth` correlates to the `border-width` CSS property, and `fontFamily` correlates to the `font-family` CSS property. In general, CSS properties are accessed in the format `node.style.styleproperty`.

Figure 10.5 introduces the `setInterval` and `clearInterval` methods of the `window` object, combining them with dynamic styles to create animated effects. This example is a basic image viewer that allows you to select a Deitel book cover and view it in a larger size. When one of the thumbnail images on the right is clicked, the larger version grows from the top-left corner of the main image area.

The body (lines 66–85) contains two `div` elements, both floated `left` using styles defined in lines 14 and 17 in order to present them side by side. The left `div` contains the full-size image `iw3http4.jpg`, the cover of this book, which appears when the page loads.
The right div contains six thumbnail images which respond to the click event by calling the display method and passing it the filename of the corresponding full-size image.

The display function (lines 46–62) dynamically updates the image in the left div to the one corresponding to the user’s click. Lines 48–49 prevent the rest of the function from executing if interval is defined (i.e., an animation is in progress.) Line 51 gets the left div by its id, imgCover. Line 52 creates a new img element. Lines 53–55 set its id to imgCover, set its src to the correct image file in the fullsize directory, and set its required alt attribute. Lines 56–59 do some additional initialization before beginning the animation in line 61. To create the growing animation effect, lines 57–58 set the image width and height to 0. Line 59 replaces the current bigImage node with newNode (created in line 52), and line 60 sets count, the variable that controls the animation, to 0.

Line 61 introduces the window object’s setInterval method, which starts the animation. This method takes two parameters—a statement to execute repeatedly, and an integer specifying how often to execute it, in milliseconds. We use setInterval to call

```javascript
function run()
{
    count += speed;
    if ( count >= 375 ) {
```

Fig. 10.5 | Dynamic styles used for animation. (Part 1 of 4.)
window.clearInterval( interval );
interval = null;
} // end if

var bigImage = document.getElementById( "imgCover" );
bigImage.style.width = .7656 * count + "px";
bigImage.style.height = count + "px";
} // end function run

// inserts the proper image into the main image area and
// begins the animation
function display( imgfile )
{
if ( interval )
return;

var bigImage = document.getElementById( "imgCover" );
var newNode = document.createElement( "img" );
newNode.id = "imgCover";
newNode.src = "fullsize/" + imgfile;
newNode.alt = "Large image";
newNode.className = "imgCover";
newNode.style.width = "0px";
newNode.style.height = "0px";
bigImage.parentNode.replaceChild( newNode, bigImage );
count = 0; // start the image at size 0
interval = window.setInterval( "run()", 10 ); // animate
} // end function display

// -->
</script>
</head>
<body>
<div id = "mainimg" class = "mainimg">
  <img id = "imgCover" src = "fullsize/iw3htp4.jpg" alt = "Full cover image" class = "imgCover" />
</div>

<div id = "thumbs" class = "thumbs">
  <img src = "thumbs/iw3htp4.jpg" alt = "iw3htp4" onclick = "display( 'iw3htp4.jpg' )" />
  <img src = "thumbs/chtpt5.jpg" alt = "chtpt5" onclick = "display( 'chtpt5.jpg' )" />
  <img src = "thumbs/cpphtp6.jpg" alt = "cpphtp6" onclick = "display( 'cpphtp6.jpg' )" />
  <img src = "thumbs/jhtp7.jpg" alt = "jhtp7" onclick = "display( 'jhtp7.jpg' )" />
  <img src = "thumbs/vbhtp3.jpg" alt = "vbhtp3" onclick = "display( 'vbhtp3.jpg' )" />
  <img src = "thumbs/vcsharphtp2.jpg" alt = "vcsharphtp2" onclick = "display( 'vcsharphtp2.jpg' )" />
</div>
</body>
</html>

Fig. 10.5  |  Dynamic styles used for animation. (Part 2 of 4.)
a) The cover viewer page loads with the cover of this book.

b) When the user clicks the thumbnail of *C How to Program*, the full-size image begins growing from the top-left corner of the window.
c) The cover continues to grow.

d) The animation finishes when the cover reaches its full size.

Fig. 10.5  |  Dynamic styles used for animation. (Part 4 of 4.)
function run every 10 milliseconds. The `setInterval` method returns a unique identifier to keep track of that particular interval—we assign this identifier to the variable `interval`. We use this identifier to stop the animation when the image has finished growing.

The `run` function, defined in lines 28–42, increases the height of the image by the value of `speed` and updates its width accordingly to keep the aspect ratio consistent. Because the `run` function is called every 10 milliseconds, this increase happens repeatedly to create an animated growing effect. Line 30 adds the value of `speed` (declared and initialized to 6 in line 24) to `count`, which keeps track of the animation’s progress and dictates the current size of the image. If the image has grown to its full height (375), line 35 uses the window’s `clearInterval` method to stop the repetitive calls of the `run` method. We pass to `clearInterval` the interval identifier (stored in `interval`) that `setInterval` created in line 61. Although it seems unnecessary in this script, this identifier allows the script to keep track of multiple intervals running at the same time and to choose which interval to stop when calling `clearInterval`.

Line 39 gets the image and lines 40–41 set its `width` and `height` CSS properties. Note that line 40 multiplies `count` by a scaling factor of .7656 in order to keep the ratio of the image’s dimensions consistent with the actual dimensions of the image. Run the code example and click on a thumbnail image to see the full animation effect.

This section demonstrated the concept of dynamically changing CSS styles using JavaScript and the DOM. We also discussed the basics of how to create scripted animations using `setInterval` and `clearInterval`.

### 10.6 Summary of the DOM Objects and Collections

As you’ve seen in the preceding sections, the objects and collections in the W3C DOM give you flexibility in manipulating the elements of a web page. We’ve shown how to access the objects in a page, how to access the objects in a collection, and how to change element styles dynamically.

The W3C DOM allows you to access every element in an XHTML document. Each element in a document is represented by a separate object. The diagram in Fig. 10.6 shows many of the important objects and collections provided by the W3C DOM. Figure 10.7 provides a brief description of each object and collection in Fig. 10.6.

![Fig. 10.6 | W3C Document Object Model.](image-url)
### Objects and Collections in the W3C Document Object Model

<table>
<thead>
<tr>
<th>Object or Collection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects</strong></td>
<td></td>
</tr>
<tr>
<td>window</td>
<td>Represents the browser window and provides access to the window's document object. Also contains history and location objects.</td>
</tr>
<tr>
<td>document</td>
<td>Represents the XHTML document rendered in a window. Provides access to every element in the document and allows dynamic modification of it. Contains collections for accessing all elements of a given type.</td>
</tr>
<tr>
<td>body</td>
<td>Provides access to the body element of an XHTML document.</td>
</tr>
<tr>
<td>history</td>
<td>Keeps track of the sites visited by the browser user. The object provides a script programmer with the ability to move forward and backward through the visited sites.</td>
</tr>
<tr>
<td>location</td>
<td>Contains the URL of the rendered document. When this object is set to a new URL, the browser immediately navigates to the new location.</td>
</tr>
<tr>
<td><strong>Collections</strong></td>
<td></td>
</tr>
<tr>
<td>anchors</td>
<td>Collection contains all the anchor elements (a) that have a name or id attribute. The elements appear in the collection in the order in which they were defined in the XHTML document.</td>
</tr>
<tr>
<td>forms</td>
<td>Contains all the form elements in the XHTML document. The elements appear in the collection in the order in which they were defined in the XHTML document.</td>
</tr>
<tr>
<td>images</td>
<td>Contains all the img elements in the XHTML document. The elements appear in the collection in the order in which they were defined in the XHTML document.</td>
</tr>
<tr>
<td>links</td>
<td>Contains all the anchor elements (a) with an href property. The elements appear in the collection in the order in which they were defined in the XHTML document.</td>
</tr>
</tbody>
</table>

**Fig. 10.7** | Objects and collections in the W3C Document Object Model.

For a complete reference on the W3C Document Object Model, see the DOM Level 3 recommendation from the W3C at [http://www.w3.org/TR/DOM-Level-3-Core/](http://www.w3.org/TR/DOM-Level-3-Core/). The DOM Level 2 HTML Specification (the most recent HTML DOM standard), available at [http://www.w3.org/TR/DOM-Level-2-HTML/](http://www.w3.org/TR/DOM-Level-2-HTML/), describes additional DOM functionality specific to HTML, such as objects for various types of XHTML elements. Keep in mind that not all web browsers implement all features included in the specification.

#### 10.7 Web Resources

**www.deitel.com/javascript/**

The Deitel JavaScript Resource Center contains links to some of the best JavaScript resources on the web. There you’ll find categorized links to JavaScript tools, code generators, forums, books, libraries, frameworks, tutorials and more. Check out the section specifically dedicated to the Document Object Model. Be sure to visit the related Resource Centers on XHTML (www.deitel.com/xhtm1/) and CSS 2.1 (www.deitel.com/css21/).
cross-bone events 346

cross-browser events (CSS) 418

CRUD (create, retrieve, update, destroy) 667, 668

CSS (Cascading Style Sheets) xix, 32, 68, 101, 412, 683

count 109

CSS 2 101

CSS 3 130

CSS 3 features 130

drop-down menu 124

property 102

rule 104

selector 103

stylesheet 805, 814

Ctrl key 333

ctrlKey property of an event object 332, 333

Cubic.easeIn function 531

curly brace ( ) 105

cursive font 105

cursor 140

custom map 37

custom tag libraries 801

custom tags 801

custom type 939

customer relationship management (CRM) 30

customize proportions 480

D

danglingElse problem 158

dashed value (border-style property) 119

data 135, 603

data cells 86

data keyword 524

data method of a Text node 398

data provider 863, 882

data pull 519

data source 603

data tier 574

data.php 610

database xx, 68, 586, 591, 653

table 587

database connectivity 630

database handle 633

database management system (DBMS) 586

database migration script 667, 670, 674, 676

dataColumn class 603

dataField attribute 536

dataGrid control

PageSize property 789

DataGrid element 531, 536

dataProvider attribute 536

DataGridColumn element 536

dataField attribute 536

headerText attribute 536

dataProvider attribute (Flex) 524, 536, 566

DataRow class 603

DataSet class 603

using in a Web service 957

DataSourceID property of a GridView

ASP.NET control 759

DataTable class 603
Examples (cont.)

Creating a new symbol with the Convert to Symbol dialog 462
Creating a rectangle with rounded corners 478
Creating a rotating object with the motion tween Rotate option 486
Creating a shape tween 481
Creating a simple book cover viewer in Flex 2 520
Creating a text field 470
Creating a title with the Text tool 468
Creating an animation with a symbol 482
Creating an oval graphic 476
Creating multiple instances of a symbol with the Library panel 479
CSS drop-down menu 124
data.php 610
Database migration script modified to add data to the table 667, 674, 676

Examples (cont.)

Database migration script modified to add data to the users table 670
Default layout of Script.aculo.us demo 693
Default namespaces demonstration 361
Default view of Script.aculo.us demo 692
Defining the hit area of a button 489
Detial Resource Centers 479
Demonstrating date and time methods of the Date object 274
Demonstrating logical operators 200
Demonstrating the onfocus and onblur events 339
Demonstrating the onload event 327
Demonstrating the onsubmit and onreset events 341
Demonstration of a document's DOM tree 300
Detect collision using hitTestObject 514
Detecting a miss 511
Dice-rolling program using arrays instead of switch 241
Differences between preincrementing and postincrementing 171
Disabling event bubbling 343
Display the name of the current action in the title bar 673
Displaying a list of messages without reloading the page 687
Displaying the cookie's contents 639
Displaying the MailList database 647
Document Type Definition (DTD) for a business letter 362
dom.html 303
domtree.html 300
DowhileTest.html 191
Duplicating movie clip symbols with the Library panel 483
Dynamic form using PHP 640
Dynamic styles 315
Dynamic styles used for animation 317
dynaticstyle.html 315
Element dimensions and text alignment 117
em measurement for text size 129
embedded style sheets 103
Employee controller provides all of the functionality for the application 668
Event registration models 324
Events onmouseover and onmouseout 333
Examination-results problem 167
Expression marked up with MathML and displayed in the Amaya browser 373
expression.php 619
Factorial calculation with a recursive function 229
Fire ball on click event 507

Examples (cont.)

First program in JavaScript 133
First XHTML example 70
first.php 608
Floating elements 120
ForCounter.html 177
form including radio buttons and a drop-down list 91
Form that allows the user to enter a new message 685
Form to query a MySQL database 531
form with hidden fields and a text box 89
form.html 89
form2.html 91
Forum model that includes validation and initialization functionality 676
Forum that allows the user to add another message on the same page 688
Forums layout that uses the default JavaScript libraries 686
ForumsController implements CRUD functionality 677
fullSizeImage view of Flickr photo viewer 699
Gathering data to be written as a cookie 635
Generated code for an Employee model 668
Generated code for the User model 671
Handle playButton click event 498
Handle the map functions for the Address Book application 541
Handle the map functions for the Weather application 553
heading elements h1 through h6 73
heading.html 73
HTML wrapper for coverViewer.swf 526
HTTPService, Validators, and new UI elements 531
BigInteger web service that performs operations on large integers 896
Images as link anchors 78
Images in XHTML files 76
Implement the actions of each button 544
Important methods of class HttpCookie 746
increment.html 171
inheritance in style sheets 107
Initializing multidimensional arrays 250
Initializing the elements of an array 235, 238
Input and password text field creation 479
Inserting a keyframe 464
Inserting special characters 80
Interaction between a web service client and a web service 894
Interest.html 183
internal hyperlinks to make pages more navigable 95
Examples (cont.)  
'internal.html' 95  
JSP file for the AddressBook page with a search form and Table component 864  
JSP file generated by Netbeans that displays the current time on the web server 803  
JSP file that allows the user to select a programming language 835, 847  
JSP file that displays book recommendations based on cookies 842  
JSP file that displays book recommendations based on language selections stored in session scope 855  
JSP for the AddressBook page with MapViewer component 883  
Labeled break statement in a nested for statement 195  
Labeled continue statement in a nested for statement 196  
Layout that displays a greeting 665  
Layout that displays the logged in user for every Forums action 681  
Link view of Script.aculo.us demo 693  
linking an external style sheet 110  
linking to an e-mail address 75  
Linking to other Web pages 74  
'links.html' 74  
'links2.html' 82  
'1st.html' 83  
LogicalOperators.html 200  
Login form used to send data to the user controller 672  
Main view for Flickr Photo Viewer 697  
'main.html' 70  
Making multiple selections with the Selection tool 459  
 Mashups 35  
Media types 122  
Message layout that links a style in and displays a message 686  
Message model containing validation and initialization functionality 675  
MessagesController that implements CRUD functionality 683  
'meta.html' 98  
Method calls on numeric and string data 656  
Modifying button states with a button's editing stage 464  
Modifying the size of a shape with the Property Inspector 460  
Move ball and set speed 506  
Move target, set direction and speed, and keep track of number of blocks hit 499  
mxmlc running in Windows XP  
Command Prompt 525  
'nav.html' 78  
Nested and ordered lists. 83  

Examples (cont.)  
Object animation in ActionScript 496  
Obtaining user input through forms 626  
Online Quiz graded with JavaScript 253  
Online quiz in a child window 294  
operators.php 613  
Optimizing sound with the Sound Properties dialog 466  
Page bean file that sets clockText to the time on the web server 806  
Page bean for adding a contact to the AddressBook 870  
Page bean for validating user input and redisplaying that input if valid 831  
Page bean that displays book recommendations based on a SessionBean property 855  
Page bean that displays book recommendations based on cookies storing user's selected languages 844  
Page bean that gets a map to display in the MapViewer component 887  
Page bean that stores language selections in a SessionBean property 852  
Page bean that stores user's language selection as a cookie on the client 839  
Page bean that suggests names in the AutoComplete TextField 879  
Partial that contains a form used to add a new forum 680  
Passing arrays and individual array elements to functions 245  
'picture.html' 76  
pit media type 122  
Printing on multiple lines with a single statement 138  
Printing on one line with separate statements 136  
Printing one line with separate statements 136  
Programmer-defined function square 206  
Programmer-defined maximum function 209  
Prompt box used on a welcome screen 141  
pt measurement for text size 126  
Querying a database and displaying the results 632  
Random image generation using arrays 242  
Random image generation using Math.random 222  
Registration form that demonstrates JSF components 821  
Regular expressions in PHP 619  
Relative positioning of elements 113  
Renaming a layer 467  
Rendering an RSS feed in a web page using XSLT and JavaScript 406  
Examples (cont.)  
Reset the ball to its original position 514  
Rich welcome page using several JavaScript concepts 288  
Rolling a six-sided die 6000 times 213  
RSS 2.0 sample feed 408  
Ruby class 459  
Running mxmlc with the theme parameter in Windows XP  
Command Prompt 558  
Schema-valid XML document describing a list of books 366  
Scoping example 224  
Script.aculo.us Demo controller 692  
Searching Strings with indexOf and lastIndexOf 266  
Selecting a background color 457  
Selecting an object with the Selection tool 462  
Sentinel-controlled repetition to calculate a class average 164  
SessionBean file for storing language selections 851  
SessionBean that initializes the data source for the AddressBook database 868  
Setting text alignment with the Property Inspector 468  
Setting the fill and stroke colors 458  
Setting the font face, size, weight and color with the Property Inspector 461  
Shifted and scaled random integers 211  
Simple controller that passes a parameter to the view 664  
Simple controller that renders a message on the web page 662  
Simple drawing program 329  
Simple PHP program 608  
Simple Ruby program 654  
SOAP request for the HugeInteger web service 915  
some common escape sequences 140  
Sorting an array with Array method sort 247  
Stops all timers and sends objects into hidden frame 510  
String methods charAt, charCodeAt, fromCharCode, toLowerCase and toUpperCase 264  
Sum.html 182  
Summation with for 182  
SwitchTest.html 186  
'table1.html' 85  
'table2.html' 87  
Template for a list action that displays a list of messages 684  
Template for a new action that adds a forum to the forums table 680  
Template for Delete action used to delete a forum 681  
Template for the list action that displays a list of messages 679
HTTPService element 538
URL attribute 538
HttpServletRequest, Validators, and new UI elements 531
HttpServletRequestRequest class 844
HttpServletRequestResponse class 842
HttpServletRequestSession class 842
HttpServletRequestSessionState class 746, 747, 748, 750, 753
Counts property 750
IsCookieless property 750
IsNewSession property 750
IsReadOnly property 750
Key property 751, 753
SessionID property 750, 751
Timeout property 750, 751
HugeInteger web service that performs operations on large integers 896
“human bandwidth” 8
hyperlink 73, 271
Hyperlink ASP.NET Web control 721
Hyperlink JSF component 820, 825
hyperlink location 263
hyperlink tag 271
hypertext link 488
Hypertext Transfer Protocol (HTTP) 570, 573, 738, 834

I

IAC (InterActiveCorp) 10
IBM 22, 31
Ice theme 558
IceRocket (blog search engine) 19, 41
td attribute 114, 522
ID attribute of an ASP.NET web control 704
td property of a DOM node 311
identifier 142, 154
identifier element (MathML) 374
identifier followed by a colon 196
IDEs
Netbeans 893
if control structure 491
if selection statement 154, 155, 156, 159
if single-selection statement 148
if...else double-selection statement 154, 156, 169, 658
ISS (Internet Information Services) 570
Image ASP.NET web control 720
Image element 522
completeEffect attribute 524
percentHeight attribute 522
percentWidth attribute 522
source attribute 523
image hyperlink 79
Image JSF component 820, 824
Image object 337
src property 338
image/gif 104
Images as link anchors 78
Images collection of the document object 313, 322
images in Web pages 76
images in XHTML files 76
ImageUrl property of an Image web control 720
img element 76, 77, 79, 111
alt attribute 77
height attribute 76
img elements in an XHTML document 322
Implement the actions of each button 544
implementation property of the document object 394
#IMPLIED keyword (DTD) 363
import 496
Import to Stage... 473
Important methods of class HttpCookie 746
importStylesheet method of the XSLTProcessor object (Firefox) 409
Impressions element in an ADRotator advertisement file 724
IMVU 41
in (inches) 109
increment 175
increment control variable 177, 181
increment expression 180
increment operator (++) 170
increment section of for statement 180
increment.html 171
Indeed 10, 40, 42
indentation body of control structure 177
index 262
index in an array 233
index value 634
indexOf method of the String object 262, 265, 266
indices for the characters in a string 264
infinity loop 160, 166, 179, 191
infinity symbol 590
information hiding 258
information tier 574
infrastructure for open source projects 42
internal linking 68
internal hyperlink 97
Internal hyperlinks to make pages more navigable 95
internal linking 68, 95
internal pointer 617
Internet bookmark 28
Internet Explorer 68, 77, 353
Install Rails 461
instantiated 259
instantiating an object 235
institutional theme 558
integer 145
Integrity symbol (MathML) 374
interaction between a web service client and a web service 894
interactive animated movies 453
InterActiveCorp (IAC) 10
interest rate 183
interface 258, 586
Interfaces
CachedRowSet 863
MessageContext 919
RequestContext 921
Servlet 800
WebServiceContext 919
internal hyperlink 97
internal hyperlinks to make pages more navigable 95
internal linking 68, 95
internal pointer 617
Internet bookmark 28
Internet Explorer 68, 77, 353
Develop toolbar 300
Developer Toolbar (go.microsoft.com/fwlink/?LinkId=92716) 300
Internet Explorer 7
load XML document dynamically with JavaScript 394
Internet Information Services (ISS) 570, 576
Internet Information Services Manager 579

INDEX 975

INNERT JOIN SQL clause 591, 596
innerHTML property (DOM) 421
innerHTML property of a DOM node 221
innerText 294, 328
input 738, 834
input and password text field creation 479
input element 90
maxlength attribute 91
name attribute 91
input text field 478
Insert Keyframe 464
insert layer 467
INSERT SQL statement 591, 597
insertBefore method of a DOM node 312
insertBefore method of a Node 396
InsertCommand property of a SqlDataSource ASP.NET control 760
inserting a keyframe 464
Inserting special characters 80
InsertQuery property of a SqlDataSource ASP.NET data control 758
insert value (border-style property) 119
installing Rails 661
instance 461, 462
instance name 471
instance variables 659, 665
Instant Rails 654
instantiated 259
instantiating an object 235
Institutional theme 558
Integer 610
integers 145
integral symbol (MathML) 374
internet service 97
Internet Explorer 68
Internet Explorer 7
load XML document dynamically with JavaScript 394
Internet Information Services (ISS) 570, 576
Internet Information Services Manager 579
movie clips 461
dmoviedimensions 458
Movie Explorer 462
Movie Explorer for CeoAssistant fla 463
Movie Properties dialog 456
MovieLens 43
movies 453
moving the mouse 324
Mozilla Firefox 68
Mozilla Firefox 2 xx
Mozilla Foundation 31, 43
Mozilla project 376
mp3 25
MP3 audio compression format 465
nrow MathML element 375
msdn.microsoft.com/vstudio/
express/vwd/default.aspx 702
MSN 9, 43
MSN Hotmail 44
MSN Virtual Earth 38
msart MathML element 375
msub sup element 375
msub sup MathML element 375
msup MathML element 374
MSXML (Microsoft XML Core Services) 350, 353
MSXML parser 394
MTV 24
Muiar (www.sounds.muinar.com) 465
multidimensional array 249
multiline comment /* and */ 143
multimedia xx
multiple conditions 197
multiple-selection statement 154
multiple-selection structure 189
multiplication assignment operator (*=) 612
Multipurpose Internet Mail Extensions (MIME) 572, 739, 835
Multipurpose Internet Mail Extensions (MIME) type 104, 110
multitier application xx, 574
music distribution sites 42
MVC (Model-View-Controller) 660
mx prefix 522
mXML mark-up language 518
mxml
  theme parameter 558
mxml command 525
mxml running in Windows XP
  Command Prompt 525
mysimon 10
MySpace 5, 19, 20, 24, 43
MySpaceTV 24
MySQL 31, 586, 600, 607, 630, 654
MySQL 5.0 Community Edition 600
MySQL command-line client tool 601, 602
MySQL Conference and Expo 7
MySQL database 602
MySQL reference manual 600
mysql_close function 634
mysql_connect function 636
mysql_error function 634
mysql_fetch_row function 634
mysql_num_rows function 634
mysql_query function 634
mysql_select db function 634
mysqld-nt.exe 601
MyStrands 43
mySql
  n-tier application 574
  name attribute 530
  name attribute (XPath) 384
  name attribute of input element 91
  name attribute of meta element 98
  name node-set function 384
  name of an array 233
  name of an attribute 71
  name of the anchor 271
  name property of an Attr object 397
  Name property of class HttpCookie 745
  Name property of HttpCookie class 746
  name XML Schema attribute 368
  named constant 612
  namedItem method of a DOM collection 315
  namespace 358
  namespace prefix 359, 360, 370
  Namespace property of a WebService
    attribute 951
  Namespaces
    System.Data 603
    System.Data.OleDb 603
    System.Data.SqlClient 603
    System.Web.UI 705
    System.Web.UI.WebControls 706
  naming collision 358
  NaN 226, 262, 265, 278
  NaN (not a number) 146
  NaN (not a number) 163
  natural linking 11
  natural logarithm 260
  natural logarithm of 10 261
  natural logarithm of 10 261
  natural logarithm of 2 261
  nav.html 78
  navigateToURL function 543
  navigateToURL function 491
  Navigate1 element in an AdRotator
    advertisement file 724
  Navigate1 property of a HyperLink
    control 721
  navigation bar on a website 774
  NAVTEQ 38
  Nested and ordered lists. 83
  nested element 71, 352
  nested for statement 195, 196, 252
  nested for...in statement 251
  nested if or else...else statement 197
  nested if statement 151, 158
  nested if...else statements 157
  nested list 82, 109
  nesting 177
  nesting multi-line comments 143
Netbeans 799
  Visual Web Pack 799
Netbeans 5.5 IDE (cont.)
  New Web Service Client 895
  Project Properties dialog 901
  Web Application project 895
Netfix 6, 43
NetNewsWire 42
Netscape 25, 32
LiveScript 32
Netsib 43
network effect 19
Network Solutions 42
new Date object 273, 276
new Document dialog 456
new JFrame Form dialog 908
new layer 467
new operator 150, 234, 237, 273
New Project dialog 895
new symbol 480
New Web Service Client dialog 905
New Web Service dialog 895
  new WebService method 395
nextInt method of class Random 944
newSIBling property of a DOM node 395
newSIBling property of a Node 396
n1 655
9rules (blog network) 13, 18, 41
no color 458, 475
No color option 475
no-repeat property 116
node (DOM tree) 385
Node object 396
Node object methods 396
node-set function 384
node set of an xsl 350
  for-each element 380
  NodeList object 396, 397
  NodeList object methods 397
  nodeName property of a DOM node 395
  nodeName property of a Node 396
  nodeType property of a DOM node 395
  nodeValue property of a DOM node 395
  nodeValue property of a Node 396
  nonbreaking space (&nbsp;) 395
  none value (border-style property) 119
  none value (font-style property) 116
  nonvalidating XML parser 350
  normal value (font-weight property) 104
  not a number 146
  not a number (NaN) 163
  Notepad text editor 69
  nouns in a system specification 259
  nu11 144, 147
  Number object 146, 185, 277
  toFixed method 185
  Number.MAX_VALUE 278
  Number.MIN_VALUE 278
  Number.NaN 278
  Number.NEGATIVE_INFINITY 226, 278
Number.POSITIVE_INFINITY 226, 278
numbered list 188
numeric character reference 80

O
O'Reilly Media 3, 29, 42
O'Reilly, Tim 4, 5, 34
object 135, 226, 257
object (or instance) 257, 258
object animation in ActionScript 496
object-based programming language 257
Object class 524
object hierarchy 299
object orientation 257
object-oriented design (OOD) 258
object-oriented language 259
object-oriented programming (OOP) 259
Object-Relational Mapping (ORM) 665
ObjectDataSource ASP.NET data control 783
oblique value (font-style property) 116
obtaining user input through forms 626
occurrence indicator 363
octal 227
off-by-one error 179
Offline Problem 33
OK button 140
01 element 82
omit-xml-declaration attribute 380
ON clause 596
onabort event 346
onblur event 338, 339
onchange event 346
onclick event 220, 265, 266, 268, 269,
283, 327, 345, 346
ondblclick event 346
1000tags.com 41
one-to-many relationship 590
onfocus event 338, 339, 346
onfocusblur.html 339, 341
onkeydown event 346
onkeypress event 346
onkeyup event 346
online advertising 42
DoubleClick 42
Federated Media 42
Google 42
Kontera 42
Microsoft 42
Quigo 42
Tribal Fusion 42
ValueClick 42
Vibrant Media 42
Yahoo! 42
online auctions 42
Amazon Auctions 42
eBay 42
Overstock.com 42
online classifieds 42
Online Quiz graded with JavaScript 253
online quiz in a child window 294
online quizzes 253
online survey sites 43
Survey Monkey 43
onload event 224, 327, 346
onload.html 327
onmousedown event 346
onmousemove event 328, 332, 333, 346
onmouseout event 333, 338, 346, 417
onmouseover event 333, 337, 338, 346,
417
onmouseup event 346
onReadyStateChange property of the
HttpRequest object 422
onReadyStateChange property of the
HttpRequest object 421
onreset event 341, 346
onresize event 346
onselect event 346
onsubmit event 255, 341, 346
ontology 47
onunload event 346
OOD (object-oriented design) 258
OOP (object-oriented programming) 259
open method of the XMLHttpRequest
object 422
open source 4, 5, 6, 25, 30, 43, 581, 600,
607
Apache 43
Free Software Foundation 43
Mozilla Foundation 43
Zend 43
Open Source (OSCON) conference 7
Open Source Initiative 31
open technology 348
openField attribute 566
OpenGIS Consortium 376
OpenLaszlo 29
operand 144
operation element 551
operation in the UML 144
operator precedence chart 649
Operators
function-call operator 207
new 234
operators '
(logical NOT or logical negation)
operator 198
&& (logical AND) operator 197
|| (logical OR) operator 197
operators.php 613
optimizing sound with the Sound
Properties dialog 466
Option 845
option element (form) 95
Oracle Corporation 586
Orbitz 44
order attribute 384
ORDER BY SQL clause 591, 594, 595
ordered list 82
ordering of records 591
ORM (Object-Relational Mapping) 665
OSCON (O'Reilly Open Source
Convention) 30
outer for loop 196, 197
Outline window 819
outset value (border-style property)
119
outsourcing marketplaces 43
Elance 43
Guru.com 43
Oval tool 458
over state 463
overflow boundaries 118
overflow property 118
overlapping text (text-decoration)
106
Overstock.com 42
Overture 9
OWL (Web Ontology Language) 47

P
p element 71
Packages
com.sun.rave.web.ui.appbase
806
com.sun.webui.jsf.component
806, 810
javax.faces.validators 825
javax.servlet.http 800, 842,
844
javax.servlet.jsp 801
javax.servlet.jsp.tagext 801
Packet8 (VoIP) 39, 44
padding 118
padding-bottom property 121
padding for individual sides of an element
121
padding-left property 121
padding property (block-level elements)
121
padding-right property 121
padding-top property 121
page bean 803, 811
Page class 705, 732, 746
Session property 746
Page directive in ASP.NET 703
Page event lifecycle (ASP.NET) 707
Page Navigation 48
Page, Larry 48
Page_Init event handler 705
Page_Load event handler 706
Page_PreInit event handler 706
Page_Unload event handler 707
Pageflakes 43
PageRank 8
PageSize property of a GridView
ASP.NET control 789
PageSize property of a
GridView control 789
paid blog posts 40
PayPerPost 40
ReviewMe 40
SponsoredReviews 40
Paint Behind 474
Paint Inside 474
Paint Selection 474
Palette 814, 817, 820
Pandora 42, 43
Panel element 522
height attribute 522
horizontalAlign attribute 522
title attribute 522
verticalAlign attribute 522
width attribute 522
panels 454
PanTool element 543
paragraph element 71
Parallel element 551
parameter 205
percent (%)

parent 399

parent element 106, 353

parent node 385

parent parameter 537

parent window 279

parent/child relationships between data 352

parentNode property of a DOM node 312, 396

parentNode property of a Node 396

parse 276, 277

parsed character data 353

parsed character data 363

parseFloat function 205, 208, 227

parseInt function 145, 146, 163, 165, 227

parser 350

_parseing_ a SOAP message 288

parsed message 288

parser 350

parse 288

pass-by-reference 243, 244

pass-by-value 243, 244

passing arrays 245

Passing arrays and individual array elements to functions 245

password 478

password_field input 94

password_field method 672

Parse Frames 475

Partial that contains a form used to add a new forum 680

participant in a virtual form 227

participant in a virtual form 209, 212, 214

participatory journalism 17

password 478

password_field input 94

password_field method 672

Patent

social networking 20

Path

property of HttpCookie class 746

Path system variable 525

path to a resource 571

pattern matching 593

Paul.Kedrosky.com/public-loos / 35 payments 43

Google Checkout 43

PayPal 43

PayPal 43

PayPerPost 40

_pay, pic—9 pic = 12 pt_ 109

#PCDATA keyword (DTD) 363

PDML (Product Data Markup Language) 372

people-assisted search 43

percent (8) SQL wildcard character 593

percent sign (%) remainder operator 147

percentage 108

percentFontSize attribute 522

percentWidth attribute 522

Performance Tips overview xxv

performance-based advertising 40

performance-intensive situation 195

Perl-compatible regular expressions 619

permalinks 17

perpetual beta 30

persistent cookie 637

persistent database connection 603

persistent information 834

personalization 737, 833

personalized start page 43

Pew Internet & American Life Project 4, 16, 18, 26

photo sharing 43

Flickr 43

Photobucket 43

PhotoShop Elements 76

PHP:IX, xx, 413, 473, 654

PHP: (PHP: Hypertext Preprocessor) 607

_php extension 608

PHP: Hypertext Preprocessor (PHP) 607

PHP quantifiers 621

PHPNuke 31

PT 261

PI (processing instruction) 378

picture element (pixel) 108

picture.html 76

pixel 76, 458

pixels 458

place holder in an initializer list 237

platform xx

play function 491

playhead 454, 480

plus sign (+) for container elements in

Internet Explorer 353

plus sign (+) occurrence indicator 363

PM 276

PNG (Portable Network Graphics) 76

podcast 25

podcasting 25, 28

POJO (Plain Old Java Object) 898

Policy Aware Web Project 46

polling 519

Polygon Mode of Lasso tool 474

polynomial 519

populating a Data Set 603

Popup Calendar JSF component 872

pop-up window 328

Portability Tips overview xxv

Portable Network Graphics (PNG) 76

Portable Operating System Interface (POSIX) 619

position number 233

position property 111

POSIX extended regular expressions 619

post 677

post request type 626

post request type 89

postback 810

postback event of an ASP.NET page 732

postcrement operator 170, 170

PostgreSQL 586

postincrement operator 170, 172

Pow method of class Math (Java) 944

pow method of Math object 85, 204, 260

power 260

PowerSet (semantic search) 7

Practical Extraction and Report Language (Perl) 608

pre element 140

precedence 172

precedence and associativity of operators 152, 173, 201

predecrement operator 170, 170

predicate 592

preg_match function 619

preincrement operator 170, 172

PreInit event of an ASP.NET Web page 706

preload 485

preloading images 338

premium content 40

“packaged” function 204

preprocess method 810, 816

prerender method 810, 816

presentation 372

presentation logic 575

presentation MathML 372

presentation of a document 68

press a key 346

pressing keys 324

previousSibling property of a Node 396

primary key 587, 590

primary property of a JSF Button 860

principal 183

print function 609

print media type 122, 122, 124

printing on multiple lines with a single statement 138

printing one line with separate statements 136

privacy 19, 21, 23

privacy protection 738, 833

probability 211

Problogger 41

procedural programming language 259

processing instruction (PI) 378

processing instruction target 378

processing instruction value 378

processor 350

Product Data Markup Language (PDML) 372

program 132, 133

program control 132

program development 132

program-construction principles 175

Programmable Web 35, 42

 Mashup Matrix 35

programmer-defined function 205

Programmer-defined function square 205, 206

Programming-defined maximum function 209

Programming languages

Ruby 653

Progress Bar JSF component 872

ProgressBar element 551

Project Properties dialog 901

Projects window 814

prolog (XML) 352

promotion 737

prompt box used on a welcome screen 260

prompt dialog 141, 144, 145, 146, 208, 216

prompt method of window object 143, 144, 145

prompt to a user 143

Properties class 841, 850

get property method 841

setProperty method 841

properties of the Math object 261

Properties
dialog 901

ps:unselectable

PS (Portable Single-Step Simulation) 814
social media 5, 23, 43
Digg 43
Flickr 43
Reddit 43
Wikipedia 43
YouTube 43
social networking 5, 19, 24, 28, 43
Bebo 43
Facebook 43
Gaia Online 43
LinkedIn 43
MySpace 43
Second Life 43
social networking patent 20
social networks
Gaia Online 22
SocialText 15, 44
Software as a Service (SaaS) 29, 43
Google 43
Microsoft Office Live 43
Microsoft Windows Live 43
Salesforce.com 43
37Signals 43
Zoho 43
software asset 260
Software Engineering Observations
overview xxv
solid value (border-style property) 119
Some common escape sequences 140
Some useful global arrays. 623
sort method 259
sort method of an ArrayObject 247, 248
sortable_element method of
Script.aculo.us library 696
Sorting an array with Array method
sort 247
sorting in XSL 384
Sound Mixer stopAll function 491
source attribute 523, 538, 540
Source mode in Visual Web Developer 711
source string 262
source tree (XSLT) 377
source-code form 69
SourceForge 31, 42
Spacer element 525
span 17, 23, 24, 29
span as a generic grouping element 114
span element 114, 806, 812
span XHTML element 704, 708
SPARQL (SPARQL Protocol and RDF
Query Language) 47
special character 79, 80, 621
special characters 261
specificity 106, 124
speech device 84
speech synthesizer 77
splash screen 485
split 268
split 269
split function 288
split method of the String object 263
SplitAndSubstring.html 268
splitting a statement in the middle of an
terminus 142
SponsoredReviews 40
Sprite class (Flash) 496
SQL 586, 587, 591, 597
alias 784
AS keyword 784
DELETE statement 591, 599
FROM clause 591
GROUP BY clause 591
INNER JOIN clause 591, 596
INSERT statement 591, 597
LIKE clause 593, 594
ON clause 596
ORDER BY clause 591, 594, 595
SELECT query 591, 592, 594, 595
SET clause 599
UPDATE statement 591
VALUES clause 597
WHERE clause 592
.sql 602
SQL keyword 591
SQLSchool (www.w3schools.com/
sql) 604
SQL script 602
SQL Server 604
SQL Server Magazine
(www.sqlmag.com) 605
SQL Server website
(www.microsoft.com/sql) 604
SQLCommand class 603
SQLConnection class 603
SqlDataAdapter class 604
SqlDataSource ASP.NET data control
755
SQLite database 567
sqrt 260
SQRT1_2 261
SQRT2 261
square brackets [...] 233
square root 260
square-root symbol (MathML) 374
src attribute 76, 79
src property of an Image object 338
srcElement property (IE) of an event
object 338
stacked control structures 166
stage 454, 455
stage 504
stage.mouseX 503
stage.mouseY 503
"standardized, interchangeable parts" 259
Stanford University 8, 9
start page 820
start tag 70, 349, 356
StartForce (webtop) 29
starting index 269
State element 530
name attribute 530
stateless protocol 738, 834
statement 135, 207
statement terminator 135
states element 530
Static Text JSF component 804, 806, 820
static web page 583
static web pages 583
status property of the
XMLHttpRequest object 422
statusText property of the
XMLHttpRequest 422
steve.museum project 26
stop 497
stop function 469, 477, 491
stops all timers and sends objects into
hidden frame 510
strerror function 619
strike 270
strike method of the String object
263
string 135
string assigned to a variable in a
declaration 262
string comparison 248
string concatenation 144, 163
string constants 261
string data type 610
string literal 135, 261
string manipulation 204
String methods charAt, charCodeAt,
fromCharCode, toLowerCase and
toupperCase 264
String object 261, 262, 264
anchor method 263, 271
charAt method 262, 264, 264
charCodeAt method 262, 264, 265
fixed method 263, 271
fromCharCode method 262, 264, 265
indexOf method 262, 265, 266
lastIndexOf method 263, 265, 266,
267
Link method 263
replace method 263
slice method 263
split method 263
strike method 263
sub method 263
substring method 263
sup method 263
toLowerCase method 263
toUpperCase method 263
String object methods 262
String object's methods that generate
XHTML markup tags 269
string representation of the number 278
string XML Schema data type 369
string's length 264
string-concatenation operator (+) 262
stroke color 458
strong element 73
struck-out text 271
structure 68
structure of a document 101
structured programming 154
Structured Query Language (SQL) 586,
587, 591
StumbleUpon 12, 28, 42, 43
style attribute 102, 176
style class 105, 105
Style element 554
style sheet 70, 353, 805, 814
stylesheet start tag 379
sub 270
sub element 81
sub method of the String object 263
sub initializer list 249
submit input 91
submitter in a virtual form 874
subscribe 81, 249
subscription site 43
trigger attribute 538
triggerEvent attribute 538
trigonometric cosine 260
trigonometric sine 260
true 148
truethe 156
Trulia 10, 43
truncate 163
trust 5
truth table 198, 200
truth table for the && logical AND operator 198
truth table for the || (logical OR) operator 199
truth table for the logical negation operator 199
try block 421
try keyword 421
Tucows 31, 42
tween 468
24-hour clock format 276
24 Hours on Craigslist 15
Twitter 23, 42
two-dimensional array 249, 250
type 104
type = "text/javascript" 134
type attribute 90, 134
type attribute in a processing instruction 378
type casting 612
type conversion 610
type property of an event object 333
type XML Schema attribute 368
type-ahead 428
TypePad 18

U
Ubuntu 31
Ubuntu Live conference 7
Uri: static Text JSF element 868
Uri: table JSF element 868
Uri: tableRowGroup JSF element 868
UI: ViewRoot class 805
ul (unordered list) element 81
ul element 81
unary operator 170, 199
decrement (--) 170
increment (+++) 170
unbounded value 369
undefined 147
underline value (text-decorating) 106
underscore (_) SQL wildcard character 593
unescape function 227, 287
Unicode 261, 264
Unicode value 264
Uniform Resource Identifier (URI) 360, 570
Uniform Resource Locator (URL) 360, 570
Uniform Resource Name (URN) 360, 570
unique session ID of an ASP.NET client 750
United States Department of Defense 37
UNIX 581
UnLoad event of an ASP.NET page 707
unordered list 81
Unordered list containing hyperlinks. 81
unordered list element (ul) 81
up state 463
UPDATE SQL statement 591, 598
UpdateCommand property of a SqlDataSource ASP.NET control 760
UpdatePanel control 797
uppercase letters 136, 142
URI (Uniform Resource Identifier) 360, 570
URL (Uniform Resource Locator) 360, 570
rewriting 738, 834
URL attribute 538
URL mapping 663
URL parameter 682
url(fileLocation) 115
URLRequest object 543
URN (Uniform Resource Name) 360
user agent 106
user input 88
user interaction 324
user interface 757
user style sheet 126
User style sheet applied with em measurement 129
User style sheet applied with pt measurement 128
User style sheet in Internet Explorer 7 127
user styles 126, 128, 129
UserController provides validation functionality for the table 671
user-defined types 259, 369
user-generated content 5, 13, 14, 23, 28
Using a FilterFunction for a live search feature 540
Using a Style element and inline styles 554
using cookies to store user identification data 285
using equality and relational operators 149
Using inline styles 102
using internal hyperlinks to make pages more navigable 95
using meta to provide keywords and a description 98
using PHP’s arithmetic operators 613
Using States to dynamically modify a user interface 527
using String object methods split and substring 268
Using the break statement in a for statement 193
Using the continue statement in a for statement 194
Using the do...while repetition statement 191
using the string-comparison operators 618
Using the switch multiple-selection statement 186
using the window object to create and modify child windows 279
using XPath to locate nodes in an XML document 398
UTC (Coordinated Universal Time) 272, 275, 276

V

valid attribute 538
valid XML document 350, 362
validate 672
validates_format_of 675
validates_presence_of method 675
validating an XML document with Microsoft’s XML Validator 357
validating parser 38
validating XML parser 350
validation 825
validation control 726
validation service 72
ValidationExpression property of a
RegularExpressionValidator control 730
ValidationExpression property of a
Validator 730
validator 726
Validator element
source attribute 538
trigger attribute 538
triggerEvent attribute 538
valid attribute 538
validator.w3.org 72
ValidatorCalloutExtender control 797
ValidatorException class 832, 833
Validators

Double Range 825
Length 825, 830
Long Range 825

validators 675
value attribute 91, 522
value of an array element 233
value of an attribute 71
value property of a text field 221
value property of an Attr object 397
Value property of a HttpCookie class 745
ValueClick 42
valueOf 273
VALUES 598
values of n (company) 29
VALUES SQL clause 597
Vander Wal, Thomas 26
var keyword 142, 207
var to declare variables is not required in JavaScript 142
variable 142
variable name 142
variable variables 647
variable’s scope 180
various markup languages derived from
XML 376
VBox element 522
hideEffect attribute 551
showEffect attribute 551
vector graphic 458
vectors 458
VeoH 24
verbs in a system specification 259
verify 677

version attribute (XSL) 379
version in xml declaration 351
vertical and horizontal positioning 115
vertical search engines 7, 10, 44
Kayak 44
Koders 44
Krugle 44
SideStep 44
Yahoo! Fare Finder 44
vertical spacing 177
verticalAlign attribute 522
verticalAxis element 566
VH1 24
vi text editor 69
Viacom 23, 24
Vibrant Media 40, 42
video blogging 18

View In Browser command in Visual Web Developer 715
view states 527
View that displays a list of employees 669
View that displays the name of the server 664
__VIEWSTATE hidden input 732
viral effect 23
virtual directory 571, 577, 579, 954
Virtual Directory Creation Wizard 577
virtual forms 863, 874
participant 874
submitter 874
virtual worlds 5, 22, 44
Habbo 44
Second Life 44
There 44
virtual worlds monetization 41
Visible property of an ASP.NET Web control 727
Visual Editor 804, 813
visual inheritance 769
Visual Web Developer
ASP.NET Web Service project 954
Visual Web Pack (Netbeans) 799
visual_effect method of
Script.aculo.us Library 692
duration parameter 692
queue parameter 693
end value 693
vlogging 18
vocabulary (XML) 349
VoiceXML 349, 376
www.voicexml.org 376
VoIP 39, 44
Lingo 44
Packet8 44
Skype 44
Vonage 44
Vonage (VoIP) 39, 44
VS1der element 522
change attribute 522
labels attribute 523
liveDragging attribute 522
maximum attribute 522
minimum attribute 522
showDataTip attribute 524
snapInterval attribute 524
tickInterval attribute 524
value attribute 522

W

W3C (World Wide Web Consortium) 38, 47, 68, 348
Weather Forecast application in Flex 2
546
Web 1.0 4
Web 2.0 3
Web 2.0 Expo 7
Web 2.0 Summit 4, 7
Web 2.0 software 44
Adobe 44
Microsoft 44
Six Apart 44
37Signals 44
web analytics 44
Alexa 44
ClickTracks 44
Google Analytics 44
WebSideStory 44
WebTrends 44
web and mobile messaging 44
web application
Ajax 414
traditional 413
Web application development 701
web application development 709
web application framework 802
Web Application project 895
web as a platform 29, 30
web-based application 68
web conferencing 44
DimDim 44
GoToMeeting 44
WebEx 44
Web control 701
Web Form 701, 748
web framework 653
web of meaning 46
Web Ontology Language (OWL) 47
web page 68
web server x, 69, 88, 570, 571, 635, 663
web service 892
implemented as a class 894
publishing a web service 900
Web Service Description Language (WSDL) 904
web service reference 905
Web services
@Resource annotation 919
@WebMethod annotation 899
@WebParam annotation 899
@WebService annotation 899
adding a web service reference to an
application 905
Build Project option in Netbeans 900
Clean and Build Project option in
Netbeans 900
Clean Project option in Netbeans 900
Client desktop application for the
HugeInteger web service 909
client-side artifacts 905, 906
consuming a web service 894, 904
Deploy Project option in Netbeans
900
deploying a web service 900
XML 412, 800
attribute 356
attribute value 356
call element 353
container element 353
declaration 351, 355
element 349
empty element 358
end tag 349
mark up 351
node 354
parent element 353
prolog 352
root 354
root element 349
start tag 349
vocabulary 349
XML (Extensible Markup Language) xx, 32, 38, 46, 348
XML declaration 804
XML document containing book information 381
XML Document Object Model 386
XML document that describes various sports 377, 401
XML document using the laptop element defined in computer.xsd 371
XML document using the <laptop> element defined in computer.xsd 371
XML DOM 387
XML element name requirements 352
XML file containing contacts 538
.xm1 file extension 360
XML instance document 370, 371
xmlns namespace prefix 359
XML namespaces demonstration 359
XML parser 350
XML Path Language (XPath) 376, 398
XML processor 350
XML Resource Center (www.deitel.com/XML/) 396
XML Schema 361, 365, 369
complex types 369
simple types 369
XML Schema document defining simple and complex types 370
XML Schema document for book.xm1 366
XML Schema URI (http://www.w3.org/2001/XMLSchema) 367
XML serialization 939
XML used to mark up an article 351
XML Validator 356
XML Validator displaying an error message 364
XML vocabularies 38
Chemical Markup Language (CML) 376
Extensible User Interface Language (XUL) 376
Geography Markup Language (GML) 376
Research Information Exchange Markup Language (RXML) 376
Synchronized Multimedia Integration Language (SMIL) 376
VoiceXML 376
XML Working Group of the W3C 348
XmlDataSource ASP.NET data control 723
XMLHttpRequest 32
XMLHttpRequest object 412, 417, 418, 421
abort method 423
GET method 422
get AllResponseHeaders method 423
getResponseHeader method 423
onReadyStateChanged property 421, 422
open method 422
properties and methods 421
readyState property 422
responseText property 421, 422
responseXML property 422
send method 422
setRequestHeader method 422
status property 422
statusText property 422
xmlns attribute in XML 360
xmlns attributes 804
XPath 377
XPath (XML Path Language) 376, 398
XPath expression 398
XPathResult object 401
.xsd file name extension 365
XSL (Extensible Stylesheet Language) 351, 361, 376
XSL document that transforms sorting.xml into XHTML 382
.xs1 filename extension 379
XSL-FO (XSL Formatting Objects) 376
XSL Formatting Objects (XSL-FO) 376
XSL style sheet 377, 384
XSL template 380
xs1 template element 380
XSL transformation 405
XSL Transformations (XSLT) 377
XSL variable 384
xs1: for-each element 380
xml: output element 380
xml: value-of element 380
XSLT processor 377
XSLT that creates elements and attributes in an XHTML document 379
XSLTProcessor object (Firefox) 409
XSS (cross-site scripting) 418
XUL (Extensible User Interface Language) 372, 376
xx-large relative font size 105
xx-small relative font size 105
Y
Yahoo! 7, 9, 12, 27, 34, 42, 43
Classifieds 42
Fare Finder 44
Pipes 35
Publisher Network 40
Search Marketing 41
Video 42
Yahoo! Mail 44
Yahoo! Map 531
Yahoo! Maps API 540
YahooMapService 537
YahooMapService element 551
Yang, Jerry 9
yField attribute 566
yield 664
YouTube 5, 13, 23, 42, 43
Z
z-index property 113
Zend Engine 607
Zend Technologies (www.zend.com) 43, 607
Zepheira 46
zero-based counting 179
zeroth element of an array 233
Zillow 10, 43
Zoho 43
Zoom effect 551
Zoom tool 458