

Fourth Edition

INTERNET & WORLD WIDE WEB HOW TO PROGRAM

WEB 2.0

Ajax

Rich Internet
Applications

HONK BEEP

Skype
TomTom

00000

SECOND
LIFE

X M L

I
E
7

JAVASCRIPT

WEB
SERVICES

MYSQL

Silverlight

XHTML
Flex

Script.aculo.us

Twittering

JavaServer™ Faces

Ruby on Rails

ASP.NET

JavaServer™ Faces

Open
Source

RSS

LONG
TAIL

PHP

PERL

ASP

VB.NET

C#

DOM

HTML



DEITEL® P.J. DEITEL
H.M. DEITEL

Internet & World Wide Web HOW TO PROGRAM

FOURTH EDITION

Deitel® Series Page

How To Program Series

Internet & World Wide Web How to Program, 4/E

Java How to Program, 7/E

C++ How to Program, 6/E

C How to Program, 5/E

Visual Basic® 2005 How to Program, 3/E

Visual C#® 2005 How to Program, 2/E

Small Java™ How to Program, 6/E

Small C++ How to Program, 5/E

Advanced Java™ 2 Platform How to Program

XML How to Program

Visual C++® .NET How to Program

Perl How to Program

Python How to Program

Simply Series

Simply C++: An Application-Driven
Tutorial Approach

Simply C#: An Application-Driven
Tutorial Approach

Simply Java™ Programming: An
Application-Driven Tutorial
Approach

Simply Visual Basic® 2005, 2/E: An
Application-Driven Tutorial
Approach

SafariX Web Books

www.deitel.com/books/SafariX.html

C++ How to Program, 5/E & 6/E

Small C++ How to Program, 5/E

Java How to Program, 6/E & 7/E

Small Java How to Program, 6/E

Simply C++: An Application-Driven
Tutorial Approach

Visual Basic 2005 How to Program, 3/E

Simply Visual Basic 2005: An Application-
Driven Tutorial Approach, 2/E

Visual C# 2005 How to Program, 2/E

To follow the Deitel publishing program, please register for the free *Deitel® Buzz Online* e-mail newsletter at:

www.deitel.com/newsletter/subscribe.html

To communicate with the authors, send e-mail to:

deitel@deitel.com

For information on corporate on-site seminars offered by Deitel & Associates, Inc. worldwide, visit:

www.deitel.com/training/

or write to

deitel@deitel.com

For continuing updates on Prentice Hall/Deitel publications visit:

www.deitel.com

www.prenhall.com/deitel

www.InformIT.com/deitel

Check out our Resource Centers for valuable web resources that will help you master C++, other important programming languages, software and Web 2.0 topics:

www.deitel.com/ResourceCenters.html

Library of Congress Cataloging-in-Publication Data
On file

Vice President and Editorial Director, ECS: ***Marcia J. Horton***

Associate Editor: ***Carole Snyder***

Supervisor/Editorial Assistant: ***Dolores Mars***

Director of Team-Based Project Management: ***Vince O'Brien***

Senior Managing Editor: ***Scott Disanno***

Managing Editor: ***Robert Engelhardt***

Production Editor: ***Marta Samsel***

A/V Production Editor: ***Greg Dulles***

Art Studio: ***Artworks, York, PA***

Art Director: ***Kristine Carney***

Cover Design: ***Abbey S. Deitel, Harvey M. Deitel, Francesco Santalucia, Kristine Carney***

Interior Design: ***Harvey M. Deitel, Kristine Carney***

Manufacturing Manager: ***Alexis Heydt-Long***

Manufacturing Buyer: ***Lisa McDowell***

Director of Marketing: ***Margaret Waples***



© 2008 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or to the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks and registered trademarks. Where those designations appear in this book, and Prentice Hall and the authors were aware of a trademark claim, the designations have been printed in initial caps or all caps. All product names mentioned remain trademarks or registered trademarks of their respective owners.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-175242-1

Pearson Education Ltd., London

Pearson Education Australia Pty. Ltd., Sydney

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd., Hong Kong

Pearson Education Canada, Inc., Toronto

Pearson Educación de Mexico, S.A. de C.V.

Pearson Education-Japan, Tokyo

Pearson Education Malaysia, Pte. Ltd.

Pearson Education, Inc., Upper Saddle River, New Jersey

Internet & World Wide Web HOW TO PROGRAM

FOURTH EDITION

P. J. Deitel

Deitel & Associates, Inc.

H. M. Deitel

Deitel & Associates, Inc.



Upper Saddle River, New Jersey 07458

Trademarks

DEITEL, the double-thumbs-up bug and DIVE INTO are registered trademarks of Deitel & Associates, Inc.

Adobe, Dreamweaver, Flex and Flash are either registered trademarks or trademarks of Adobe Systems, Inc.

Apache is a trademark of The Apache Software Foundation.

CSS, DOM, XHTML and XML are registered trademarks of the World Wide Web Consortium.

Del.icio.us and Flickr are trademarks of Yahoo! Inc.

Digg is a trademark of Digg Inc.

Firefox is a registered trademark of the Mozilla Foundation.

Google is a trademark of Google, Inc.

JavaScript, Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Microsoft, Internet Explorer, Silverlight and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

MySpace is a registered trademark of MySpace.com.

UNIX is a registered trademark of The Open Group.

Ruby on Rails is a trademark of David Heinemeier Hansson.

MySQL is a registered trademark of MySQL AB.

Second Life is a registered trademark of Linden Research, Inc.

Skype is a trademark of eBay, Inc.

Web 2.0 is a service mark of CMP Media.

Wikipedia is a registered trademark of WikiMedia.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.

To Tim O'Reilly and John Battelle:

*For your extraordinary efforts in
bringing Web 2.0 to the world through
the annual Web 2.0 Summit conference
and so much more.*

Paul and Harvey Deitel

Deitel Resource Centers

Our Resource Centers focus on the vast amounts of free content available online. Start your search here for resources, downloads, tutorials, documentation, books, e-books, journals, articles, blogs, RSS feeds and more on many of today's hottest programming and technology topics. For the most up-to-date list of our Resource Centers, visit:

www.deitel.com/ResourceCenters.html

Let us know what other Resource Centers you'd like to see! Also, please register for the free *Deitel® Buzz Online* e-mail newsletter at:

www.deitel.com/newsletter/subscribe.html

Programming	Web3D Technologies	Attention Economy	Wikis
.NET	XHTML	Blogging	Internet Business
.NET 3.0	XML	Building Web	Affiliate Programs
Adobe Flex	Software	Communities	Google Adsense
Ajax	Apache	Community-	Google Analytics
Apex On-Demand Programming Language	DotNetNuke (DNN)	Generated Content	Google Services
ASP.NET	Eclipse	Google AdSense	Internet Advertising
ASP.NET Ajax	Firefox	Google Analytics	Internet Business
C	Flash CS3 (Flash 9)	Google Base	Initiative
C#	Internet Explorer 7	Google Services	Internet Public
C++	Linux	Google Video	Relations
C++ Boost Libraries	MySQL	Google Web Toolkit	Link Building
C++ Game Programming	Open Source	Internet Advertising	Podcasting
Code Search Engines and Code Sites	Search Engines	Internet Business	Search Engine
Computer Game Programming	Web Servers	Initiative	Optimization
CSS 2.1	Wikis	Internet Public	Sitemaps
Dojo Toolkit	Windows Vista	Relations	Web Analytics
Flash 9	Microsoft	Internet Video	Website Monetization
Flex	.NET	Joost	Open Source
Java	.NET 3.0	Link Building	Apache
Java Certification and Assessment Testing	ASP.NET	Location-Based	DotNetNuke (DNN)
Java Design Patterns	ASP.NET Ajax	Services	Eclipse
Java EE 5	C#	Mashups	Firefox
Java SE 6	DotNetNuke (DNN)	Microformats	Linux
JavaFX	Internet Explorer 7	Podcasting	MySQL
JavaScript	Silverlight	Recommender	Open Source
JSON	Visual Basic	Systems	Perl
OpenGL	Visual C++	RSS	PHP
Perl	Windows Vista	Search Engine	Python
PHP	Java	Optimization	Ruby
Programming Projects	Java Certification and Assessment Testing	Search Engines	Ruby on Rails
Python	Java Design Patterns	Selling Digital Content	Other Topics
Ruby	Java EE 5	Skype	Computer Games
Ruby on Rails	Java SE 6	Social Media	Computing Jobs
Silverlight	JavaFX	Social Networking	Gadgets and Gizmos
Visual Basic	Web 2.0 and Internet Business	Software as a Service (SaaS)	Sudoku
Visual C++	Affiliate Programs	Virtual Worlds	
Web Services	Alert Services	Web 2.0	
		Web 3.0	
		Web Analytics	
		Website Monetization	
		Widgets	

Contents

Preface	xxi
Before You Begin	xxxii
Part I: Introduction	I
1 Introduction to Computers and the Internet	2
1.1 Introduction	3
1.2 What Is a Computer?	7
1.3 Computer Organization	7
1.4 Machine Languages, Assembly Languages and High-Level Languages	8
1.5 History of the Internet and World Wide Web	10
1.6 World Wide Web Consortium (W3C)	11
1.7 Web 2.0	12
1.8 Personal, Distributed and Client/Server Computing	13
1.9 Hardware Trends	14
1.10 Key Software Trend: Object Technology	15
1.11 JavaScript: Object-Based Scripting for the Web	16
1.12 Browser Portability	17
1.13 C, C++ and Java	17
1.14 BASIC, Visual Basic, Visual C++, C# and .NET	18
1.15 Software Technologies	18
1.16 Notes about <i>Internet & World Wide Web How to Program</i> , 4/e	20
1.17 Web Resources	20
2 Web Browser Basics: Internet Explorer and Firefox	28
2.1 Introduction to the Internet Explorer 7 and Firefox 2 Web Browsers	29
2.2 Connecting to the Internet	29
2.3 Internet Explorer 7 and Firefox 2 Features	30
2.4 Customizing Browser Settings	36
2.5 Searching the Internet	37
2.6 Keeping Track of Your Favorite Sites	40
2.7 File Transfer Protocol (FTP)	40
2.8 Online Help	42

2.9	Other Web Browsers	44
2.10	Wrap-Up	44
2.11	Web Resources	44

3 [**Dive Into® Web 2.0**](#) **50**

3.1	Introduction	51
3.2	What Is Web 2.0?	52
3.3	Search	55
3.4	Content Networks	60
3.5	User-Generated Content	61
3.6	Blogging	64
3.7	Social Networking	67
3.8	Social Media	71
3.9	Tagging	74
3.10	Social Bookmarking	76
3.11	Software Development	77
3.12	Rich Internet Applications (RIAs)	80
3.13	Web Services, Mashups, Widgets and Gadgets	82
3.14	Location-Based Services	85
3.15	XML, RSS, Atom, JSON and VoIP	87
3.16	Web 2.0 Monetization Models	88
3.17	Web 2.0 Business Models	89
3.18	Future of the Web	93
3.19	Wrap-Up	96
3.20	Where to Go for More Web 2.0 Information	97
3.21	Web 2.0 Bibliography	97
3.22	Web 2.0 Glossary	104

Part 2: The Ajax Client **117**

4 [**Introduction to XHTML**](#) **118**

4.1	Introduction	119
4.2	Editing XHTML	120
4.3	First XHTML Example	120
4.4	W3C XHTML Validation Service	123
4.5	Headings	123
4.6	Linking	124
4.7	Images	127
4.8	Special Characters and Horizontal Rules	130
4.9	Lists	132
4.10	Tables	135
4.11	Forms	139
4.12	Internal Linking	146
4.13	meta Elements	149
4.14	Wrap-Up	150
4.15	Web Resources	151

5	Cascading Style Sheets™ (CSS)	159
5.1	Introduction	160
5.2	Inline Styles	161
5.3	Embedded Style Sheets	162
5.4	Conflicting Styles	165
5.5	Linking External Style Sheets	168
5.6	Positioning Elements	170
5.7	Backgrounds	174
5.8	Element Dimensions	176
5.9	Box Model and Text Flow	177
5.10	Media Types	181
5.11	Building a CSS Drop-Down Menu	184
5.12	User Style Sheets	186
5.13	CSS 3	189
5.14	Wrap-Up	190
5.15	Web Resources	190
6	JavaScript: Introduction to Scripting	197
6.1	Introduction	198
6.2	Simple Program: Displaying a Line of Text in a Web Page	199
6.3	Modifying Our First Program	202
6.4	Obtaining User Input with <code>prompt</code> Dialogs	207
6.4.1	Dynamic Welcome Page	207
6.4.2	Adding Integers	211
6.5	Memory Concepts	214
6.6	Arithmetic	215
6.7	Decision Making: Equality and Relational Operators	217
6.8	Wrap-Up	223
6.9	Web Resources	223
7	JavaScript: Control Statements I	234
7.1	Introduction	235
7.2	Algorithms	235
7.3	Pseudocode	236
7.4	Control Structures	236
7.5	<code>if</code> Selection Statement	239
7.6	<code>if...else</code> Selection Statement	240
7.7	<code>while</code> Repetition Statement	245
7.8	Formulating Algorithms: Counter-Controlled Repetition	246
7.9	Formulating Algorithms: Sentinel-Controlled Repetition	250
7.10	Formulating Algorithms: Nested Control Statements	256
7.11	Assignment Operators	260
7.12	Increment and Decrement Operators	261
7.13	Wrap-Up	265
7.14	Web Resources	265

8	JavaScript: Control Statements II	278
8.1	Introduction	279
8.2	Essentials of Counter-Controlled Repetition	279
8.3	for Repetition Statement	281
8.4	Examples Using the for Statement	286
8.5	switch Multiple-Selection Statement	290
8.6	do...while Repetition Statement	295
8.7	break and continue Statements	297
8.8	Labeled break and continue Statements	299
8.9	Logical Operators	302
8.10	Summary of Structured Programming	306
8.11	Wrap-Up	311
8.12	Web Resources	311
9	JavaScript: Functions	321
9.1	Introduction	322
9.2	Program Modules in JavaScript	322
9.3	Programmer-Defined Functions	324
9.4	Function Definitions	324
9.5	Random Number Generation	329
9.6	Example: Game of Chance	334
9.7	Another Example: Random Image Generator	341
9.8	Scope Rules	342
9.9	JavaScript Global Functions	345
9.10	Recursion	346
9.11	Recursion vs. Iteration	349
9.12	Wrap-Up	350
9.13	Web Resources	351
10	JavaScript: Arrays	362
10.1	Introduction	363
10.2	Arrays	363
10.3	Declaring and Allocating Arrays	365
10.4	Examples Using Arrays	365
10.5	Random Image Generator Using Arrays	373
10.6	References and Reference Parameters	374
10.7	Passing Arrays to Functions	375
10.8	Sorting Arrays	378
10.9	Searching Arrays: Linear Search and Binary Search	380
10.10	Multidimensional Arrays	386
10.11	Building an Online Quiz	390
10.12	Wrap-Up	393
10.13	Web Resources	393
11	JavaScript: Objects	403
11.1	Introduction	404

11.2	Introduction to Object Technology	404
11.3	Math Object	407
11.4	String Object	409
11.4.1	Fundamentals of Characters and Strings	409
11.4.2	Methods of the String Object	410
11.4.3	Character-Processing Methods	411
11.4.4	Searching Methods	413
11.4.5	Splitting Strings and Obtaining Substrings	415
11.4.6	XHTML Markup Methods	417
11.5	Date Object	419
11.6	Boolean and Number Objects	425
11.7	document Object	426
11.8	window Object	427
11.9	Using Cookies	432
11.10	Final JavaScript Example	436
11.11	Using JSON to Represent Objects	444
11.12	Wrap-Up	445
11.13	Web Resources	445

12 Document Object Model (DOM): Objects and Collections **458**

12.1	Introduction	459
12.2	Modeling a Document: DOM Nodes and Trees	459
12.3	Traversing and Modifying a DOM Tree	462
12.4	DOM Collections	473
12.5	Dynamic Styles	475
12.6	Summary of the DOM Objects and Collections	481
12.7	Wrap-Up	482
12.8	Web Resources	483

13 JavaScript: Events **487**

13.1	Introduction	488
13.2	Registering Event Handlers	488
13.3	Event <code>onload</code>	491
13.4	Event <code>onmousemove</code> , the <code>event</code> Object and <code>this</code>	492
13.5	Rollovers with <code>onmouseover</code> and <code>onmouseout</code>	497
13.6	Form Processing with <code>onfocus</code> and <code>onblur</code>	502
13.7	More Form Processing with <code>onsubmit</code> and <code>onreset</code>	505
13.8	Event Bubbling	507
13.9	More Events	509
13.10	Wrap-Up	510
13.11	Web Resources	511

14 XML and RSS **515**

14.1	Introduction	516
14.2	XML Basics	516

14.3	Structuring Data	519
14.4	XML Namespaces	526
14.5	Document Type Definitions (DTDs)	529
14.6	W3C XML Schema Documents	533
14.7	XML Vocabularies	540
14.7.1	MathML™	540
14.7.2	Other Markup Languages	544
14.8	Extensible Stylesheet Language and XSL Transformations	544
14.9	Document Object Model (DOM)	553
14.10	RSS	570
14.11	Wrap-Up	578
14.12	Web Resources	578

15 Ajax-Enabled Rich Internet Applications **588**

15.1	Introduction	589
15.2	Traditional Web Applications vs. Ajax Applications	590
15.3	Rich Internet Applications (RIAs) with Ajax	592
15.4	History of Ajax	594
15.5	“Raw” Ajax Example Using the XMLHttpRequest Object	594
15.6	Using XML and the DOM	600
15.7	Creating a Full-Scale Ajax-Enabled Application	604
15.8	Dojo Toolkit	617
15.9	Wrap-Up	626
15.10	Web Resources	627

Part 3: Rich Internet Application Client Technologies **635**

16 Adobe® Flash® CS3 **636**

16.1	Introduction	637
16.2	Flash Movie Development	638
16.3	Learning Flash with Hands-On Examples	640
16.3.1	Creating a Shape with the Oval Tool	642
16.3.2	Adding Text to a Button	644
16.3.3	Converting a Shape into a Symbol	645
16.3.4	Editing Button Symbols	647
16.3.5	Adding Keyframes	648
16.3.6	Adding Sound to a Button	649
16.3.7	Verifying Changes with Test Movie	650
16.3.8	Adding Layers to a Movie	651
16.3.9	Animating Text with Tweening	652
16.3.10	Adding a Text Field	654
16.3.11	Adding ActionScript	655
16.4	Publishing Your Flash Movie	656

16.5	Creating Special Effects with Flash	657
16.5.1	Importing and Manipulating Bitmaps	657
16.5.2	Creating an Advertisement Banner with Masking	658
16.5.3	Adding Online Help to Forms	661
16.6	Creating a Website Splash Screen	669
16.7	ActionScript	675
16.8	Wrap-Up	675
16.9	Web Resources	676

17 **Adobe® Flash® CS3: Building an Interactive Game**

683

17.1	Introduction	684
17.2	Object-Oriented Programming	686
17.3	Objects in Flash	686
17.4	Cannon Game: Preliminary Instructions and Notes	688
17.5	Adding a Start Button	689
17.6	Creating Moving Objects	689
17.7	Adding the Rotating Cannon	694
17.8	Adding the Cannonball	696
17.9	Adding Sound and Text Objects to the Movie	699
17.10	Adding the Time Counter	700
17.11	Detecting a Miss	702
17.12	Adding Collision Detection	703
17.13	Finishing the Game	706
17.14	ActionScript 3.0 Elements Introduced in This Chapter	707

18 **Adobe® Flex™ 2 and Rich Internet Applications**

711

18.1	Introduction	712
18.2	Flex Platform Overview	713
18.3	Creating a Simple User Interface	714
18.4	Accessing XML Data from Your Application	725
18.5	Interacting with Server-Side Applications	739
18.6	Customizing Your User Interface	748
18.7	Creating Charts and Graphs	752
18.8	Connection-Independent RIAs on the Desktop: Adobe Integrated Runtime (AIR)	760
18.9	Flex 3 Beta	761
18.10	Wrap-Up	761
18.11	Web Resources	761

19 **Microsoft® Silverlight™ and Rich Internet Applications**

770

19.1	Introduction	771
19.2	Platform Overview	772

19.3	Silverlight 1.0 Installation and Overview	772
19.4	Creating a Movie Viewer for Silverlight 1.0	773
19.4.1	Creating a User Interface In XAML Using Expression Blend	773
19.4.2	Using Storyboards	775
19.4.3	Creating Controls	776
19.4.4	Using JavaScript for Event Handling and DOM Manipulation	785
19.5	Embedding Silverlight in HTML	793
19.6	Silverlight Streaming	794
19.7	Silverlight 1.1 Installation and Overview	798
19.8	Creating a Cover Viewer for Silverlight 1.1 Alpha	798
19.9	Building an Application with Third-Party Controls	807
19.10	Consuming a Web Service	812
19.10.1	Consuming the HugeInteger Web Service	815
19.11	Silverlight Demos, Games and Web Resources	820
19.12	Wrap-Up	823

20 **Adobe® Dreamweaver® CS3** **830**

20.1	Introduction	831
20.2	Adobe Dreamweaver CS3	831
20.3	Text Styles	835
20.4	Images and Links	841
20.5	Symbols and Lines	842
20.6	Tables	843
20.7	Forms	846
20.8	Scripting in Dreamweaver	849
20.9	Spry Framework for Creating Ajax Applications	850
20.10	Site Management	852
20.11	Wrap-Up	852
20.12	Web Resources	852

Part 4: Rich Internet Application Server Technologies **857**

21 **Web Servers (IIS and Apache)** **858**

21.1	Introduction	859
21.2	HTTP Transactions	859
21.3	Multitier Application Architecture	863
21.4	Client-Side Scripting versus Server-Side Scripting	864
21.5	Accessing Web Servers	865
21.6	Microsoft Internet Information Services (IIS)	865
21.6.1	Microsoft Internet Information Services (IIS) 5.1 and 6.0	865
21.6.2	Microsoft Internet Information Services (IIS) 7.0	868
21.7	Apache HTTP Server	870
21.8	Requesting Documents	872
21.9	Web Resources	873

22 Database: SQL, MySQL, ADO.NET 2.0 and Java DB	879
22.1 Introduction	880
22.2 Relational Databases	881
22.3 Relational Database Overview: A books Database	882
22.4 SQL	885
22.4.1 Basic SELECT Query	885
22.4.2 WHERE Clause	886
22.4.3 ORDER BY Clause	888
22.4.4 Combining Data from Multiple Tables: INNER JOIN	890
22.4.5 INSERT Statement	891
22.4.6 UPDATE Statement	892
22.4.7 DELETE Statement	893
22.5 MySQL	894
22.6 Instructions for Installing MySQL	894
22.7 Instructions for Setting Up a MySQL User Account	895
22.8 Creating a Database in MySQL	896
22.9 ADO.NET Object Model	896
22.10 Java DB/Apache Derby	898
22.11 Wrap-Up	898
22.12 Web Resources	898
23 PHP	905
23.1 Introduction	906
23.2 PHP Basics	907
23.3 String Processing and Regular Expressions	917
23.3.1 Comparing Strings	917
23.3.2 Regular Expressions	918
23.4 Form Processing and Business Logic	922
23.5 Connecting to a Database	929
23.6 Using Cookies	933
23.7 Dynamic Content	939
23.8 Operator Precedence Chart	948
23.9 Wrap-Up	950
23.10 Web Resources	950
24 Ruby on Rails	956
24.1 Introduction	957
24.2 Ruby	957
24.3 Rails Framework	964
24.4 ActionController and ActionView	966
24.5 A Database-Driven Web Application	969
24.6 Case Study: Message Forum	974
24.6.1 Logging In and Logging Out	974
24.6.2 Embellishing the Models	978

24.6.3	Generating Scaffold Code	980
24.6.4	Forum Controller and Forum Views	981
24.6.5	Message Controller and Message Views	986
24.6.6	Ajax-Enabled Rails Applications	990
24.7	Script.aculo.us	995
24.8	Wrap-Up	1003
24.9	Web Resources	1003

25 ASP.NET 2.0 and ASP.NET Ajax 1009

25.1	Introduction	1010
25.2	Creating and Running a Simple Web Form Example	1011
25.2.1	Examining an ASPX File	1012
25.2.2	Examining a Code-Behind File	1014
25.2.3	Relationship Between an ASPX File and a Code-Behind File	1015
25.2.4	How the Code in an ASP.NET Web Page Executes	1015
25.2.5	Examining the XHTML Generated by an ASP.NET Application	1016
25.2.6	Building an ASP.NET Web Application	1017
25.3	Web Controls	1025
25.3.1	Text and Graphics Controls	1025
25.3.2	AdRotator Control	1030
25.3.3	Validation Controls	1035
25.4	Session Tracking	1046
25.4.1	Cookies	1047
25.4.2	Session Tracking with <code>HttpSessionState</code>	1055
25.5	Case Study: Connecting to a Database in ASP.NET	1062
25.5.1	Building a Web Form That Displays Data from a Database	1063
25.5.2	Modifying the Code-Behind File for the Guestbook Application	1072
25.6	Case Study: Secure Books Database Application	1074
25.6.1	Examining the Completed Secure Books Database Application	1074
25.6.2	Creating the Secure Books Database Application	1078
25.7	ASP.NET Ajax	1102
25.8	Wrap-Up	1106
25.9	Web Resources	1107

26 JavaServer™ Faces Web Applications 1118

26.1	Introduction	1119
26.2	Java Web Technologies	1120
26.2.1	Servlets	1120
26.2.2	JavaServer Pages	1121
26.2.3	JavaServer Faces	1122
26.2.4	Web Technologies in Netbeans	1122
26.3	Creating and Running a Simple Application in Netbeans	1123
26.3.1	Examining a JSP File	1124
26.3.2	Examining a Page Bean File	1126
26.3.3	Event-Processing Life Cycle	1130
26.3.4	Relationship Between the JSP and Page Bean Files	1131

26.3.5	Examining the XHTML Generated by a Java Web Application	1131
26.3.6	Building a Web Application in Netbeans	1133
26.4	JSF Components	1140
26.4.1	Text and Graphics Components	1140
26.4.2	Validation Using Validator Components and Custom Validators	1145
26.5	Session Tracking	1153
26.5.1	Cookies	1154
26.5.2	Session Tracking with the SessionBean Object	1166
26.6	Wrap-Up	1176
26.7	Web Resources	1177

27 Ajax-Enabled JavaServer™ Faces Web Applications **1187**

27.1	Introduction	1188
27.2	Accessing Databases in Web Applications	1189
27.2.1	Building a Web Application That Displays Data from a Database	1189
27.2.2	Modifying the Page Bean File for the AddressBook Application	1198
27.3	Ajax-Enabled JSF Components	1201
27.4	AutoComplete Text Field and Virtual Forms	1202
27.4.1	Configuring Virtual Forms	1203
27.4.2	JSP File with Virtual Forms and an AutoComplete Text Field	1204
27.4.3	Providing Suggestions for an AutoComplete Text Field	1208
27.5	Google Maps Map Viewer Component	1210
27.5.1	Obtaining a Google Maps API Key	1211
27.5.2	Adding a Map Viewer Component to a Page	1211
27.5.3	JSP File with a Map Viewer Component	1212
27.5.4	Page Bean That Displays a Map in the Map Viewer Component	1216
27.6	Wrap-Up	1219
27.7	Web Resources	1220

28 Web Services **1225**

28.1	Introduction	1226
28.2	Java Web Services Basics	1228
28.3	Creating, Publishing, Testing and Describing a Web Service	1228
28.3.1	Creating a Web Application Project and Adding a Web Service Class in Netbeans	1229
28.3.2	Defining the HugeInteger Web Service in Netbeans	1229
28.3.3	Publishing the HugeInteger Web Service from Netbeans	1234
28.3.4	Testing the HugeInteger Web Service with Sun Java System Application Server's Tester Web page	1234
28.3.5	Describing a Web Service with the Web Service Description Language (WSDL)	1238
28.4	Consuming a Web Service	1239
28.4.1	Creating a Client in Netbeans to Consume the HugeInteger Web Service	1239
28.4.2	Consuming the HugeInteger Web Service	1242

28.5	SOAP	1248
28.6	Session Tracking in Web Services	1249
28.6.1	Creating a Blackjack Web Service	1250
28.6.2	Consuming the Blackjack Web Service	1254
28.7	Consuming a Database-Driven Web Service from a Web Application	1265
28.7.1	Configuring Java DB in Netbeans and Creating the Reservation Database	1265
28.7.2	Creating a Web Application to Interact with the Reservation Web Service	1268
28.8	Passing an Object of a User-Defined Type to a Web Service	1273
28.9	REST-Based Web Services in ASP.NET	1283
28.9.1	REST-Based Web Service Functionality	1284
28.9.2	Creating an ASP.NET REST-Based Web Service	1288
28.9.3	Adding Data Components to a Web Service	1291
28.10	Wrap-Up	1294
28.11	Web Resources	1295

Part 5: Appendices **1303**

A	XHTML Special Characters	1304
B	XHTML Colors	1305
C	JavaScript Operator Precedence Chart	1308
D	ASCII Character Set	1310
E	Number Systems	1311
E.1	Introduction	1312
E.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1315
E.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1316
E.4	Converting from Binary, Octal or Hexadecimal to Decimal	1317
E.5	Converting from Decimal to Binary, Octal or Hexadecimal	1318
E.6	Negative Binary Numbers: Two's Complement Notation	1319
F	Unicode®	1325
F.1	Introduction	1326
F.2	Unicode Transformation Formats	1327
F.3	Characters and Glyphs	1328
F.4	Advantages/Disadvantages of Unicode	1328
F.5	Unicode Consortium's Website	1329
F.6	Using Unicode	1330
F.7	Character Ranges	1334

Index **1338**

Preface

*Science and technology and the various forms of art,
all unite humanity in a single and interconnected system.*

—Zhores Aleksandrovich Medvede

Welcome to Internet and web programming and *Internet & World Wide Web How to Program, Fourth Edition!* At Deitel & Associates, we write programming language textbooks and professional books for Prentice Hall, deliver corporate training worldwide and develop Web 2.0 Internet businesses. The book has been substantially reworked to reflect today's Web 2.0 Rich Internet Application-development methodologies. We have significantly tuned each of the chapters and added new chapters on some of the latest technologies.

New and Updated Features

Here's a list of updates we've made to the fourth edition of *Internet & World Wide Web How to Program*:

- Substantially reworked to reflect today's Web 2.0 Rich Internet Application-development methodologies.
- Coverage of the two leading web browsers—Internet Explorer 7 and Firefox 2. All client-side applications in the book run correctly on both browsers.
- New focus on Web 2.0 technologies and concepts.
- New chapter on Web 2.0 and Internet Business (reviewed by leaders in the Web 2.0 community).
- New focus on building Rich Internet Applications with the interactivity of desktop applications.
- New chapter on building Ajax-enabled web applications, with applications that demonstrate partial-page updates and type-ahead capabilities.
- New 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.
- New chapter on Microsoft Silverlight (a competitor to Adobe Flash and Flex)—a cross-browser and cross-platform plug-in for delivering .NET-based Rich Internet Applications that include audio, video and animations over the web.
- New chapter on rapid applications development of database-driven web applications with Ruby on Rails; also, discusses developing Ajax applications with the included Prototype and Script.aculo.us libraries.
- Updated chapter on Adobe Dreamweaver CS3 (Creative Suite 3), including new sections on CSS integration and the Ajax-enabled Spry framework.

- Updated chapters on Adobe Flash CS3, including a chapter on building a computer game.
- Significantly enhanced treatments of XHTML DOM manipulation and JavaScript events.
- Significantly enhanced treatment of XML DOM manipulation with JavaScript.
- New chapter on building SOAP-based web services with Java and REST-based web services with ASP.NET (using Visual Basic).
- Upgraded and enhanced the PHP chapter to PHP 5.
- Updated ASP.NET 1.1 coverage to ASP.NET 2.0, featuring ASP.NET Ajax.
- New JavaServer Faces (JSF) coverage emphasizing building Ajax-enabled JSF applications (replaces Servlets and JavaServer Pages).
- Client-side case studies that enable students to interact with preimplemented server-side applications and web services that we host at test.deitel.com.
- Several new and updated case studies including Deitel Cover Viewer (JavaScript/DOM), Address Book (Ajax), Cannon Game (Flash), Weather/Yahoo! Maps Mashup (Flex), Movie Player (Silverlight), 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).
- The Perl 5 and Python chapters from the previous edition of this book are posted in PDF form at www.deitel.com/books/iw3htp4/.

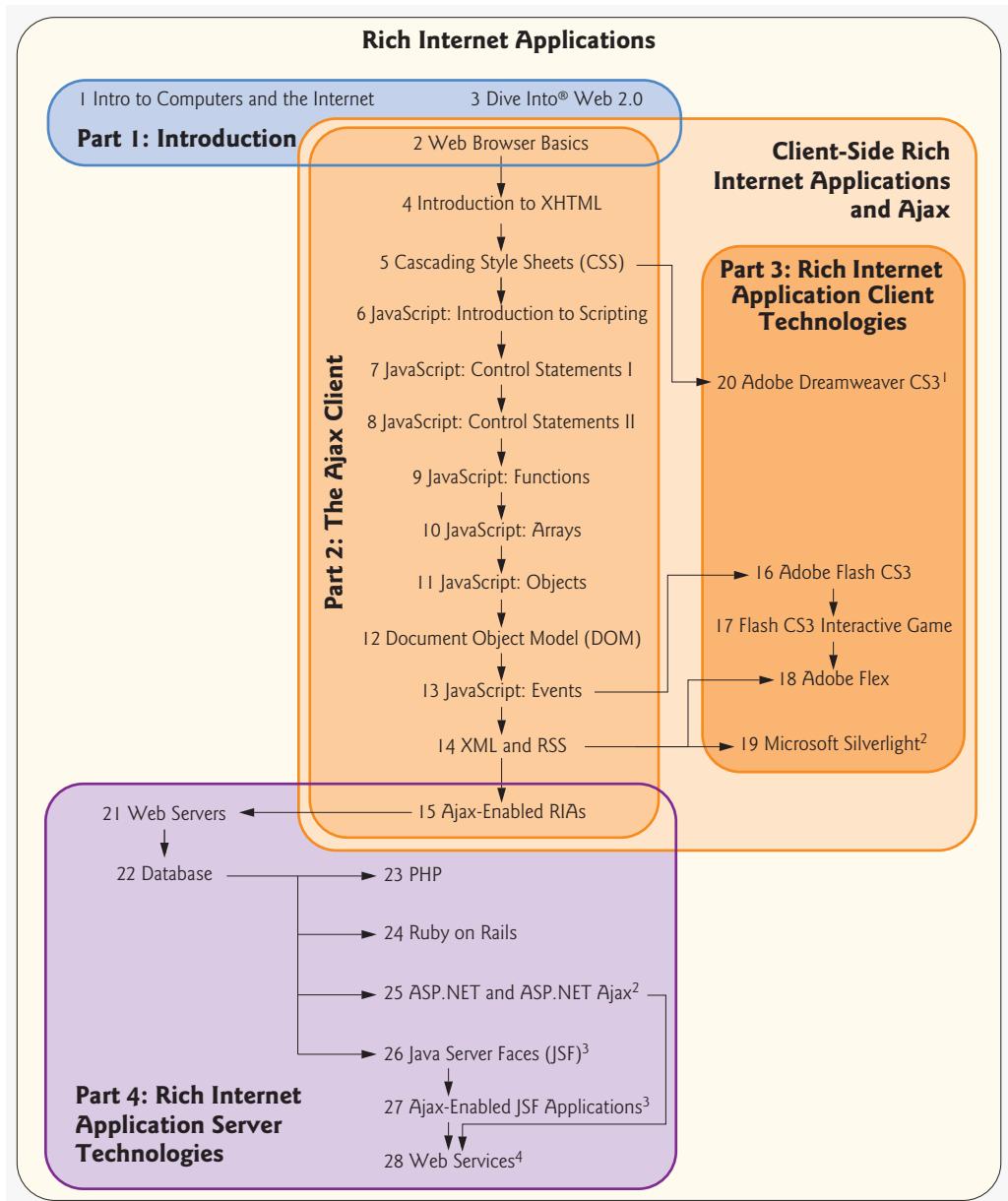
All of this has been carefully reviewed by distinguished academics and industry developers.

We believe that this book and its support materials will provide students and professionals with an informative, interesting, challenging and entertaining Internet and web programming educational experience. The book includes a suite of ancillary materials that help instructors maximize their students' learning experience.

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. Please let us know of other Resource Centers you'd like to see.

Dependency Chart

Figure 1 illustrates the dependencies that exist between chapters in the book. An arrow pointing into a chapter indicates that the chapter depends on the content of the chapter from which the arrow points. For example, Chapter 28, Web Services, depends on both Chapters 25 and 27. We recommend that you study all of a given chapter's dependencies before studying that chapter, though other orders are certainly possible. Some of the dependencies apply only to sections of chapters, so we advise readers to browse the material before designing a course of study. We've also commented on some additional dependen-



1. Section 20.9 requires a basic understanding of Ajax concepts (Chapter 15). Section 20.8 uses JavaScript events (Chapter 13).

2. Sections 19.8–19.10 and Chapter 25 require knowledge of Visual Basic and .NET.

3. Chapters 26–27 require knowledge of Java Standard Edition 6.

4. Sections 28.2–28.8 require Chapters 26–27. Section 28.9 requires Chapter 25.

Fig. 1 | *Internet & World Wide Web How to Program, 4/e* chapter dependency chart.

cies in the diagram's footnotes. This book is widely used in courses that teach pure client-side web programming, courses that teach pure server-side web programming, and courses

that mix and match some client-side and some server-side web programming. Curricula for courses in web programming are still evolving, and many syllabi exist to organize the material in this book. You can search the web for “syllabus,” “Internet,” “web” and “Deitel” to find syllabi currently used with this book. Readers interested in studying server-side technologies should understand how to build web pages using XHTML, CSS and object-based programming in JavaScript. Note the positioning of XML and Ajax in the dependency chart—these can be taught as part of a client-side unit, at the beginning of a server-side unit or split between the two.

Teaching Approach

Internet & World Wide Web How to Program, 4/e contains a rich collection of examples. The book concentrates on the principles of good software engineering and stresses program clarity. We teach by example. 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. *Internet & World Wide Web How to Program, 4/e* 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 Coloring. We syntax color 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-coloring conventions are as follows:

```
comments appear in green  
keywords appear in dark blue  
PHP, Ruby, ASP.NET and JSP delimiters, and errors appear in red  
constants and literal values appear in light blue  
all other code appears in black
```

Code Highlighting. We place gray rectangles around the key code segments in each program.

Using Fonts and Colors for Emphasis. We place the key terms and the index’s page reference for each defining occurrence in **bold blue** 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 *Internet & World Wide Web How to Program, 4/e* are available for download from:

www.deitel.com/books/iw3htp4/

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.

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 students—a mathematics major—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

Students 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 Tips

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 Tips

Students like to “turbo charge” their programs. These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.



Portability Tips

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 Observations

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

Wrap-Up Section. Each chapter ends with a brief “wrap-up” section that recaps the chapter content and transitions to the next chapter.

Summary Bullets. Each chapter ends with additional pedagogic features. We present a thorough, bulleted-list-style summary of the chapter, section by section.

Terminology. We include an alphabetized list of the important terms defined in each chapter. Each term also appears in the index, with its defining occurrence highlighted with a **bold, blue** page number.

Self-Review Exercises and Answers. Extensive self-review exercises and answers are included for self-study.

Exercises. Each chapter concludes with a substantial set of exercises, including simple recall of important terminology and concepts; writing individual statements; writing complete functions and scripts; and writing major term projects. The large number of exercises enables instructors to tailor their courses to the unique needs of their students and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes, major examinations and term projects. See our Programming Projects Resource Center (www.deitel.com/ProgrammingProjects/) for many additional exercise and project possibilities.

[**NOTE:** Please do not write to us requesting access to the Prentice Hall Instructor's Resource Center. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Pearson representatives.]

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.

Bibliography. An extensive bibliography of books, articles and online documentation is included to encourage further reading.

Student Resources Included with *Internet & World Wide Web How to Program, 4/e*

Many Internet and web development tools are available. We wrote *Internet & World Wide Web How to Program, 4/e* using Internet Explorer 7, Firefox 2 and other free-for-download software. 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/iw3htp4/

Instructor Resources for *Internet & World Wide Web How to Program, 4/e*

Internet & World Wide Web How to Program, 4/e has extensive instructor resources. The Prentice Hall Instructor's Resource Center contains the *Solutions Manual* with solutions to the end-of-chapter exercises, a *Test Item File* of multiple-choice questions (approximately

two per book section) and PowerPoint® slides containing all the code and figures in the text, plus bulleted items that summarize the key points in the text. Instructors can customize the slides. If you are not already a registered faculty member, contact your Pearson representative or visit vig.prenhall.com/replocator/.

Deitel® Buzz Online Free E-mail Newsletter

Each week, the *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 *Internet & World Wide Web How to Program, 4/e*. To subscribe, visit

www.deitel.com/newsletter/subscribe.html

The Deitel Online Resource Centers

Our website, www.deitel.com, provides Resource Centers on various topics including programming languages, software, Web 2.0, Internet business and open source projects (Fig. 2). 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. The Deitel Resource Centers are a starting point for your own research. We help you wade through the vast amount of content on the Internet by providing links to the most valuable resources. 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—Andrew Faden, a computer engineering major at Northeastern University; Scott Wehrwein, a computer science major at Middlebury College; Ilana Segall, a mathematical and computational science major at Stanford University; Mark Kagan, a computer science, economics and math major at Brandeis University; Jennifer Fredholm, an English and computer science major at New York University; Jessica Henkel, a psychology major and business minor at Northeastern University; and Kyle Banks, a computer science and business major at Northeastern University.

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. Carole Snyder and Dolores Mars did a remarkable job recruiting the book's large review team and managing the review process. Francesco Santalucia (an independent artist)

Deitel Resource Centers		
<i>Web 2.0 and Internet Business</i>	Linux	Java EE 5
Affiliate Programs	MySQL	Java SE 6
Alert Services	Open Source	JavaFX
Attention Economy	Perl	JavaScript
Blogging	PHP	JSON
Building Web Communities	Python	OpenGL
Community-Generated Content	Ruby	Perl
Google Adsense	Ruby on Rails	PHP
Google Analytics	<i>Software</i>	Programming Projects
Google Base	Apache	Python
Google Services	DotNetNuke (DNN)	Ruby
Google Video	Eclipse	Ruby on Rails
Google Web Toolkit	Firefox	Silverlight
Internet Advertising	Flash CS3 (Flash 9)	Visual Basic
Internet Business Initiative	Internet Explorer 7	Visual C++
Internet Public Relations	Linux	Web Services
Internet Video	MySQL	Web 3D Technologies
Joost	Open Source	XHTML
Link Building	Search Engines	XML
Location-Based Services	Web Servers	<i>Java</i>
Mashups	Wikis	Java
Microformats	Windows Vista	Java Certification and Assessment Testing
Podcasting	<i>Programming</i>	Java Design Patterns
Recommender Systems	.NET	Java EE 5
RSS	.NET 3.0	Java SE 6
Search Engine Optimization	Adobe Flex	JavaFX
Selling Digital Content	Ajax	<i>Microsoft</i>
Sitemaps	Apex On-Demand Programming Language	.NET
Skype	ASP.NET	.NET 3.0
Social Media	ASP.NET Ajax	ASP.NET
Social Networking	C	ASP.NET Ajax
Software as a Service (SaaS)	C#	C#
Virtual Worlds	C++	DotNetNuke (DNN)
Web 2.0	C++ Boost Libraries	Internet Explorer 7
Web 3.0	C++ Game Programming	Silverlight
Web Analytics	Code Search Engines and Code Sites	Visual Basic
Website Monetization	Computer Game Programming	Visual C++
Widgets	CSS 2.1	Windows Vista
<i>Open Source and LAMP Stack</i>	Dojo	<i>Other Topics</i>
Apache	Flash 9	Computer Games
DotNetNuke (DNN)	Java	Computing Jobs
Eclipse	Java Certification and Assessment Testing	Gadgets and Gizmos
Firefox	Java Design Patterns	Sudoku

Fig. 2 | Deitel Resource Centers www.deitel.com/resourcecenters.html

and Kristine Carney of Prentice Hall did a wonderful job designing the book's cover; we provided the concept, and they made it happen. Vince O'Brien, Scott Disanno, Bob Engelhardt and Marta Samsel did a marvelous job managing the book's production.

We wish to acknowledge the efforts of our reviewers. 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.

We sincerely appreciate the efforts of our third edition post-publication reviewers and our fourth edition reviewers:

Internet & World Wide Web How to Program, 4/e Reviewers

Roland Bouman (MySQL AB), Chris Bowen (Microsoft), Peter Brandano (KoolConnect Technologies, Inc.), Matt Chotin (Adobe), Chris Cornutt (PHPDeveloper.org), Phil Costa (Adobe), Umachitra Damodaran (Sun Microsystems), Vadiraj Deshpande (Sun Microsystems), Justin Erenkrantz (The Apache Software Foundation), Christopher Finke (Netscape), Jesse James Garrett (Adaptive Path), Mike Harsh (Microsoft), Kevin Henrikson (Zimbra.com), Tim Heuer (Microsoft), Molly E. Holtzschlag (W3C), Ralph Hooper (University of Alabama, Tuscaloosa), John Hrvatin (Microsoft), Johnvey Hwang (Splunk, Inc.), Joe Kromer (New Perspective and the Pittsburgh Adobe Flash Users Group), Eric Lawrence (Microsoft), Pete LePage (Microsoft), Billy B. L. Lim (Illinois State University), Shobana Mahadevan (Sun Microsystems), Patrick Mineault (Freelance Flash Programmer), Anand Narayanaswamy (Microsoft), Tim O'Reilly (O'Reilly Media, Inc.), 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), 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).

Internet & World Wide Web How to Program, 3/e Reviewers

America Azevedo (University of California at Berkeley), Tim Buntel (Macromedia, Inc.), Sylvia Candelaria de Ram (Cognizor, LLC; HumanMarkup.org), Wesley J. Chun (Cyber-Web Consulting), Marita Ellixson (Eglin AFB), Jay Glynn (American General AIG), James Greenwood (Poulternet), Timothy Greer (Middle Tennessee State University), James Huddleston (Independent Consultant), Lynn Kyle (Yahoo!, Inc.), Dan Livingston (Independent Consultant), Oge Marques (Florida Atlantic University), Mark Merkow (American Express Technologies), Dan Moore (Independent Consultant), George Semeczko (Royal & Sun Alliance Insurance Canada), Deborah Shapiro (Cittone Institutes), Matt Smith (Institute of Technology at Blanchardstown), Narayana Rao Surapaneni (Patni Computer Systems Limited), Stephanie Tauber (Tufts University), Yateen Thakkar (Syntel India, Ltd.), Cynthia Waddell (International Center for Disability Resources on the Internet), Lorain Walker (Lawrence Technological University) and Alnisa White (ILL Designs).

Well, there you have it! Welcome to the exciting world of Internet and web programming in a Web 2.0 world. We hope you enjoy this look at contemporary computer programming. Good luck! 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/iw3htp4/

We hope you enjoy reading *Internet & World Wide Web How to Program, Fourth Edition* as much as we enjoyed writing it!

Paul J. Deitel

Dr. Harvey M. Deitel

Maynard, Massachusetts

August 2007

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 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 45 years of academic and industry 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, Advanced Java, C, C#, Visual C++, Visual Basic, XML, object technology and Internet and web programming. The founders of Deitel & Associates, Inc., are Dr. Harvey M. Deitel and Paul J. Deitel. The company's clients include many of the world's largest companies, government agencies, branches of the military, and academic institutions. Through its 31-year publishing partnership with Prentice Hall, Deitel & Associates, Inc. publishes leading-edge programming textbooks, professional books, interactive multimedia *Cyber Classrooms*, *Complete Training Courses*, Web-based training courses, online and offline 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.deitel.com

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

www.deitel.com/newsletter/subscribe.html

Check out the growing list of online Deitel Resource Centers at:

www.deitel.com/resourcecenters.html

Individuals wishing to purchase Deitel publications can do so through:

www.deitel.com/books/index.html

Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Prentice Hall. For more information, visit

www.prenhall.com/misctm/support.html#order

Before You Begin

Please follow these instructions to download the book's examples and ensure you have a current web browser before you begin using this book.

Downloading the *Internet & World Wide Web How to Program, 4/e* Source Code

The source code in *Internet & World Wide Web How To Program, 4/e* can be downloaded as a ZIP archive file from www.deitel.com/books/iw3htp4/. After you register and log in, click the link for the examples under **Download Code Examples and Other Premium Content for Registered Users**. Extract the example files to your hard disk using a ZIP file extractor program, such as WinZip (www.winzip.com). On Windows, we suggest that you extract the files to a folder such as C:\iw3htp4_examples. On Mac OS X and Linux, we suggest that you extract the files to a folder named iw3htp4_examples in your home folder. [Note: If you are working in a computer lab, ask your instructor where you can save the example code.]

Web Browsers Used in This Book

We've tested every example in this book using Mozilla's Firefox 2 and Microsoft's Internet Explorer 7 web browsers. Before you begin, ensure that you have one or both of these browsers installed on your computer. Internet Explorer 7 is available only for Microsoft Windows operating systems. If you are a Windows user and do not have Internet Explorer 7, you can get download it from www.update.microsoft.com using Microsoft's Windows Update service. Firefox 2 is available for most platforms. You can download Firefox 2 from www.firefox.com.

Many of the book's examples *will not work* in Internet Explorer 6. Though the examples in this book may run on other web browsers, such as Opera (www.opera.com) or Apple's Safari (www.apple.com/safari/), we have not tested the examples on these or any other browsers.

You are now ready to begin your web programming studies with *Internet & World Wide Web How to Program, 4/e*. We hope you enjoy the book! If you have any questions, please feel free to email us at deitel@deitel.com. We'll respond promptly.

1

PART

Introduction

The renaissance of interest in the web that we call Web 2.0 has reached the mainstream.

—Tim O'Reilly

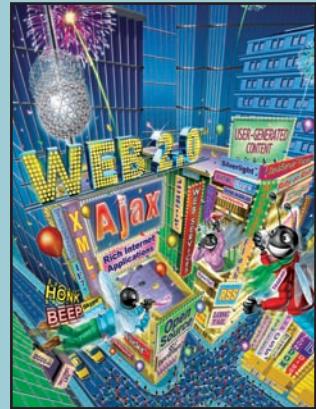


Introduction to Computers and the Internet

OBJECTIVES

In this chapter you will learn:

- Basic computing concepts.
- The different types of programming languages.
- The evolution of the Internet and the World Wide Web.
- What Web 2.0 is and why it's having such an impact among Internet-based and traditional businesses.
- What Rich Internet Applications (RIAs) are and the key software technologies used to build RIAs.



The renaissance of interest in the web that we call Web 2.0 has reached the mainstream.

—Tim O'Reilly

Billions of queries stream across the servers of these Internet services—the aggregate thoughtstream of humankind, online.

—John Battelle, *The Search*

People are using the web to build things they have not built or written or drawn or communicated anywhere else.

—Tim Berners-Lee

Some people take what we contribute and extend it and contribute it back [to Ruby on Rails]. That's really the basic open source success story.

—David Heinemeier Hansson, interviewed by Chris Karr at www.Chicagoist.com

Outline

- 1.1 Introduction
- 1.2 What Is a Computer?
- 1.3 Computer Organization
- 1.4 Machine Languages, Assembly Languages and High-Level Languages
- 1.5 History of the Internet and World Wide Web
- 1.6 World Wide Web Consortium (W3C)
- 1.7 Web 2.0
- 1.8 Personal, Distributed and Client/Server Computing
- 1.9 Hardware Trends
- 1.10 Key Software Trend: Object Technology
- 1.11 JavaScript: Object-Based Scripting for the Web
- 1.12 Browser Portability
- 1.13 C, C++ and Java
- 1.14 BASIC, Visual Basic, Visual C++, C# and .NET
- 1.15 Software Technologies
- 1.16 Notes about *Internet & World Wide Web How to Program, 4/e*
- 1.17 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

1.1 Introduction

Welcome to Internet and World Wide 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. As you read this book, you may want to refer to www.deitel.com for updates and additional information.

The technologies you'll learn in this book are fun for novices, and simultaneously are appropriate for experienced professionals who build substantial information systems. *Internet & World Wide Web How to Program, Fourth Edition*, is designed to be an effective learning tool for each of these audiences. How can one book appeal to both groups? The answer is that the core of this book emphasizes achieving program clarity through the proven techniques of structured programming, object-based programming and object-oriented programming. Beginners will learn programming the right way from the beginning. Experienced programmers will 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 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." These examples are available for download from our website, www.deitel.com/books/iw3htp4/.

The early chapters introduce computer fundamentals, the Internet and the web. We show how to use software for browsing the web. We 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 (Cas-

cading Style Sheets) and the DOM (Document Object Model). We often refer to “programming” as scripting—for reasons that will soon become clear. Novices will find that the material in the JavaScript chapters presents a solid foundation for the deeper treatment of scripting in the Adobe Flash, Adobe Flex, Microsoft Silverlight, 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.

Most people are familiar with the exciting things that computers can do. Using this textbook, you’ll learn how to command computers to perform specific tasks. **Software** (i.e., the instructions you write to command the computer to perform **actions** and make **decisions**) controls computers (often referred to as **hardware**). JavaScript and PHP are among today’s most popular software development languages for web-based applications.

Computer use is increasing in almost every field of endeavor. In an era of steadily rising costs, computing costs have been decreasing dramatically because of rapid developments in both hardware and software technologies. Computers that filled large rooms and cost millions of dollars just two decades ago can now be inscribed on the surfaces of silicon chips smaller than fingernails, costing perhaps a few dollars each. Silicon is one of the most abundant materials on earth—it is an ingredient in common sand. Silicon-chip technology has made computing so economical that more than a billion general-purpose computers worldwide are now helping people in business, industry, government, education and in their personal lives. And billions more computers are embedded in cell phones, appliances, automobiles, security systems, game systems and so much more.

Through the early 1990s most students in introductory programming courses learned only the methodology called structured programming. As you study the various scripting languages in this book, you’ll learn both structured programming and the newer methodology called object-based programming. After this, you’ll be well prepared to study today’s popular full-scale programming languages such as C++, Java, C# and Visual Basic .NET and to learn the even more powerful programming methodology of object-oriented programming. We believe that object-oriented programming will be the key programming methodology for at least several decades.

Today’s users are accustomed to 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** (i.e., different types of computers running different operating systems).

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 concentrate on using technologies such as the Extensible HyperText Markup Language (XHTML), JavaScript, CSS, Flash, Flex, Silverlight, Dreamweaver and Extensible Markup Language (XML) to build the portions of web-based applications that reside on the **client side** (i.e., the portions of applications that typically run in your web browsers such as Mozilla’s Firefox 2 or

Microsoft's Internet Explorer 7). Later in the book we concentrate on using technologies such as web servers, databases (integrated collections of data), 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. We begin with a discussion of computer hardware and software fundamentals. If you are generally familiar with computers, the Internet and the web, you may want to skip some or all of this chapter.

To keep up to date with Internet and web programming developments, and the latest information on *Internet & World Wide Web How to Program, 4/e*, at Deitel & Associates, please register for our free e-mail newsletter, the *Deitel® Buzz Online*, at

www.deitel.com/newsletter/subscribe.html

Please check out our growing list of Internet and web programming, and Internet business Resource Centers at

www.deitel.com/resourcecenters.html

Each week, we announce our latest Resource Centers in the newsletter. Figure 2 in the Preface includes a complete list of Deitel Resource Centers at the time of this writing. The Resource Centers include links to, and descriptions of, key tutorials, demos, free software tools, articles, e-books, white papers, videos, podcasts, blogs, RSS feeds and more that will help you deepen your knowledge of most of the subjects we discuss in this book.

Errata and updates for the book are posted at

www.deitel.com/books/iw3htp4/

You're embarking on a challenging and rewarding path. As you proceed, if you have any questions, please send e-mail to

deitel@deitel.com

We'll respond promptly. We hope that you'll enjoy learning with *Internet & World Wide Web How to Program, Fourth Edition*.

Architecture of Internet & World Wide Web How to Program, 4/e

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. In the interim since the previous edition of this book, Deitel has evolved into a development organization, while maintaining 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 edition has been enhanced with discussions of many practical issues we've encountered in developing that infrastructure.

Figure 1.1 shows the architecture of *Internet & World Wide Web How to Program, 4/e*. The book is divided into several parts. The first part, Chapters 1–3, provides an introduction to the Internet and the web, web browsers and Web 2.0. These chapters provide a foundation for understanding Web 2.0 and Rich Internet Application development. Chapter 1 introduces hardware, software, communications and Web 2.0 topics. If you are

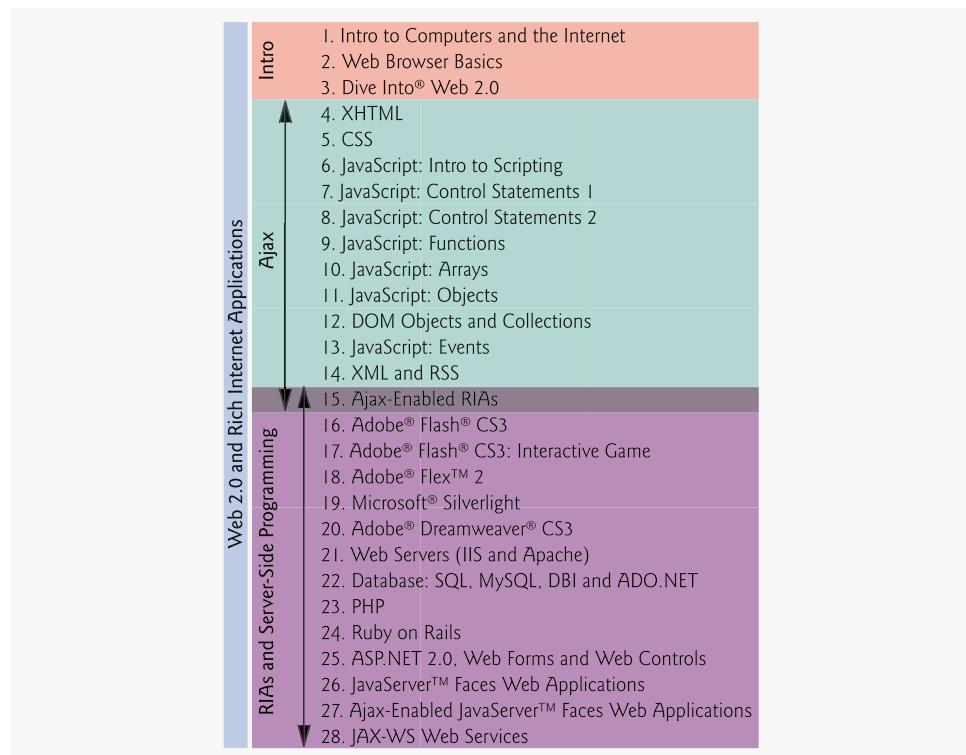


Fig. 1.1 | Architecture of *Internet & World Wide Web How to Program*, 4/e.

a serious web developer, you'll want to test your web applications across many browsers and platforms. The examples for this book target Microsoft's Internet Explorer 7 (IE7) and Mozilla's Firefox 2 (FF2) browsers, each of which is introduced in Chapter 2. The examples execute correctly in both browsers. Many of the examples will also work in other browsers such as Opera and Safari. Many of the examples will not work on earlier browsers. Microsoft Windows users should upgrade to IE7 and install FF2; readers with other operating systems should install Firefox 2. Web browsers—a crucial component of web applications—are free, as are most of the software technologies we present in this book. Chapter 3 discusses Web 2.0 from technical, business and social perspectives.

The second part of the book, Chapters 4–15, presents a 12-chapter treatment of Ajax component technologies that concludes with Chapter 15's treatment of Ajax development. Ajax is not a new technology—we've been writing about all but one of its component technologies since the first edition of this book in 1999, and many of the technologies existed before that. However, Ajax is one of the key technologies of Web 2.0 and RIAs. Several later chapters in the book demonstrate technologies that encapsulate Ajax functionality to help make it operate across a wide variety of browsers and browser versions.

The third part of the book, Chapters 15–28, focuses on both the client and server sides of the GUI and graphical part of RIA development. Here we cover client-side technologies such as Adobe Flash, Adobe Flex and Microsoft Silverlight 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, several 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 web services.

You may have noticed that Chapter 15, Ajax-Enabled Rich Internet Applications, overlaps the second and third parts of the book. Chapter 15 serves as a bridge from “raw” Ajax development to sophisticated RIA development.

1.2 What Is a Computer?

A **computer** is a device that can perform computations and make logical decisions billions of times faster than human beings can. For example, many of today’s personal computers can perform several billion additions per second. A person could operate a desk calculator for an entire lifetime and still not complete as many calculations as a powerful personal computer can perform in one second! (Points to ponder: How would you know whether the person added the numbers correctly? How would you know whether the computer added the numbers correctly?) Today’s fastest **supercomputers** can perform trillions of additions per second!

Computers process data under the control of sets of instructions called **computer programs**. These programs guide the computer through orderly sets of actions specified by people called **computer programmers**.

A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disk, memory, DVD drives and processing units). The programs that run on a computer are referred to as software. Hardware costs have been declining dramatically in recent years, to the point that personal computers have become a commodity. In this book, you’ll learn proven methods that are reducing software development costs—object-oriented programming and object-oriented design.

1.3 Computer Organization

Regardless of differences in physical appearance, virtually every computer may be envisioned as divided into six **logical units** or sections:

1. **Input unit.** This is the “receiving” section of the computer. It obtains information (data and computer programs) from **input devices** and places this information at the disposal of the other units for processing. Most information is entered into computers through keyboards and mouse devices. Information also can be entered in many other ways, including by speaking to your computer, scanning images, uploading digital photos and videos, and receiving information from a network, such as the Internet.
2. **Output unit.** This is the “shipping” section of the computer. It takes information that the computer has processed and places it on various **output devices** to make the information available for use outside the computer. Most information output from computers today is displayed on screens, printed on paper or used to control other devices. Computers also can output their information to networks, such as the Internet.

3. **Memory unit.** This is the rapid-access, relatively low-capacity “warehouse” section of the computer. It stores computer programs while they are being executed. It retains information that has been entered through the input unit, so that it will be immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is typically lost when the computer’s power is turned off. The memory unit is often called either **memory** or **primary memory**.
4. **Arithmetic and logic unit (ALU).** This is the “manufacturing” section of the computer. It is responsible for performing calculations, such as addition, subtraction, multiplication and division. It contains the decision mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they are equal, or if one is larger than the other.
5. **Central processing unit (CPU).** This is the computer’s “administrative” section. It coordinates and supervises the other sections’ operations. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today’s computers have multiple CPUs and, hence, can perform many operations simultaneously—such computers are called **multiprocessors**.
6. **Secondary storage unit.** This is the computer’s long-term, high-capacity “warehousing” section. Programs or data not actively being used by the other units normally are placed on secondary storage devices, such as your hard drive, until they are needed, possibly hours, days, months or even years later. Information in secondary storage takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Other secondary storage devices include CDs and DVDs, which can hold hundreds of millions and billions of characters, respectively.

1.4 Machine Languages, Assembly Languages and High-Level Languages

Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate **translation** steps. Hundreds of computer languages are in use today. These may be divided into three general types:

1. Machine languages
2. Assembly languages
3. High-level languages

Any computer can directly understand only its own **machine language**. Machine language is the “natural language” of a computer and as such is defined by its hardware design. [Note: Machine language is often referred to as **object code**. This term predates “object-oriented programming.” These two uses of “object” are unrelated.] Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time. Machine languages are **machine dependent** (i.e., a particular machine language can be used on only one type of computer). Such languages are cumbersome for humans, as illustrated by the following

section of an early machine-language program that adds overtime pay to base pay and stores the result in gross pay:

```
+1300042774  
+1400593419  
+1200274027
```

Machine-language programming was simply too slow, tedious and error prone for most programmers. Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations. These abbreviations formed the basis of **assembly languages**. **Translator programs** called **assemblers** were developed to convert early assembly-language programs to machine language at computer speeds. The following section of an assembly-language program also adds overtime pay to base pay and stores the result in gross pay:

```
load    basepay  
add     overpay  
store   grosspay
```

Although such code is clearer to humans, it is incomprehensible to computers until translated to machine language.

Computer usage increased rapidly with the advent of assembly languages, but programmers still had to use many instructions to accomplish even the simplest tasks. To speed the programming process, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks. Translator programs called **compilers** convert high-level language programs into machine language. High-level languages allow programmers to write instructions that look almost like everyday English and contain commonly used mathematical notations. A payroll program written in a high-level language might contain a statement such as

```
grossPay = basePay + overTimePay;
```

From your standpoint, obviously, high-level languages are preferable to machine and assembly language. C, C++, Microsoft's .NET languages (e.g., Visual Basic, Visual C++ and Visual C#) and Java are among the most widely used high-level programming languages. All of the Internet and web application development languages that you'll learn in this book are high-level languages.

The process of compiling a high-level language program into machine language can take a considerable amount of computer time. **Interpreter** programs were developed to execute high-level language programs directly, although much more slowly. In this book, we study several key programming languages, including JavaScript, ActionScript, PHP and Ruby on Rails—each of these **scripting languages** is processed by interpreters. We also study markup languages such as XHTML and XML, which can be processed by interpreted scripting languages. You'll see that interpreters have played an especially important role in helping scripting languages and markup languages achieve their goal of portability across a variety of platforms.



Performance Tip 1.1

Interpreters have an advantage over compilers in scripting. An interpreted program can begin executing as soon as it is downloaded to the client's machine, without the need to be compiled before it can execute. On the downside, scripts generally run much slower than compiled code.



Portability Tip 1.1

Interpreted languages are more portable than compiled languages. Interpreters can be implemented for each platform on which the interpreted languages need to execute.



Software Engineering Observation 1.1

Interpreted languages are more dynamic than compiled languages. For example, server-side applications can generate code in response to user interactions, and that code can then be interpreted in a browser.

1.5 History of the Internet and World Wide Web

In the late 1960s, one of the authors (HMD) was a graduate student at MIT. His research at MIT’s Project MAC (now the Laboratory for Computer Science—the home of the World Wide Web Consortium) was funded by ARPA—the Advanced Research Projects Agency of the Department of Defense. ARPA sponsored a conference at which several dozen ARPA-funded graduate students were brought together at the University of Illinois at Urbana-Champaign to meet and share ideas. During this conference, ARPA rolled out the blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions. They were to be connected with communications lines operating at a then-stunning 56 Kbps (i.e., 56,000 bits per second)—this at a time when most people (of the few who could) were connecting over telephone lines to computers at a rate of 110 bits per second. There was great excitement at the conference. Researchers at Harvard talked about communicating with the Univac 1108 “supercomputer” at the University of Utah to handle calculations related to their computer graphics research. Many other intriguing possibilities were raised. Academic research about to take a giant leap forward. Shortly after this conference, ARPA proceeded to implement the **ARPANET**, which eventually evolved into today’s **Internet**.

Things worked out differently from what was originally planned. Rather than enabling researchers to share each other’s computers, it rapidly became clear that enabling researchers to communicate quickly and easily via what became known as **electronic mail** (**e-mail**, for short) was the key early benefit of the ARPANET. This is true even today on the Internet, as e-mail facilitates communications of all kinds among a billion people worldwide.

One of the primary goals for ARPANET was to allow multiple users to send and receive information simultaneously over the same communications paths (e.g., phone lines). The network operated with a technique called **packet switching**, in which digital data was sent in small bundles called **packets**. The packets contained address, error-control and sequencing information. The address information allowed packets to be routed to their destinations. The sequencing information helped in reassembling the packets—which, because of complex routing mechanisms, could actually arrive out of order—into their original order for presentation to the recipient. Packets from different senders were intermixed on the same lines. This packet-switching technique greatly reduced transmission costs, as compared with the cost of dedicated communications lines.

The network was designed to operate without centralized control. If a portion of the network failed, the remaining working portions would still route packets from senders to receivers over alternative paths for reliability.

The protocol for communicating over the ARPANET became known as **TCP**—the **Transmission Control Protocol**. TCP ensured that messages were properly routed from sender to receiver and that they arrived intact.

As the Internet evolved, organizations worldwide were implementing their own networks for both intraorganization (i.e., within the organization) and interorganization (i.e., between organizations) communications. A wide variety of networking hardware and software appeared. One challenge was to get these different networks to communicate. ARPA accomplished this with the development of **IP**—the **Internet Protocol**, truly creating a “**network of networks**,” the current architecture of the Internet. The combined set of protocols is now commonly called **TCP/IP**.

Initially, Internet use was limited to universities and research institutions; then the military began using the Internet. Eventually, the government decided to allow access to the Internet for commercial purposes. Initially, there was resentment in the research and military communities—these groups were concerned that response times would become poor as “the Net” became saturated with users.

In fact, the exact opposite has occurred. Businesses rapidly realized that they could tune their operations and offer new and better services to their clients, so they started spending vast amounts of money to develop and enhance the Internet. This generated fierce competition among communications carriers and hardware and software suppliers to meet this demand. The result is that **bandwidth** (i.e., the information-carrying capacity) on the Internet has increased tremendously and costs have decreased significantly.

The **World Wide Web** allows computer users to locate and view multimedia-based documents on almost any subject over the Internet. Though the Internet was developed decades ago, the web is a relatively recent creation. In 1989, **Tim Berners-Lee** of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via hyperlinked text documents. Berners-Lee called his invention the **HyperText Markup Language (HTML)**. He also wrote communication protocols to form the backbone of his new information system, which he called the World Wide Web. In particular, he wrote the **Hypertext Transfer Protocol (HTTP)**—a communications protocol used to send information over the web. Web use exploded with the availability in 1993 of the Mosaic browser, which featured a user-friendly graphical interface. Marc Andreessen, whose team at NCSA developed Mosaic, went on to found Netscape, the company that many people credit with initiating the explosive Internet economy of the late 1990s.

In the past, most computer applications ran on computers that were not connected to one another, whereas today’s applications can be written to communicate among the world’s computers. The Internet mixes computing and communications technologies. It makes our work easier. It makes information instantly and conveniently accessible worldwide. It enables individuals and small businesses to get worldwide exposure. It is changing the way business is done. People can search for the best prices on virtually any product or service. Special-interest communities can stay in touch with one another. Researchers can be made instantly aware of the latest breakthroughs. The Internet and the web are surely among humankind’s most profound creations.

1.6 World Wide Web Consortium (W3C)

In October 1994, Tim Berners-Lee founded an organization—called the **World Wide Web Consortium (W3C)**—devoted to developing nonproprietary, interoperable technol-

ogies for the World Wide Web. One of the W3C's primary goals is to make the web universally accessible—regardless of ability, language or culture. The W3C home page (www.w3.org) provides extensive resources on Internet and web technologies.

The W3C is also a standardization organization. Web technologies standardized by the W3C are called **Recommendations**. W3C Recommendations include the Extensible HyperText Markup Language (XHTML), Cascading Style Sheets (CSS), HyperText Markup Language (HTML—now considered a “legacy” technology) and the Extensible Markup Language (XML). A recommendation is not an actual software product, but a document that specifies a technology’s role, syntax rules and so forth.

1.7 Web 2.0

In 2003 there was a noticeable shift in how people and businesses were using the web and developing web-based applications. The term **Web 2.0** was coined by **Dale Dougherty** of **O’Reilly® Media**¹ in 2003 to describe this trend. Although it became a major media buzzword, few people really know what Web 2.0 means. Generally, Web 2.0 companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis, etc.).

Web 1.0 (the state of the web through the 1990s and early 2000s) was focused on a relatively small number of companies and advertisers producing content for users to access (some people called it the “brochure web”). Web 2.0 *involves* the user—not only is the content often created by the users, but users help organize it, share it, remix it, critique it, update it, etc. One way to look at Web 1.0 is as a *lecture*, a small number of professors informing a large audience of students. In comparison, Web 2.0 is a *conversation*, with everyone having the opportunity to speak and share views.

Web 2.0 is providing new opportunities and connecting people and content in unique ways. Web 2.0 embraces an **architecture of participation**—a design that encourages user interaction and community contributions. You, the user, are the most important aspect of Web 2.0—so important, in fact, that in 2006, *TIME Magazine*’s “Person of the Year” was “you.”² The article recognized the social phenomenon of Web 2.0—the shift away from a powerful few to an empowered many. Several popular blogs now compete with traditional media powerhouses, and many Web 2.0 companies are built almost entirely on user-generated content. For websites like MySpace®, Facebook®, Flickr™, YouTube, eBay® and Wikipedia®, users create the content, while the companies provide the platforms. These companies *trust their users*—without such trust, users cannot make significant contributions to the sites.

The architecture of participation has influenced software development as well. Open source software is available for anyone to use and modify with few or no restrictions. Using **collective intelligence**—the concept that a large diverse group of people will create smart ideas—communities collaborate to develop software that many people believe is better and more robust than proprietary software. Rich Internet Applications (RIAs) are being devel-

1. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1>>.
2. Grossman, L. "TIME's Person of the Year: You." *TIME*, December 2006 <<http://www.time.com/time/magazine/article/0,9171,1569514,00.html>>.

oped using technologies (such as Ajax) that have the look and feel of desktop software, enhancing a user's overall experience. Software as a Service (SaaS)—software that runs on a server instead of a local computer—has also gained prominence because of sophisticated new technologies and increased broadband Internet access.

Search engines, including Google™, Yahoo!®, MSN®, Ask™, and many more, have become essential to sorting through the massive amount of content on the web. Social bookmarking sites such as del.icio.us and Ma.gnolia allow users to share their favorite sites with others. Social media sites such as Digg™, Spotplex™ and Netscape® enable the community to decide which news articles are the most significant. The way we find the information on these sites is also changing—people are **tagging** (i.e., labeling) web content by subject or keyword in a way that helps anyone locate information more effectively.

Web services have emerged and, in the process, have inspired the creation of many Web 2.0 businesses. Web services allow you to incorporate functionality from existing applications and websites into your own web applications quickly and easily. For example, using Amazon Web Services™, you can create a specialty bookstore and earn revenues through the Amazon Associates Program; or, using Google™ Maps web services with eBay web services, you can build location-based “mashup” applications to find auction items in certain geographical areas. Web services, inexpensive computers, abundant high-speed Internet access, open source software and many other elements have inspired new, exciting, **lightweight business models** that people can launch with only a small investment. Some types of websites with rich and robust functionality that might have required hundreds of thousands or even millions of dollars to build in the 1990s can now be built for nominal amounts of money.

In the future, we'll see computers learn to understand the meaning of the data on the web—the beginnings of the Semantic Web are already appearing. Continual improvements in hardware, software and communications technologies will enable exciting new types of applications.

These topics and more are covered in a detailed walkthrough in Chapter 3, Dive Into® Web 2.0. The chapter highlights the major characteristics and technologies of Web 2.0, providing examples of popular Web 2.0 companies and Web 2.0 Internet business and monetization models. You'll learn about user-generated content, blogging, content networks, social networking, location-based services and more. In Chapters 4–28, you'll learn key software technologies for building web-based applications in general, and Ajax-enabled, web-based Rich Internet Applications in particular. See our Web 2.0 Resource Center at www.deitel.com/web2.0/ for more information.

1.8 Personal, Distributed and Client/Server Computing

In 1977, Apple Computer popularized **personal computing**. Computers became so economical that people could buy them for their own personal or business use. In 1981, IBM, the world's largest computer vendor, introduced the IBM Personal Computer. This quickly legitimized personal computing in business, industry and government organizations, where IBM mainframes were heavily used.

These computers were for the most part “stand-alone” units—people transported disks back and forth between them to share information (this was often called “sneakernet”). Although early personal computers were not powerful enough to timeshare several users, these machines could be linked together in computer networks, sometimes over tele-

phone lines and sometimes in **local area networks (LANs)** within an organization. This led to the phenomenon of **distributed computing**, in which an organization's computing, instead of being performed only at some central computer installation, is distributed over networks to the sites where the organization's work is performed. Personal computers were powerful enough to handle the computing requirements of individual users as well as the basic communications tasks of passing information between computers electronically.

Today's personal computers are as powerful as the million-dollar machines of just a few decades ago. The most powerful desktop machines—called **workstations**—provide individual users with enormous capabilities. Information is shared easily across computer networks, where computers called **servers** (file servers, database servers, web servers, etc.) offer data storage and other capabilities that may be used by **client** computers distributed throughout the network, hence the term **client/server computing**. Today's popular operating systems, such as UNIX, Linux, Mac OS X and Microsoft's Windows-based systems, provide the kinds of capabilities discussed in this section.

1.9 Hardware Trends

The Internet community thrives on the continuing stream of dramatic improvements in hardware, software and communications technologies. In general, people expect to pay at least a little more for most products and services every year. The opposite generally has been the case in the computer and communications industries, especially with regard to the hardware costs of supporting these technologies. For many decades, and with no change expected in the foreseeable future, hardware costs have fallen rapidly. This is a phenomenon of technology. **Moore's Law** states that the power of hardware doubles every two years, while the price remains essentially the same.³ Significant improvements also have occurred in the communications field, especially in recent years, with the enormous demand for communications bandwidth attracting tremendous competition, forcing communications bandwidth to increase and prices to decline. We know of no other fields in which technology moves so quickly and costs fall so rapidly.

When computer use exploded in the 1960s and 1970s, there was talk of the huge improvements in human productivity that computing and communications would bring about. However, these productivity improvements did not immediately materialize. Organizations were spending vast sums on computers and distributing them to their workforces, but without immediate productivity gains. On the hardware side, it was the invention of microprocessor chip technology and its wide deployment in the late 1970s and 1980s which laid the groundwork for significant productivity improvements in the 1990s. On the software side, productivity improvements are now coming from object technology, which we use throughout this book.

Recently, hardware has been moving toward mobile, wireless technology. Small handheld devices are now more powerful than early 1970s supercomputers. Portability is now a major focus for the computer industry. Wireless data-transfer speeds have become so fast that many Internet users' primary web access is through wireless networks. The next few years will see dramatic advances in wireless capabilities for personal users and businesses.

3. Moore, G. "Cramming More Components onto Integrated Circuits." *Electronics*, April 1965 <[ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf](http://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf)>.

1.10 Key Software Trend: Object Technology

One of the authors, HMD, remembers the great frustration felt in the 1960s by software development organizations, especially those working on large-scale projects. During his undergraduate years, he had the privilege of working summers at a leading computer vendor on the teams developing timesharing, virtual-memory operating systems. This was a great experience for a college student. But, in the summer of 1967, reality set in when the company “decommitted” from producing as a commercial product the particular system on which hundreds of people had been working for many years. It was difficult to get this thing called software right—software is “complex stuff.”

Improvements to software technology did emerge, with the benefits of structured programming (and the related disciplines of [structured systems analysis and design](#)) being realized in the 1970s. Not until the technology of object-oriented programming became widely used in the 1990s, though, did software developers feel they had the necessary tools for making major strides in the software development process.

What are objects and why are they special? Actually, object technology is a packaging scheme that helps us create meaningful software units. These can be large and are highly focused on particular applications areas. There are date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects and so on. In fact, almost any noun can be reasonably represented as an object.

We live in a world of objects. Just look around you. There are cars, planes, people, animals, buildings, traffic lights, elevators and the like. Before object-oriented languages appeared, procedural programming languages (such as Fortran, COBOL, Pascal, BASIC and C) were focused on actions (verbs) rather than on things or objects (nouns). Programmers living in a world of objects programmed primarily using verbs. This made it awkward to write programs. Now, with the availability of popular object-oriented languages, such as C++, Java, Visual Basic and C#, programmers continue to live in an object-oriented world *and can program in an object-oriented manner*. This is a more natural process than procedural programming and has resulted in significant productivity gains.

A key problem with procedural programming is that the program units do not effectively mirror real-world entities, so these units are not particularly reusable. It’s not unusual for programmers to “start fresh” on each new project and have to write similar software “from scratch.” This wastes time and money, as people repeatedly “reinvent the wheel.” With object technology, the software entities created (called [classes](#)), if properly designed, tend to be reusable on future projects. Using libraries of reusable componentry can greatly reduce effort required to implement certain kinds of systems (compared to the effort that would be required to reinvent these capabilities on new projects).



Software Engineering Observation 1.2

Extensive class libraries of reusable software components are available on the Internet. Many of these libraries are free.



Software Engineering Observation 1.3

Some organizations report that the key benefit object-oriented programming gives them is not software that is reusable but, rather, software that is more understandable, better organized and easier to maintain, modify and debug. This can be significant, because perhaps as much as 80 percent of software cost is associated not with the original efforts to develop the software, but with the continued evolution and maintenance of that software throughout its lifetime.

1.11 JavaScript: Object-Based Scripting for the Web

JavaScript is an object-based scripting language with strong support for proper software engineering techniques. Students learn to create and manipulate objects from the start in JavaScript. JavaScript is available free in today's popular web browsers.

Does JavaScript provide the solid foundation of programming principles typically taught in first programming courses—a portion of the intended audience for this book? We think so.

The JavaScript chapters of the book are more than just an introduction to the language. They also present an introduction to computer programming fundamentals, including control structures, functions, arrays, recursion, strings and objects. Experienced programmers will read Chapters 6–13 quickly and master JavaScript by reading our live-code examples and by examining the corresponding screenshots. Beginners will learn computer programming in these carefully paced chapters by reading the code explanations and completing the many exercises.

JavaScript is a powerful scripting language. Experienced programmers sometimes take pride in creating strange, contorted, convoluted JavaScript code. This kind of coding makes programs more difficult to read, test and debug. This book is also geared for novice programmers; for all readers we stress program clarity.



Good Programming Practice 1.1

Write your programs in a simple and straightforward manner. This is sometimes referred to as KIS (“keep it simple”). One key aspect of keeping it simple is another interpretation of KIS—“keep it small.” Do not “stretch” the language by trying bizarre uses.

You'll see that JavaScript is a portable scripting language and that programs written in JavaScript can run in many web browsers. Actually, portability is an elusive goal.



Portability Tip 1.2

Although it is easier to write portable programs in JavaScript than in many other programming languages, differences among interpreters and browsers make portability difficult to achieve. Simply writing programs in JavaScript does not guarantee portability. Programmers occasionally need to research platform variations and write their code accordingly.



Portability Tip 1.3

When writing JavaScript programs, you need to deal directly with cross-browser portability issues. Such issues are hidden by JavaScript libraries (e.g., Dojo, Prototype, Script.aculo.us and ASP.NET Ajax) which provide powerful, ready-to-use capabilities that simplify JavaScript coding by making it cross-browser compatible.



Error-Prevention Tip 1.1

Always test your JavaScript programs on all systems and in all web browsers for which they are intended.



Good Programming Practice 1.2

Read the documentation for the JavaScript version you are using to access JavaScript's rich collection of features.



Error-Prevention Tip 1.2

Your computer and JavaScript interpreter are good teachers. If you are not sure how a feature works, even after studying the documentation, experiment and see what happens. Study each error or warning message and adjust the code accordingly.

JavaScript was created by Netscape, the company that created the first widely successful web browser. Both Netscape and Microsoft have been instrumental in the standardization of JavaScript by ECMA International (formerly the European Computer Manufacturers Association) as ECMAScript. In Chapters 16–17, we discuss Adobe Flash, which uses another scripting language named ActionScript. ActionScript and JavaScript are converging in the next version of the JavaScript standard (JavaScript 2/ECMA Script version 4) currently under development by ECMA. This will result in a universal client scripting language, greatly simplifying web application development.

1.12 Browser Portability

Ensuring a consistent look and feel on client-side browsers is one of the great challenges of developing web-based applications. Currently, a standard does not exist to which software developers must adhere when creating web browsers. Although browsers share a common set of features, each browser might render pages differently. Browsers are available in many versions and on many different platforms (Microsoft Windows, Apple Macintosh, Linux, UNIX, etc.). Vendors add features to each new version that sometimes result in cross-platform incompatibility issues. Clearly it is difficult to develop web pages that render correctly on all versions of each browser. In this book we develop web applications that execute on both the Internet Explorer 7 and Firefox 2 browsers.



Portability Tip 1.4

The web is populated with many different browsers, which makes it difficult for authors and web application developers to create universal solutions. The W3C is working toward the goal of a universal client-side platform.

1.13 C, C++ and Java

C

The **C** language was developed by Dennis Ritchie at Bell Laboratories. C was implemented in 1972. C initially became known as the development language of the UNIX operating system. Today, virtually all new major operating systems are written in C and/or C++.

C++

Bjarne Stroustrup developed **C++**, an extension of C, in the early 1980s. C++ provides a number of features that “spruce up” the C language, but more importantly, it provides capabilities for object-oriented programming. C++ is a hybrid language: It is possible to program in either a C-like style (procedural programming), in which the focus is on actions, or an object-oriented style, in which the focus is on objects, or both. C and C++ have influenced many subsequent programming languages, such as Java, C#, JavaScript and PHP, each of which has a syntax similar to C and C++.

Java

Microprocessors are having a profound impact in intelligent consumer electronic devices. Recognizing this, Sun Microsystems in 1991 funded an internal corporate research project code-named Green to provide software for these devices. The project resulted in the development of a C++-based language that its creator, James Gosling, called Oak after an oak tree outside his window at Sun. It was later discovered that there already was a computer language called Oak. When a group of Sun people visited a local coffee shop, the name **Java** was suggested and it stuck.

The Green project ran into some difficulties. The marketplace for intelligent consumer electronic devices did not develop in the early 1990s as quickly as Sun had anticipated. The project was in danger of being canceled. By sheer good fortune, the World Wide Web exploded in popularity in 1993, and Sun saw the immediate potential of using Java to add **dynamic content** (e.g., interactivity, animations and the like) to web pages. This breathed new life into the project.

Sun formally announced Java at an industry conference in May 1995. Java garnered the attention of the business community because of the phenomenal interest in the web. Java is now used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (e.g., cell phones, pagers and personal digital assistants) and for many other purposes.

1.14 BASIC, Visual Basic, Visual C++, C# and .NET

The **BASIC** (Beginner's All-purpose Symbolic Instruction Code) programming language was developed in the mid-1960s at Dartmouth College as a means of writing simple programs. BASIC's primary purpose was to familiarize novices with programming techniques. Microsoft's Visual Basic language, introduced in the early 1990s to simplify the development of Microsoft Windows applications, has become one of the most popular programming languages in the world.

Microsoft's latest development tools are part of its corporatewide strategy for integrating the Internet and the web into computer applications. This strategy is implemented in Microsoft's **.NET platform**, which provides the capabilities developers need to create computer applications that can execute on computers distributed across the Internet. Microsoft's three primary programming languages are **Visual Basic** (based on the original BASIC), **Visual C++** (based on C++) and **Visual C#** (a relatively new language based on C++ and Java that was developed expressly for the .NET platform). Developers using .NET can write software components in the language they are most familiar with, then form applications by combining those components with others written in any .NET language.

1.15 Software Technologies

In this section, we discuss some software engineering topics and buzzwords that you'll hear in the software development community. We've created Resource Centers on most of these topics, with many more on the way.

Agile Software Development is a set of methodologies that try to get software implemented quickly with fewer resources than previous methodologies. Check out the Agile Alliance (www.agilealliance.org) and the Agile Manifesto (www.agilemanifesto.org).

Refactoring involves reworking code to make it clearer and easier to maintain while preserving its functionality. It's widely employed with agile development methodologies. Many refactoring tools are available to do major portions of the reworking automatically.

Design patterns are proven architectures for constructing flexible and maintainable object-oriented software. The field of design patterns tries to enumerate those recurring patterns, encouraging software designers to reuse them to develop better-quality software with less time, money and effort.

Game programming. The computer game business is larger than the first-run movie business. College courses and even majors are now devoted to the sophisticated software techniques used in game programming. Chapter 17 discusses building interactive games with Adobe Flash CS3. Also check out our Resource Centers on Game Programming, C++ Game Programming and Programming Projects.

Open source software is developed in a way unlike the proprietary development that dominated software's early years and remains strong today. With open source development, individuals and companies contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge. Open source code generally gets scrutinized by a much larger audience than proprietary software, so bugs may be removed faster. Open source also encourages more innovation. Sun recently open sourced Java. Some organizations you'll hear a lot about in the open source community are the Eclipse Foundation (the Eclipse IDE is popular for C++ and Java software development), the Mozilla Foundation (the creators of the Firefox browser), the Apache Software Foundation (the creators of the Apache web server) and SourceForge (which provides the tools for managing open source projects and currently has over 150,000 open source projects under development).

Linux is an open source operating system and one of the greatest successes of the open source movement. Apache is the most popular open source web server. **MySQL** (see Chapters 22–24) is an open source database management system. **PHP** (see Chapter 23) is the most popular open source server-side "scripting" language for developing Internet-based applications. **LAMP** is an acronym for the set of open source technologies that many developers used to build web applications—it stands for Linux, Apache, MySQL and PHP (or Perl or Python—two other scripting languages used for similar purposes).

Ruby on Rails (see Chapter 24) combines the scripting language Ruby with the Rails web application framework developed by the company 37Signals. Their book, *Getting Real*, is a must read for today's web application developers; read it free at getting-real.37signals.com/toc.php. Many Ruby on Rails developers have reported significant productivity gains over using other languages when developing database-intensive web applications.

Software has generally been viewed as a product; most software still is offered this way. If you want to run an application, you buy a software package from a software vendor. You then install that software on your computer and run it as needed. As new versions of the software appear, you upgrade your software, often at significant expense. This process can become cumbersome for organizations with tens of thousands of systems that must be maintained on a diverse array of computer equipment. With **Software as a Service (SaaS)**, the software runs on servers elsewhere on the Internet. When those servers are updated, all clients worldwide see the new capabilities; no local installation is needed. You access the service through a browser—these are quite portable, so you can run the same applications

on different kinds of computers from anywhere in the world. Salesforce.com, Google, Microsoft and 37Signals all offer SaaS.

1.16 Notes about Internet & World Wide Web How to Program, 4/e

In 1995, we saw an explosion of interest in the Internet and the World Wide Web. We immersed ourselves in these technologies, and a clear picture started to emerge in our minds of the next direction to take in writing textbooks for introductory programming courses. **Electronic commerce**, or **e-commerce**, as it is typically called, began to dominate the business, financial and computer industry news. This was a reconceptualization of the way business should be conducted. We still wanted to teach programming principles, but we felt compelled to do it in the context of the technologies that businesses and organizations need to create Internet-based and web-based applications. With this realization, the first edition of *Internet & World Wide Web How to Program* was born and published in December of 1999.

Internet & World Wide Web How to Program, Fourth Edition teaches programming languages and programming language principles. In addition, we focus on the broad range of technologies that will help you build real-world Internet-based and web-based applications that interact with other applications and with databases. These capabilities allow you to develop the kinds of enterprise-level, distributed applications popular in industry today.

You'll learn computer programming and basic principles of computer science and information technology. You also will learn proven software development methods—top-down stepwise-refinement, functionalization and object-based programming. Our primary programming language is JavaScript, a compact language that is especially designed for developing Internet- and web-based applications. Chapters 6–13 present a rich discussion of JavaScript and its capabilities, including dozens of complete examples followed by screen images that illustrate typical program inputs and outputs.

After you learn programming principles from the detailed JavaScript discussions, we present condensed treatments of four other popular Internet/web programming languages for building the server side of Internet- and web-based client/server applications. Chapter 23 introduces the popular PHP scripting language. Chapter 24 introduces Ruby, the scripting language used with the Ruby on Rails framework for rapid development of database-driven web applications. In Chapter 25, we discuss ASP.NET 2.0—Microsoft's technology for server-side scripting. ASP.NET pages can be written in Visual Basic and C#; we code ASP.NET pages using Visual Basic. In Chapters 26–27, we discuss JavaServer Faces, which uses the Java programming language. Finally, in Chapter 28, we discuss web services (using examples in both Java and ASP.NET).

1.17 Web Resources

www.deitel.com/

Check this site frequently for updates, corrections and additional resources for all Deitel & Associates, Inc., publications.

www.deitel.com/resourcecenters.html

Check out the complete list of Deitel Resource Centers, including numerous programming, open source, Web 2.0 and Internet business topics.

netforbeginners.about.com

The About.com *Internet for Beginners* guide provides valuable resources for further exploration of the history and workings of the Internet and the web.

www.learnthenet.com/english/index.html

Learn the Net is a website containing a complete overview of the Internet, the web and the underlying technologies. The site contains much information appropriate for novices.

www.w3.org

The World Wide Web Consortium (W3C) website offers a comprehensive description of web technologies. For each Internet technology with which the W3C is involved, the site provides a description of the technology, its benefits to web designers, the history of the technology and the future goals of the W3C in developing the technology.

Summary

Section 1.1 Introduction

- In an era of steadily rising costs, computing costs have been decreasing dramatically because of rapid developments in both hardware and software technologies.
- Technologies such as Extensible HyperText Markup Language (XHTML), JavaScript, Flash, Flex, Dreamweaver and Extensible Markup Language (XML) are used to build the portions of web-based applications that reside on the client side (i.e., the portions of applications that typically run on web browsers such as Firefox or Microsoft's Internet Explorer).
- Technologies such as web servers, databases, ASP.NET, PHP, Ruby on Rails and JavaServer Faces are used to build the server side of web-based applications. These parts of applications typically run on "heavy-duty" computer systems on which organizations' business-critical websites reside.

Section 1.2 What Is a Computer?

- A computer is a device capable of performing computations and making logical decisions at speeds billions of times faster than human beings can.
- A computer processes data under the control of sets of instructions called computer programs, which guide it through orderly sets of actions specified by computer programmers.
- The various devices that comprise a computer system are referred to as hardware.
- The computer programs that run on a computer are referred to as software.

Section 1.3 Computer Organization

- The input unit is the "receiving" section of the computer. It obtains information from input devices and places it at the disposal of the other units for processing.
- The output unit is the "shipping" section of the computer. It takes information processed by the computer and places it on output devices to make it available for use outside the computer.
- The memory unit is the rapid-access, relatively low-capacity "warehouse" section of the computer. It retains information that has been entered through the input unit, making it immediately available for processing when needed, and retains information that has already been processed until it can be placed on output devices by the output unit.
- The arithmetic and logic unit (ALU) is the "manufacturing" section of the computer. It is responsible for performing calculations and making decisions.
- The central processing unit (CPU) is the "administrative" section of the computer. It coordinates and supervises the operation of the other sections.

- The secondary storage unit is the long-term, high-capacity “warehousing” section of the computer. Programs or data not being used by the other units are normally placed on secondary storage devices (e.g., disks) until they are needed, possibly hours, days, months or even years later.

Section 1.4 Machine Languages, Assembly Languages and High-Level Languages

- Any computer can directly understand only its own machine language, which generally consists of strings of numbers ultimately reduced to 1s and 0s that instruct the computer to perform its most elementary operations.
- English-like abbreviations form the basis of assembly languages. Translator programs called assemblers convert assembly-language programs to machine language.
- Compilers translate high-level language programs into machine-language programs. High-level languages contain English words and conventional mathematical notations.
- Interpreter programs directly execute high-level language programs, eliminating the need to compile them into machine language.

Section 1.5 History of the Internet and World Wide Web

- In the late 1960s, ARPA, the Advanced Research Projects Agency of the U.S. Department of Defense rolled out the blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions. ARPA then proceeded to implement the ARPANET, the predecessor to today’s Internet.
- The World Wide Web allows computer users to locate and view multimedia-based documents (i.e., documents with text, graphics, animations, audios or videos) on almost any subject.
- In 1989, Tim Berners-Lee of CERN began to develop the World Wide Web and several communication protocols that form the backbone of the web.
- Web use exploded with the availability in 1993 of the Mosaic browser, which featured a user-friendly graphical interface. Marc Andreessen, whose team at NCSA developed Mosaic, went on to found Netscape, the company that many people credit with initiating the explosive Internet economy of the late 1990s.

Section 1.6 World Wide Web Consortium (W3C)

- In October 1994, Tim Berners-Lee founded the World Wide Web Consortium (W3C)—an organization devoted to developing nonproprietary, interoperable technologies for the web.

Section 1.7 Web 2.0

- Web 2.0 companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis, etc.).
- Web 1.0 (the state of the web through the 1990s and early 2000s) was focused on a relatively small number of companies and advertisers producing content for users to access.
- Web 2.0 embraces an architecture of participation—a design that encourages user interaction and community contributions.
- Using the collective intelligence—the concept that a large diverse group of people will create smart ideas—communities collaborate to develop open source software that many people believe is better and more robust than proprietary software.
- Rich Internet Applications (RIAs) are being developed using technologies (such as Ajax) that have the look and feel of desktop software, enhancing a user’s overall experience.
- Web services, inexpensive computers, abundant high-speed Internet access, open source software and many other elements have inspired new, exciting, lightweight business models that people can launch with only a small investment.

Section 1.8 Personal, Distributed and Client/Server Computing

- Apple Computer popularized personal computing.
- IBM's Personal Computer quickly legitimized personal computing in business, industry and government organizations, where IBM mainframes were heavily used.
- Although early personal computers were not powerful enough to timeshare several users, these machines could be linked together in computer networks, sometimes over telephone lines and sometimes in local area networks (LANs) within an organization. This led to the phenomenon of distributed computing.
- Today's personal computers are as powerful as the million-dollar machines of just a few decades ago, and information is shared easily across computer networks.

Section 1.9 Hardware Trends

- Moore's Law states that the power of hardware doubles every two years, while the price remains essentially the same.

Section 1.10 Key Software Trend: Object Technology

- Objects are essentially reusable software components that model real-world items.
- Not until object-oriented programming became widely used in the 1990s did software developers feel they had the tools to make major strides in the software development process.
- Object technology is a packaging scheme that helps us create meaningful software units.
- A key problem with procedural programming is that the program units do not effectively mirror real-world entities, so these units are not particularly reusable.
- With object technology, the software entities created (called classes), if properly designed, tend to be reusable on future projects. Using libraries of reusable componentry can greatly reduce effort required to implement certain kinds of systems.
- Some organizations report that the key benefit object-oriented programming gives them is the production of software which is more understandable, better organized and easier to maintain, modify and debug.

Section 1.11 JavaScript: Object-Based Scripting for the Web

- JavaScript is an object-based scripting language with strong support for proper software engineering techniques.
- JavaScript was created by Netscape. Both Netscape and Microsoft have been instrumental in the standardization of JavaScript by ECMA International as ECMAScript.

Section 1.12 Browser Portability

- Ensuring a consistent look and feel on client-side browsers is one of the great challenges of developing web-based applications.

Section 1.13 C, C++ and Java

- C initially became known as the development language of the UNIX operating system. Today, virtually all new major operating systems are written in C and/or C++.
- C++ provides a number of features that "spruce up" the C language, but more importantly, it provides capabilities for object-oriented programming.
- Java is used to create dynamic and interactive content for web pages, develop enterprise applications, enhance web-server functionality, provide applications for consumer devices and more.

Section 1.14 BASIC, Visual Basic, Visual C++, C# and .NET

- The BASIC programming language was developed in the mid-1960s at Dartmouth College. Its primary purpose was to familiarize novices with programming techniques.
- Microsoft's Visual Basic was introduced in the early 1990s to simplify the process of developing Microsoft Windows applications.
- Microsoft has a corporatewide strategy for integrating the Internet and the web into computer applications. This strategy is implemented in Microsoft's .NET platform.
- The .NET platform's three primary programming languages are Visual Basic, Visual C++ and Visual C#.
- .NET developers can write software components in their preferred language, then form applications by combining those components with components written in any .NET language.

Section 1.15 Software Technologies

- Agile Software Development is a set of methodologies that try to get software implemented quickly with fewer resources than previous methodologies.
- Refactoring involves reworking code to make it clearer and easier to maintain while preserving its functionality.
- Design patterns are proven architectures for constructing flexible and maintainable object-oriented software.
- Open source development allows individuals and companies to contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.
- With Software as a Service (SaaS), the software runs on servers elsewhere on the Internet, rather than on the desktop.

Terminology

actions	Dale Dougherty
Agile Software Development	data
architecture of participation	data structure
arithmetic and logic unit (ALU)	decision
ARPANET	design pattern
assemblers	distributed computing
assembly language	DOM (Document Object Model)
bandwidth	dynamic content
BASIC	electronic commerce (e-commerce)
C	electronic mail (e-mail)
C++	function
central processing unit (CPU)	game programming
class	hardware
client side	high-level languages
client/server computing	HTML (HyperText Markup Language)
collective intelligence	HTTP (Hypertext Transfer Protocol)
compilers	input device
computer	input unit
computer program	Internet
computer programmer	interpreter
CSS	IP (Internet Protocol)

Java	primary memory
JavaScript	refactoring
LAMP	Ruby on Rails
library	scripting
lightweight business models	scripting language
Linux	secondary storage unit
local area networks (LANs)	server side
logical unit	server
machine dependent	software
machine language	Software as a Service (SaaS)
memory	structured programming
memory unit	structured systems analysis and design
method	supercomputer
Moore's Law	tagging
multiprocessor	TCP (Transmission Control Protocol)
MySQL	TCP/IP
.NET platform	translation
O'Reilly Media	translator program
object code	Visual Basic
object-based programming	Visual C#
object-oriented programming	Visual C++
open source software	Web 1.0
output devices	Web 2.0
output unit	workstation
packet	World Wide Web
packet switching	World Wide Web Consortium (W3C)
personal computer	XHTML (Extensible HyperText Markup Language)
PHP	XML (Extensible Markup Language)
platform	

Self-Review Exercises

1.1 Fill in the blanks in each of the following:

- a) The company that popularized personal computing was _____.
- b) The computer that made personal computing legitimate in business and industry was the _____.
- c) Computers process data under the control of sets of instructions called computer _____.
- d) The six key logical units of the computer are the _____, _____, _____, _____, _____, and the _____.
- e) The three classes of languages discussed in the chapter are _____, _____, and _____.
- f) The programs that translate high-level language programs into machine language are called _____.
- g) _____, or labeling content, is another key part of the collaborative theme of Web 2.0.
- h) With Internet applications, the desktop evolves to the _____.
- i) _____ involves reworking code to make it clearer and easier to maintain while preserving its functionality.
- j) With _____ development, individuals and companies contribute their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.

- k) The _____ was the predecessor to the Internet.
- l) The information-carrying capacity of a communications medium like the Internet is called _____.
- m) The acronym TCP/IP stands for _____.

1.2 Fill in the blanks in each of the following statements.

- a) The _____ allows computer users to locate and view multimedia-based documents on almost any subject over the Internet.
- b) _____ founded an organization—called the World Wide Web Consortium (W3C)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web.
- c) _____ are reusable software components that model items in the real world.
- d) In a typical client/server relationship, the _____ requests that some action be performed and the _____ performs the action and responds.

Answers to Self-Review Exercises

1.1 a) Apple. b) IBM Personal Computer. c) programs. d) input unit, output unit, memory unit, arithmetic and logic unit, central processing unit, secondary storage unit. e) machine languages, assembly languages and high-level languages. f) compilers. g) Tagging. h) webtop. i) Refactoring. j) open source. k) ARPANET. l) bandwidth. m) Transmission Control Protocol/Internet Protocol.

1.2 a) World Wide Web. b) Tim Berners-Lee. c) Objects. d) client, server.

Exercises

1.3 Categorize each of the following items as either hardware or software:

- a) CPU
- b) ALU
- c) input unit
- d) an editor program

1.4 Fill in the blanks in each of the following statements:

- a) Which logical unit of the computer receives information from outside the computer for use by the computer? _____.
- b) The process of instructing the computer to solve specific problems is called _____.
- c) What type of computer language uses English-like abbreviations for machine-language instructions? _____.
- d) Which logical unit of the computer sends information that has already been processed by the computer to various devices so that the information may be used outside the computer? _____.
- e) Which logical units of the computer retain information? _____.
- f) Which logical unit of the computer performs calculations? _____.
- g) Which logical unit of the computer makes logical decisions? _____.
- h) The level of computer language most convenient for you to write programs quickly and easily is _____.
- i) The only language that a computer directly understands is called that computer's _____.
- j) Which logical unit of the computer coordinates the activities of all the other logical units? _____.

- k) Some organizations report that the key benefit _____ gives them is the production of software which is more understandable, better organized and easier to maintain, modify and debug.
- l) Web 2.0 embraces an _____—a design that encourages user interaction and community contributions.
- m) _____ is the concept that a large diverse group of people will create smart ideas.

1.5 Fill in the blanks in each of the following statements (based on Section 1.15, Software Technologies):

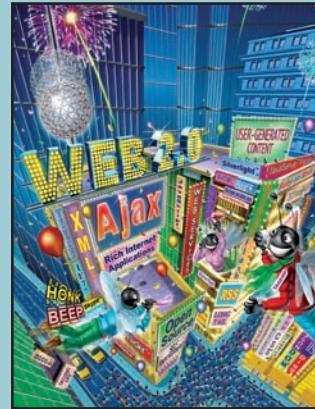
- a) The open source database management system used in LAMP development is _____.
- b) A key advantage of Software as a Service (SaaS) is _____.
- c) _____ are proven architectures for constructing flexible and maintainable object-oriented software.
- d) _____ is the most popular open source server-side “scripting” language for developing Internet-based applications.

1.6 What is the relationship between JavaScript and ECMAScript?

1.7 In this chapter, we discussed a few popular Web 2.0 businesses including MySpace, Flickr, YouTube and Wikipedia. Identify a Web 2.0 business and describe why it fits the Web 2.0 business model.

2

Web Browser Basics: Internet Explorer and Firefox



Give us the tools, and we will finish the job.

—Sir Winston Churchill

OBJECTIVES

In this chapter you will learn:

- To understand the Microsoft Internet Explorer 7 (IE7) and Mozilla Firefox 2 (FF2) web browsers' capabilities.
- To use IE7 and FF2 to search the information available on the World Wide Web.
- To customize a browser according to your own needs and preferences.
- To understand the differences among various browsers.

We must learn to explore all the options and possibilities that confront us in a complex and rapidly changing world.

—J. William Fulbright

Outline

- 2.1 Introduction to the Internet Explorer 7 and Firefox 2 Web Browsers
- 2.2 Connecting to the Internet
- 2.3 Internet Explorer 7 and Firefox 2 Features
- 2.4 Customizing Browser Settings
- 2.5 Searching the Internet
- 2.6 Keeping Track of Your Favorite Sites
- 2.7 File Transfer Protocol (FTP)
- 2.8 Online Help
- 2.9 Other Web Browsers
- 2.10 Wrap-Up
- 2.11 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

2.1 Introduction to the Internet Explorer 7 and Firefox 2 Web Browsers

The Internet is an essential medium for communicating and interacting with people worldwide. The need to publish and share information has fueled the rapid growth of the web. **Web browsers** are software programs that allow users to access the web's rich content. Whether for business or personal use, millions of people use web browsers to access the tremendous amount of information available on the web and to share or exchange this content with other users. The **www** portion of the Internet, which we encounter often in this chapter, is made up of hyperlinked documents written in XHTML and rich media.

Popular web browsers at the time of publication are Microsoft's *Internet Explorer*, Mozilla's *Firefox*, Apple's *Safari* and Opera Software's *Opera*. This chapter focuses on the features of Internet Explorer (IE7) and Firefox 2 (FF2), which are the most widely used of these browsers. All examples in this book are supported by both IE7 and FF2.

2.2 Connecting to the Internet

A computer alone is not enough to access the Internet. In addition to web browser software, the computer needs specific hardware and a connection to an Internet Service Provider to view web pages. This section describes the components that enable Internet access.

First, a computer must have a **modem** or **network card**. A modem is hardware that enables a computer to connect to a **network** via phone lines. A modem converts data to audio tones and transmits the data over phone lines. A network card, also called a **network interface card (NIC)**, is hardware that allows a computer to connect to the Internet through a network or a high-speed Internet connection, such as a **local area network (LAN)**, **cable modem** or **Digital Subscriber Line (DSL)**.

After ensuring that a computer has a modem or a network card (most computers come with one or both of these), the next step is to register with an **Internet Service Provider (ISP)**. Computers connect to an ISP using a modem and phone line, or via a NIC using a LAN, DSL or cable modem. The ISP connects computers to the Internet. Most college and university campuses offer network connections, and many communities now offer wireless access. If a network connection is not available, then popular commercial ISPs,

such as AOL (www.aol.com), Comcast (www.comcast.net), Earthlink (www.earthlink.net), Verizon (www.verizon.com), Microsoft Network (www.msn.com) and NetZero (www.netzero.net) are alternatives.

Bandwidth and cost are two considerations when deciding which commercial ISP service to use. Bandwidth refers to the amount of data that can be transferred through a communications medium in a fixed amount of time. Different ISPs offer different types of high-speed connections, called **broadband connections**—which include DSL, cable modem and **Integrated Services Digital Network (ISDN)**—and slower **dial-up connections**. Each connection type has a different bandwidth and cost to users.

Broadband is a category of high-bandwidth Internet service that is most often provided to home users by cable television and telephone companies. DSL is a broadband service that allows computers to be connected at all times to the Internet over existing phone lines, without interfering with telephone services. DSL requires a special modem provided by the ISP. Like DSL, cable modems enable the computer to be connected to the Internet at all times. Cable modems transmit data over the cables that bring television to homes and businesses. Unlike DSL, the bandwidth is shared by many users. This sharing can reduce the bandwidth available to each person when many use the system simultaneously. ISDN provides Internet service over either digital or standard telephone lines. ISDN requires specialized hardware, called a **terminal adapter (TA)**, which is usually obtained from the ISP.

Dial-up service uses an existing telephone line. If a computer is connected to the Internet, the user usually cannot receive voice calls during this time. If the voice calls do connect, the Internet connection is interrupted. To prevent this, users may choose to install an extra phone line dedicated to Internet service.

Fiber optics are replacing traditional metal cables in many computer networks due to their greater bandwidth and mechanical advantages that provide a better signal. Though their popularity is currently limited by the high cost of materials and installation, consistent improvements in the industry will allow fiber optic cables to become a key element of the communications industry in the near future.

Once a computer is connected to a network, the user must choose a web browser for navigating the Internet. Internet Explorer is preinstalled on all Windows machines, and your version can be updated at www.microsoft.com/ie. Firefox can be downloaded at www.mozilla.com/firefox, and can be installed on many different operating systems. When installing this browser, select **Custom** when prompted for a setup type, and ensure that the **DOM Inspector** option is selected in the next screen. Doing so will ensure that you have additional Firefox functionality that we discuss in Chapter 12.

2.3 Internet Explorer 7 and Firefox 2 Features

A web browser is software that allows the user to view certain types of Internet files in an interactive environment. Figure 2.1 shows the Deitel Home Page using Internet Explorer 7 web browser, and Fig. 2.2 uses Firefox 2. The **URL (Uniform Resource Locator)** <http://www.deitel.com> is found in the **Address** bar in IE7, and the **Location** bar in FF2. The URL specifies the address (i.e., location) of the web page displayed in the browser window. Each web page on the Internet is associated with a unique URL. URLs usually begin with **http://**, which stands for **Hypertext Transfer Protocol (HTTP)**, the standard protocol (or set of communication rules) for transferring web documents over the Internet. URLs of websites that handle private information, such as credit card numbers, often

2.3 Internet Explorer 7 and Firefox 2 Features

31

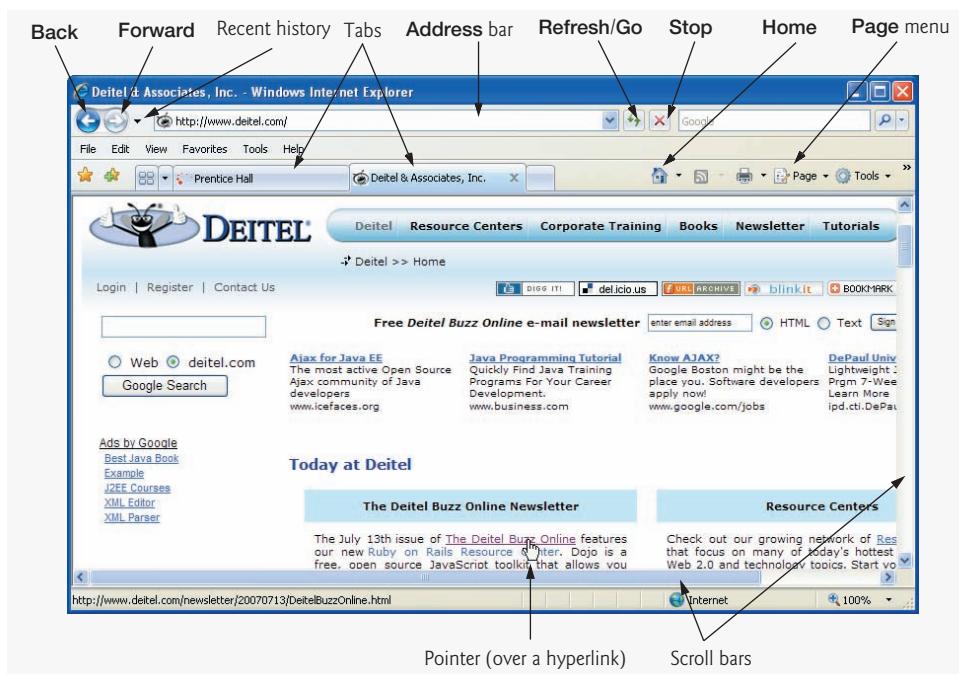


Fig. 2.1 | Deitel® website in Internet Explorer 7.

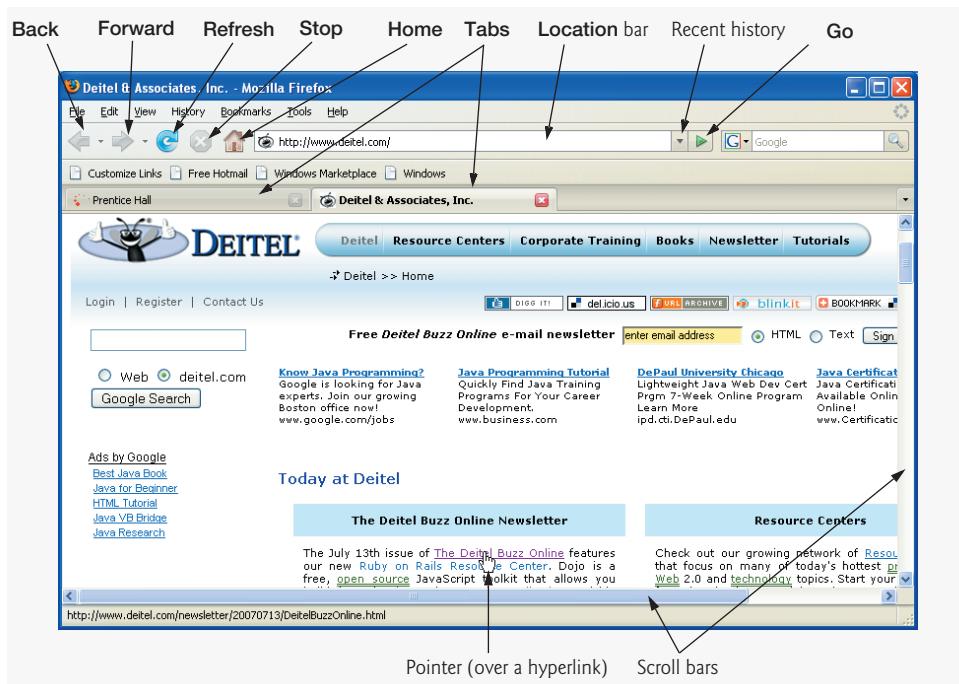


Fig. 2.2 | Deitel® website in Firefox 2.

begin with `https://`, the abbreviation for **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)**, the standard for transferring encrypted data on the web.

There are several techniques for navigating between URLs. You can click the **Address** field and type a web page's URL, then press *Enter* or click **Go** (in IE7, this is the same button as **Refresh**) to request the web page located at that URL. For example, to visit Yahoo!'s website, type `www.yahoo.com` in the **Address** bar and press the *Enter* key. Clicking **Refresh** loads the latest version of the web page from the current website. IE7 and FF2, as well as most other popular browsers, add the `http://` prefix to the website name because HTTP is the default protocol used for the web.

Hyperlinks

Another way to navigate the web is via visual elements on web pages called **hyperlinks** that, when clicked, load a specified web document. Both images and text may be hyperlinked. When the mouse pointer hovers over a hyperlink, the default arrow pointer changes into a hand with the index finger pointing upward. Often hyperlinked text appears underlined and as in different color from regular text in a web page. Originally used as a publishing tool for scientific research, hyperlinks are widely used to reference sources, or sites that have more information on a particular topic. The paths created by hyperlinking create the effect of the “web.”

Hyperlinks can reference other web pages, e-mail addresses, files and more. If a hyperlink's URL is in the form `mailto:emailAddress`, clicking the link loads your default e-mail program and opens a **message window** addressed to the specified e-mail address. Note that these hyperlinks are generally displayed on the screen as just the e-mail address or the recipient's name.

If a hyperlink references a file that the browser is incapable of displaying, the browser prepares to **download** the file, and generally prompts the user for information about how the file should be stored. When a file is downloaded, it is copied onto the user's computer. Programs, documents, images and sound files are all examples of downloadable files.

Tabbed Browsing

Many browsers, including IE7 and FF2, provide **tabbed browsing**. Holding down the *Ctrl* key and pressing the letter *T* while in the IE7 or FF2 browser opens another tab in the same window, allowing the user to browse multiple pages without cluttering the desktop with many windows. [Note: For Mac users, all references to the *Ctrl* key in this chapter's shortcuts should be replaced with the *Command* key.] Also, pressing *Ctrl* while clicking a link will open the requested page in a new tab. Clicking on the tabs switches between the different pages in the browser, and web pages are then accessed normally. Using tabs is an excellent way to keep the browser organized when viewing multiple pages at once.

Using the History Feature

IE7 and FF2 maintain a **History** list of previously visited URLs in chronological order. This feature allows users to return to recently visited websites easily. The history feature can be accessed several different ways. The simplest and most frequently used method is to click the **Forward** and **Back** buttons located at the top of the browser window (see Fig. 2.1). The **Back** button reloads into the browser the page you last visited. Assuming that you used the **Back** button to view previously visited pages, the **Forward** button would load the next URL from the history into the browser. The keyboard shortcut for **Forward** is `<Alt>` and

the *Right Arrow* key or just *Shift* and *Backspace*, and the shortcut for **Back** is *<Alt>* and the **Left Arrow** key or simply *Backspace*.

In IE7, the user can view the last and next nine web pages visited and the current page by clicking the down arrows immediately to the right of the **Forward** button; the user can then request one of the recently viewed pages by clicking the title of the page in the drop-down list. In FF2, there are separate menus to the right of both the **Forward** and the **Back** buttons. Each displays the previous and following fifteen pages in the history, respectively. Note that these methods only display history results from the browser's current **session**, which is the period when the browser remains open. In IE7 and FF2, there is a menu to the right of the address bar which displays a longer but more basic history of visited sites (it does not include any URLs accessed through hyperlinks), including websites that were visited in previous sessions. Another way to display sites from a previous session is to use **History**.

Selecting **History** from the down-arrow menu in IE7, or clicking the **History** menu, then the **Show In Sidebar** option in FF2, divides the browser window into two sections: the **History** window (on the left) and the content window (Figs. 2.3–2.4). In IE7, clicking the yellow star icon in the upper left of the window, then selecting the **History** option, displays a similar menu. By default, the **History** window lists the URLs visited in the past twenty days in IE7 and nine days in FF2.

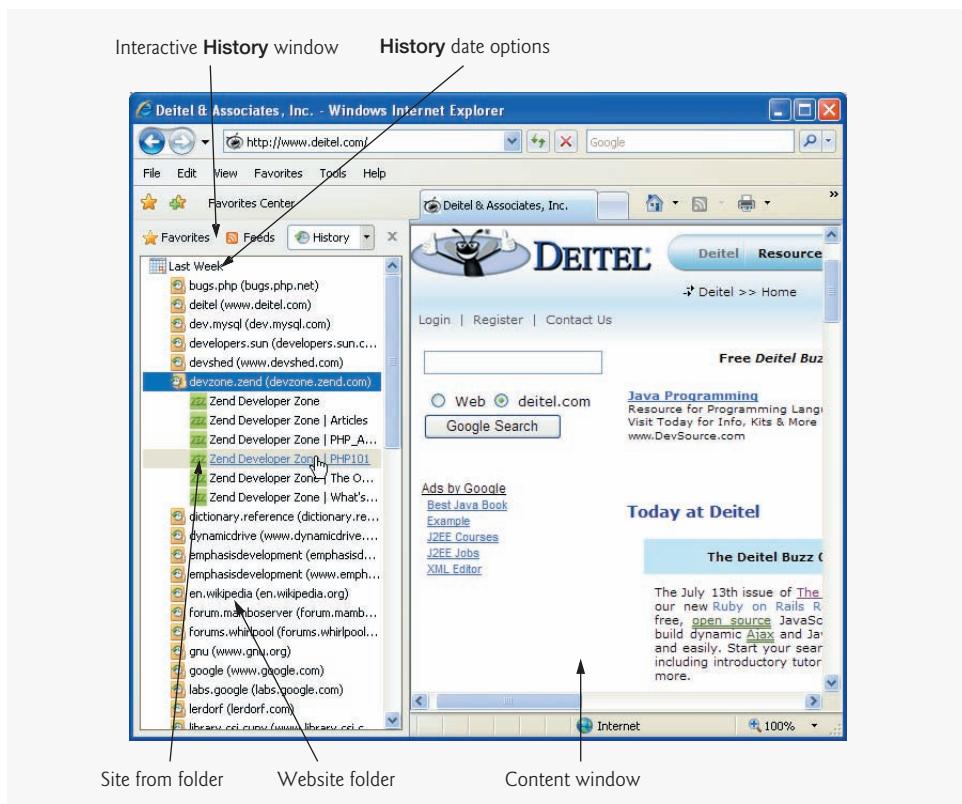


Fig. 2.3 | The **History** menu in Internet Explorer 7.

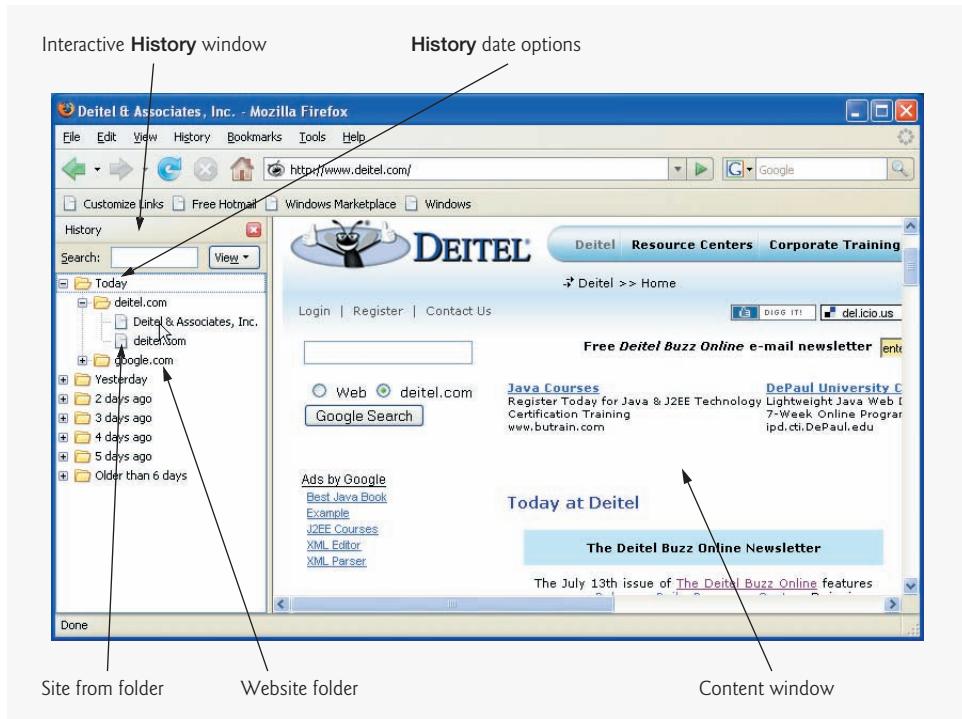


Fig. 2.4 | The **History** menu in Firefox 2.

The **History** window contains heading levels ordered chronologically. Within each time frame (e.g., **Today**) headings are alphabetized by website name (although the organization can be changed clicking the **History** drop-down menu in IE7 or the **View** drop-down menu of FF2, both located in the **History** window). This window is useful for finding previously visited websites without having to remember the exact URL. Selecting a URL from the **History** window loads the web page into the content window.

AutoComplete

URLs from the history can be displayed in a drop-down list when a user types a URL into the **Address** bar. This feature is called **AutoComplete**. Any URL from this drop-down list can be selected with the mouse to load the web page at that URL into the browser (Fig. 2.5).

Off-Line Browsing

For some users, such as those with dial-up connections, maintaining a connection for long periods of time may not be practical. For this reason, web pages can be saved directly to the computer's hard drive for **off-line** browsing (i.e., browsing while not connected to the Internet). Select **Save As...** in IE7, or **Save Page As...** in FF2, both from the **File** menu to save a web page and all its components, including the images. [Note: To display the **File** menu in IE7, press the **Alt** key.] This option is also available under the **Page** menu in IE7 (Fig. 2.1). Individual images from a website can also be saved by clicking the image with the right mouse button and selecting **Save Picture As...** (IE7) or **Save Image As...** (FF2) from the displayed **context menu** (Fig. 2.6).

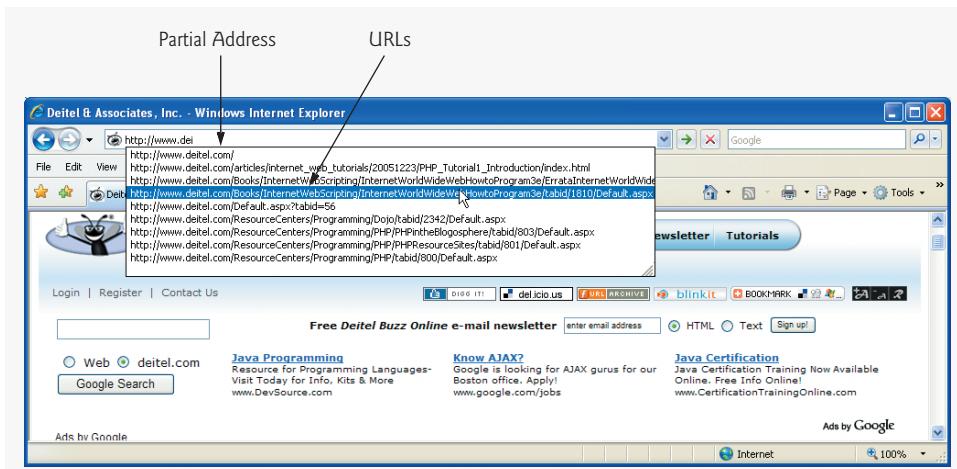


Fig. 2.5 | AutoComplete suggests possible URLs when given a partial address.

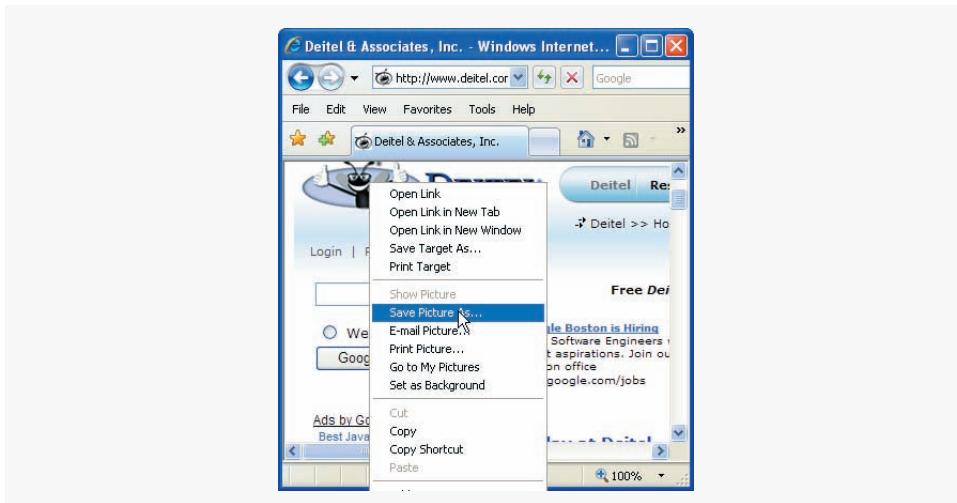


Fig. 2.6 | Saving a picture from a website.

Downloads

As mentioned earlier, files from the Internet may be copied to a computer's hard drive by a process called **downloading**. This section discusses the types of documents commonly downloaded from the Internet and techniques for downloading them. [Note: You should always be cautious when downloading files from the Internet, as they may contain viruses. Only download from sites that you trust.]

Some common Internet downloads are **applications** (i.e., software that performs specific functions, such as word processing), **plug-ins** and **extensions**. Plug-ins are specialized pieces of software that help the browser support additional content types. An example of an IE7 and FF2 plug-in is the **Acrobat Reader** from **Adobe, Inc.** (www.adobe.com/products/acrobat/readstep2.html), which allows users to view **PDF (Portable Document Format)**.

ment Format) documents that otherwise cannot be rendered by the browser. Another popular plug-in allows the browser to render **Flash** content, which adds audio, video and animation effects to a website. To view sites enabled with Flash, download the Adobe Flash Player plug-in at www.adobe.com/products/flashplayer. Microsoft's rich media plug-in, **Silverlight**, is available for download at silverlight.net/GetStarted. (Both Flash and Silverlight are discussed in much greater depth in Chapters 16, 17 and 19). Normally the browser prompts the user to download a plug-in when one is needed. Plug-ins may also be downloaded from CNET (www.download.com). This site has a large, searchable index and database of many plug-in programs available for download.

Extensions are add-ons that enhance the preexisting functionality of the browser. Examples of extensions include blog editors, universal uploaders and various translation dictionaries and tools. Many IE7 add-ons can be found at www.ieaddons.com, and FF2 add-ons can be browsed and downloaded at <https://addons.mozilla.org>.

Viewing Source Code

Clicking on the **View** menu followed by the **Source** option in IE7 and **Page Source** in FF2 allows you to view the **source code**, or the original code written to create the web page you are viewing. Generally, source code is easy for humans to read and interpret, and allows the viewer to understand how the programmer created the page. For example, if an element of a web page does not display properly, examining the source code can help to inform the user what the programmer was trying to do. Examining source code is a useful tool for debugging your own code, or for learning how web developers create some of the elements you see on the web.

2.4 Customizing Browser Settings

Browsers have many settings that determine how sites are displayed, how security measures are applied and how outputs are rendered. Most of these settings are located in the **Internet Options** dialog (Fig. 2.7) in the **Tools** menu of IE7, and in **Options** under the **Tools** menu in FF2 in Windows (Fig. 2.8) [Note: For Firefox on a Mac, this is called the **Preferences** menu.]. The default settings are usually adequate for normal browsing, but these settings can be customized to suit each user's preferences.

Some privacy settings for IE7 and FF2 can be set under the **Privacy** tab. In IE7 there are six levels of privacy. The most lenient level permits the downloading of **cookies** (text files that are placed on the computer by websites to retain or gather information about the user); the most strict level blocks all cookies from all websites and constantly updates a report to the user about browsing privacy. Using this level may prevent certain websites from working correctly. In FF2 the **Privacy** tab displays options about how data is remembered in the system and when cookies should be accepted.

Security options for both browsers can be found under the **Security** tab. The browsers' options are significantly different, but both allow you to specify how much information you want to hide from unfamiliar sites, as well as how much of the site's content you would like to block from your own computer. Both browsers allow you to distinguish between trusted sites and the rest of the web, and to browse safe sites with lower security settings.

A personal home page can be specified under the **General** tab in IE7 and **Main** in FF2. The home page is the web page that loads when the browser is first opened and appears when the **Home** button at the top of the browser window is clicked (Figs. 2.1–2.2).

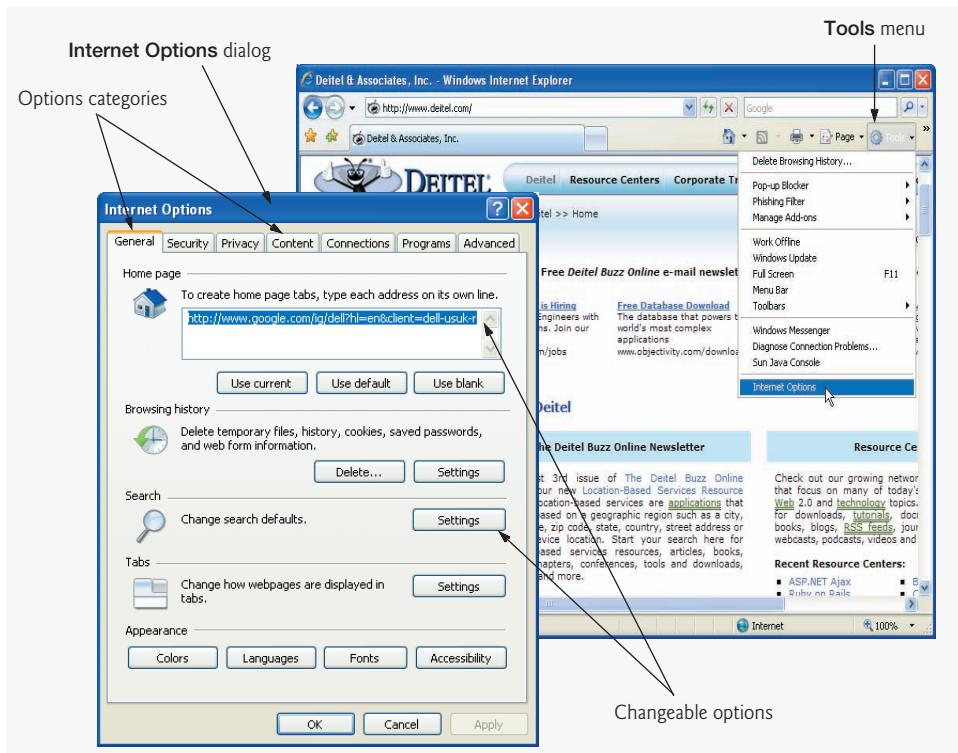


Fig. 2.7 | Internet Options in Internet Explorer 7.

History options also may be adjusted in this category. By clicking the **Settings** button in the **Browsing history** section of the **General** tab in IE7, or the **Network** option in the **Advanced** tab of FF2, the amount of disk space to be reserved for the web page **cache** can be set. The cache is an area of temporary storage that a browser designates for saving web pages for rapid future access. When a page is viewed that has been visited recently, IE7 and FF2 check whether they already have some elements on that page (such as images) saved in the cache, to reduce download time. Having a large cache can considerably speed up web browsing, whereas having a small cache saves disk space. Caching can sometimes cause problems, because Internet Explorer and Firefox do not always check to ensure that a cached page is the same as the latest version residing on the web server. Holding down the *Ctrl* key and pressing *F5* in either browser, or pressing *Ctrl*, *Shift* and *R* in FF2, remedies this problem by forcing the browser to retrieve the latest version of the web page from the website. Once the **Internet Options** are set, click **OK** in both browsers.

2.5 Searching the Internet

The Internet provides a wealth of information on virtually any topic. The sheer volume of information on the web can make it difficult for users to find specific information. To help users locate information, many websites provide **search engines** that explore the Internet and maintain searchable records containing information about website content. This section explains how search engines work and discusses two types of search engines.

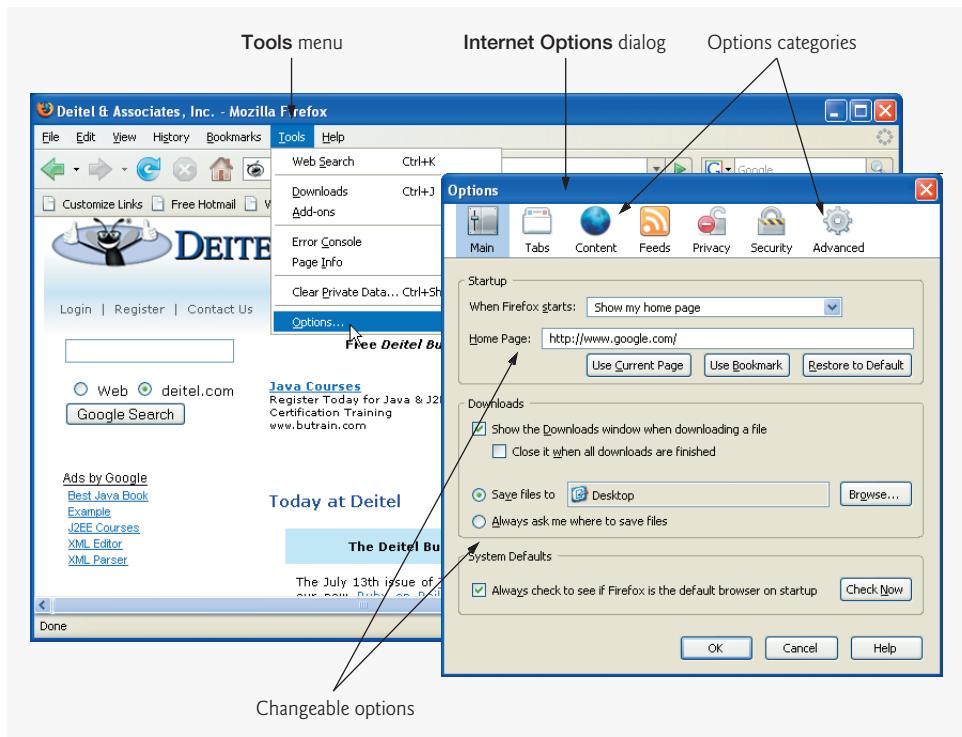


Fig. 2.8 | Options in Firefox 2.

Search engines such as Google (www.google.com), Yahoo! (www.yahoo.com), MSN (www.msn.com), AltaVista (www.altavista.com) and Ask.com (www.ask.com) store information in data repositories called **databases** that facilitate quick information retrieval. When the user enters a word or phrase, the search engine returns a list of hyperlinks to sites that satisfy the search criteria. Each search engine site has different criteria for narrowing searches, such as publishing date, language and relevance. Using multiple search engines may provide the best results in finding the desired content quickly. Sites such as MetaCrawler (www.metacrawler.com) use **metasearch engines**, which do not maintain databases. Instead, they send the search criteria to other search engines and aggregate the results. Many web browsers, including IE7 and FF2 (Figs. 2.9–2.10.), have a built-in search box placed in the window that can be used to browse the web. In both browsers, the user can choose which search engine to use by clicking the down-arrow menu (Fig. 2.9–2.10.).

Search engines can also be used to help resolve programming errors. There are many websites that contain documentation about specific functions, how to use them correctly and related common errors. Putting a function name or error message into a search engine can often help a programmer discover where a mistake may have been made in the code. Also, websites such as www.thescripts.com allow users to post specific programming questions that can be answered by other programmers. Other websites like this one, as well as communities for specific languages, can be found using search engines.

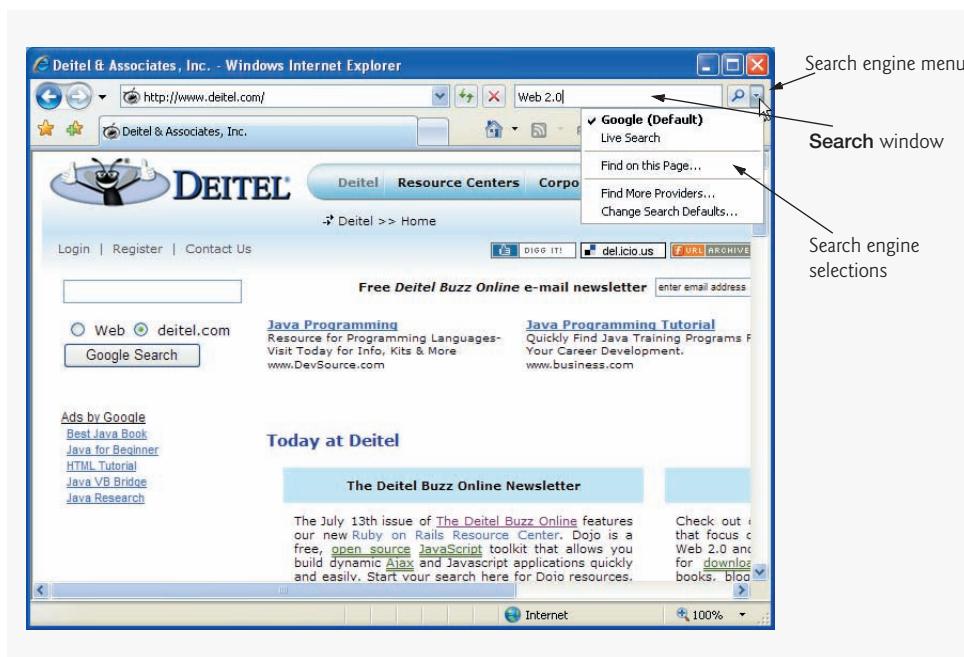


Fig. 2.9 | Searching the Internet with Internet Explorer 7.

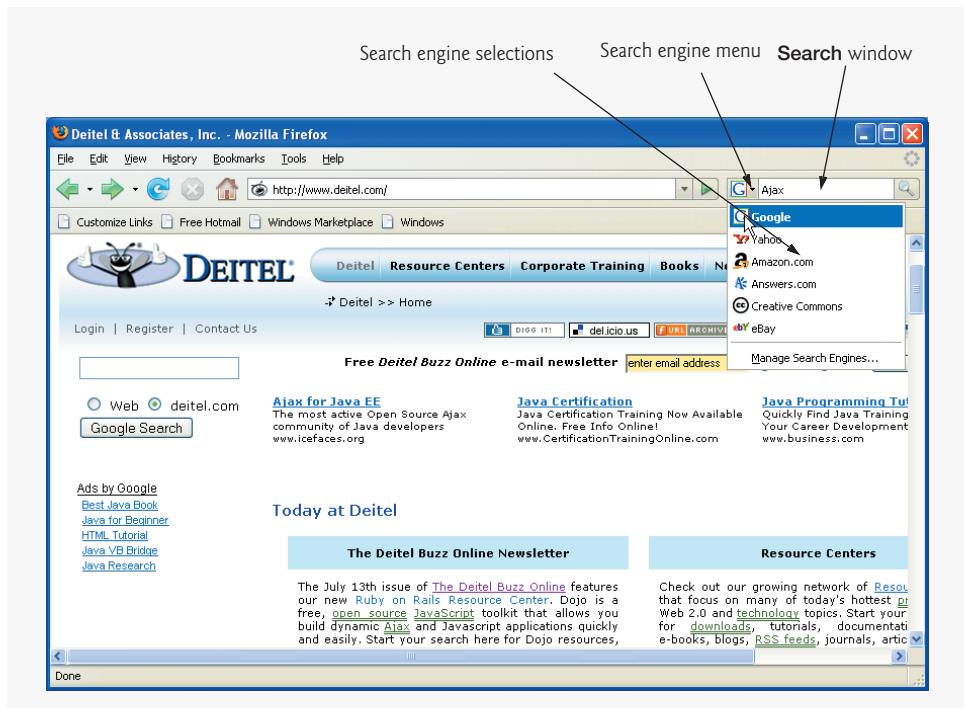


Fig. 2.10 | Searching the Internet with Firefox 2.

2.6 Keeping Track of Your Favorite Sites

As users browse the web, they often visit certain sites repeatedly and may want to record their URL and title. IE7 provides a feature called **favorites** for bookmarking (keeping track of) such sites (Fig. 2.11). Any page's URL can be added to the list of favorites using the **Favorites** menu's **Add to Favorites...** command, or by pressing the yellow star and green plus icon in the upper left corner of the window. A **Favorites** window can also be accessed by clicking the yellow star icon on the toolbar and clicking the **Favorites** option. Favorites can be accessed at any time by selecting them with the mouse from the **Favorites** menu. Favorites can be categorized and grouped into folders in the **Organize Favorites** dialog (displayed when **Organize Favorites...** is selected from the **Favorites** menu). These folders appear as submenus in the **Favorites** menu. The **Organize Favorites** dialog also allows users to rename, delete and move favorites between folders.

FF2 has a similar feature called **bookmarks**, which can be added with the **Bookmark This Page...** option in the **Bookmark** menu and used the same way that **Favorites** are described in this section (Fig. 2.12). Most browsers have their own version of **Favorites** or **Bookmarks**.

2.7 File Transfer Protocol (FTP)

The **File Transfer Protocol (FTP)** is a set of rules by which computers transfer data, especially large files, over the Internet. An FTP site's URL begins with `ftp://` rather than `http://`, and can also be accessed either with the web browser or software that supports FTP. **Filezilla** is a popular, open source FTP client for Windows that functions outside a

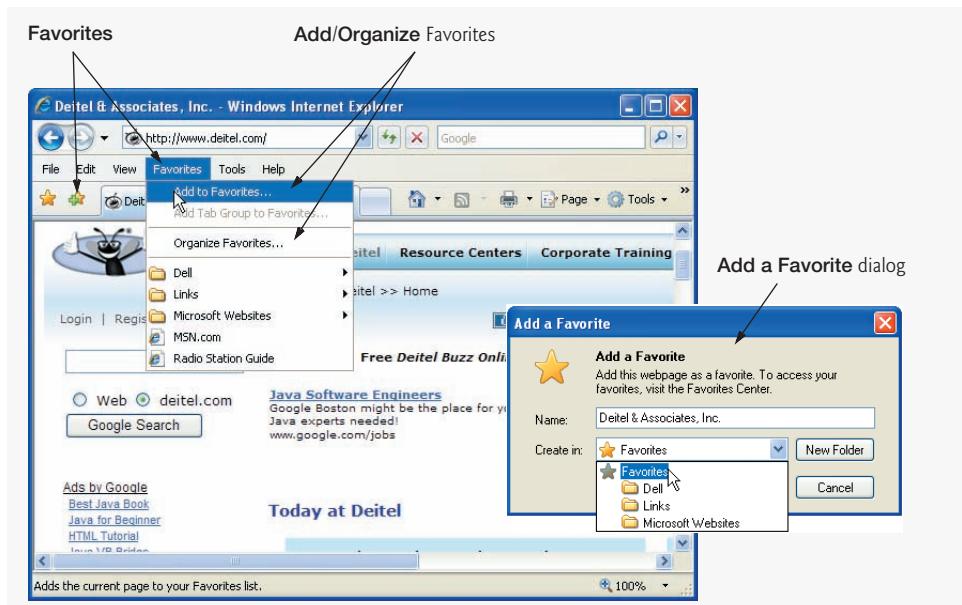


Fig. 2.11 | The **Favorites** menu helps organize frequently visited websites in Internet Explorer 7.

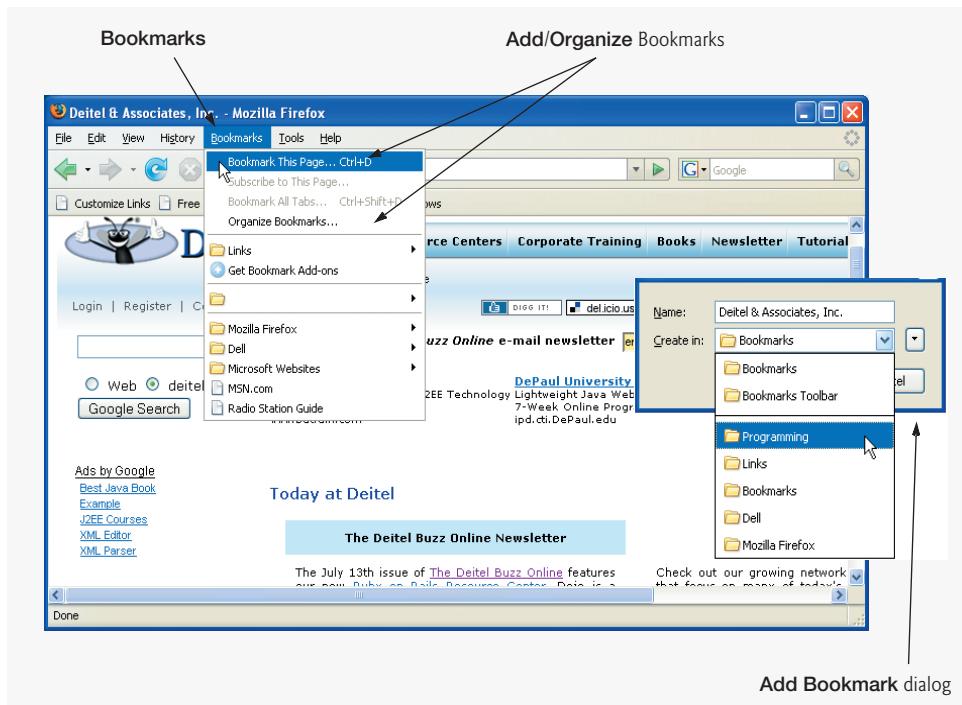


Fig. 2.12 | The Bookmarks menu helps organize frequently visited websites in Firefox 2.

web browser. It can be downloaded for free from <http://filezilla.sourceforge.net>. FF2 also has an extension, FireFTP, that adds the functionality of an FTP client to the browser. This add-on is available at <http://fireftp.mozdev.org>.

When your browser is pointed to an FTP site, the contents of the specified site directory appear in the window, and can be browsed as though they were files on the local computer. Files are downloaded by clicking their icons and following the browser's download instructions, or by dragging the file or folder with the mouse onto the desktop or into another directory. Windows users may copy and paste the URL into the address bar of the **My Computer** window, called the Windows Explorer (Fig. 2.13), which has a particularly straightforward interface for FTP. Windows Explorer can be accessed from the **Start** menu, or by clicking the **Page** menu, then selecting **Open FTP Site in Windows Explorer** in IE7.

When accessing an FTP site, the user may or may not be prompted for login information. Many FTP sites allow **anonymous FTP access**, where any user is permitted to view and download files. If login is required, the username is set by default to *anonymous*, and the user either is prompted for an e-mail address or should put an e-mail address in the password field. The browser sends the user's e-mail address and name to the website for tracking and information purposes. Other FTP sites contain directories with **restricted access**—only users with authorized usernames and passwords are permitted to access such directories. When a user is trying to enter a restricted-access FTP directory, a **Log On As** dialog like the one in Fig. 2.13 is displayed, prompting the user for login information.

Transferring a file from the local machine to another location on the Internet is called **uploading** and can be accomplished using the FTP protocol. Files can be transferred from

the local machine (your computer) to the remote machine (server), with an FTP client. The specific instructions for each client are different, but almost all FTP clients allow you to upload, download and perform other file-managing tasks on your data. Understanding FTP is especially important for web developers, since uploading files to a web server is a necessary part of creating a website.

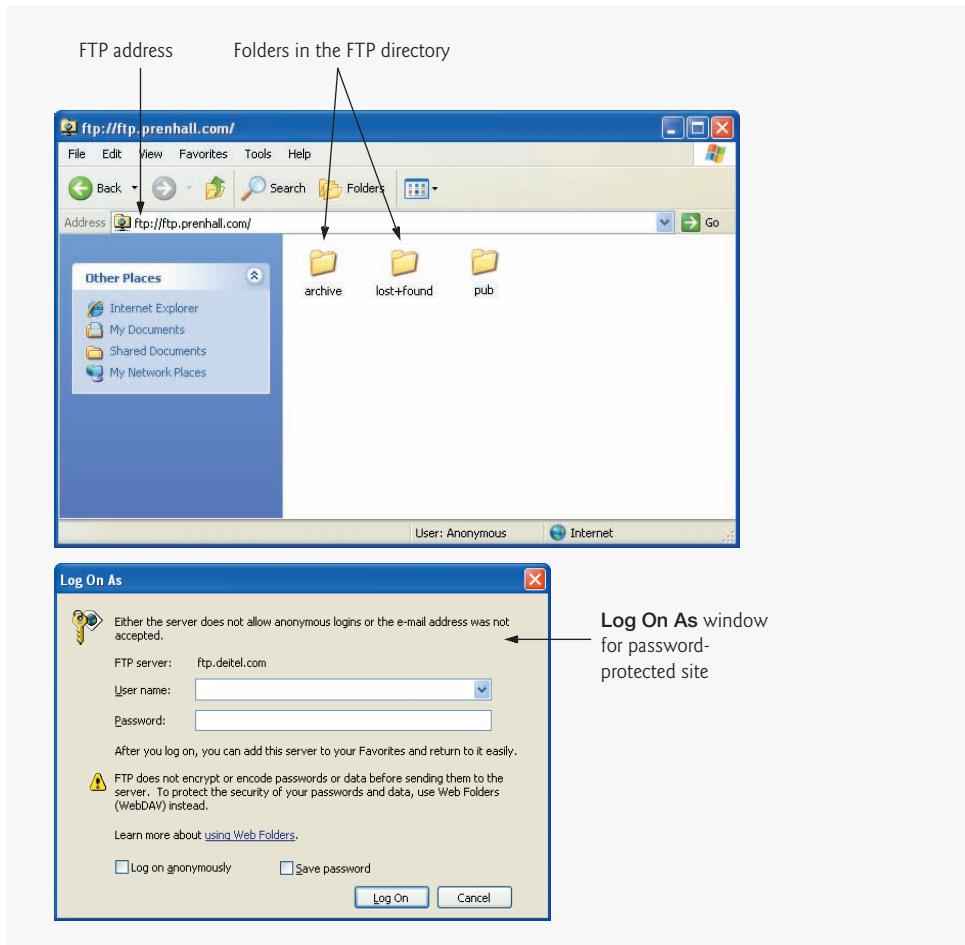


Fig. 2.13 | FTP site access.

2.8 Online Help

Web browsers are complex pieces of software with rich functionality. Although browser designers make every effort to produce user-friendly software, users still need time to familiarize themselves with each web browser and its particular features. Answers to frequently asked questions about using the web browser are included with FF2 and IE7, as well as most other browsers. This information is accessible through the built-in help feature available in the **Help** menu (Figs. 2.14–2.15).

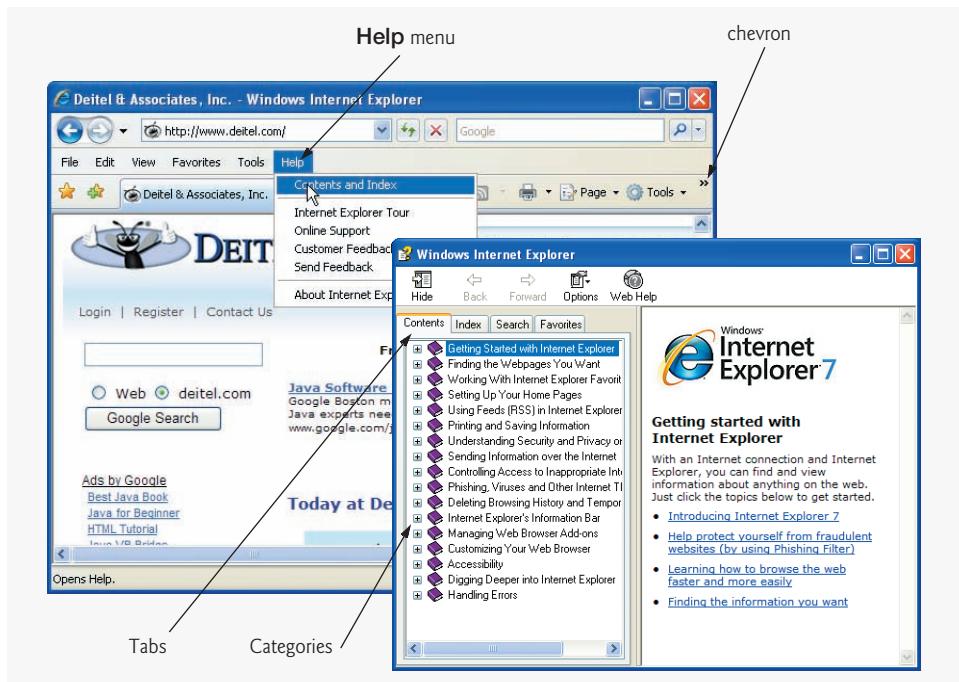


Fig. 2.14 | Internet Explorer 7 Help dialog.

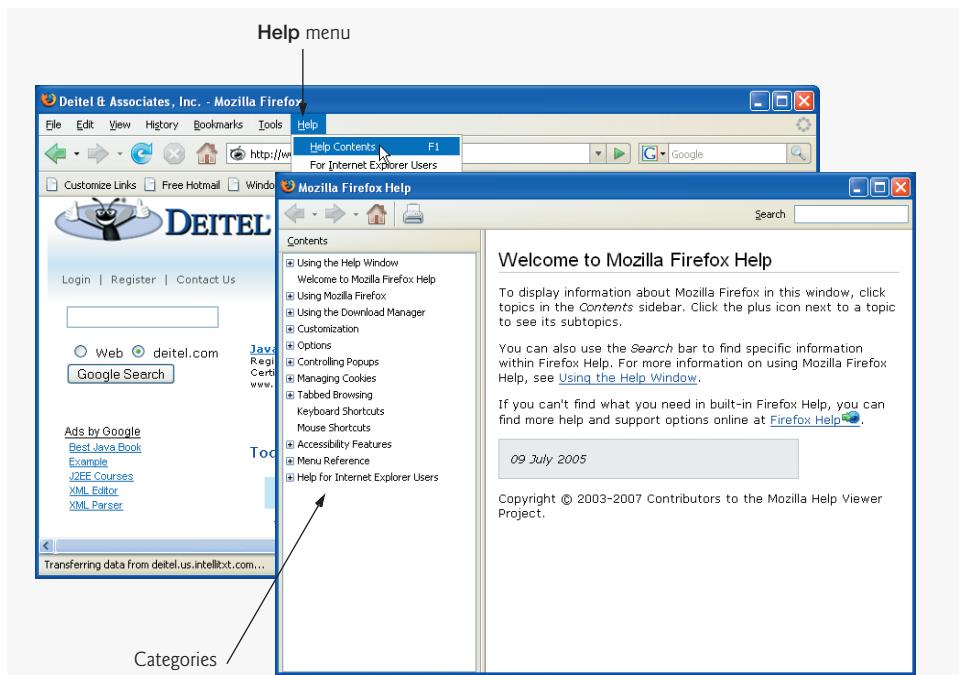


Fig. 2.15 | Firefox 2 Help dialog.

A good source for locating help about a specific feature is the **Contents and Index** menu item in IE7 and **Help Contents** in FF2, both accessible through the **Help** menu. IE's help menu can also be accessed by clicking the **Help** option on the toolbar's chevron. When these items are selected, the browser's help dialog is displayed. In IE7, the **Contents** tab organizes the help topics by category, the **Index** tab contains an alphabetical list of **Help** topics and the **Search** tab provides capabilities for searching the help documents. The **Favorites** tab allows users to maintain a list of frequently used help topics. FF2's **Help** window provides a search box and an expandable table of contents for browsing.

2.9 Other Web Browsers

Besides Internet Explorer and Firefox, many other web browsers are available, including *Opera* (www.opera.com) and *Safari* (www.apple.com/safari). All these browsers differ in functionality, performance and features. Also, they employ different HTML layout engines, which determine how a web page displays in a browser. Firefox 2 uses Gecko as its layout engine, Safari uses a modified version of the KHTML layout engine and Opera and IE7 have their own engines. Gecko and KHTML are both free and open source.

Opera, as well as IE7 and FF2, is a browser designed to be accessible to all users, including those with visual or mobility impairments. Opera software also recently developed a lightweight "Mini" version of the browser that runs effectively on mobile devices. Safari, originally created for Apple's Mac OS, features an elegantly simple interface and impressive speed, especially when executing JavaScript (discussed in Chapters 6–11). Because browsers use different HTML layout engines, they may display the same web page differently. Additionally, some capabilities supported in one browser may not be supported in another. The existence of different browser functionality and features makes cross-browser compatibility difficult to achieve.

2.10 Wrap-Up

In this chapter, we described the requirements for connecting to the Internet, and we introduced the basic features of Microsoft's Internet Explorer 7 and Mozilla's Firefox 2. You learned how to customize your browsing experience to fit your personal needs, and how to exchange data safely using the Internet. We also discussed how to use search engines to find information on the web and demonstrated how to keep track of useful sites. In the next chapter, we discuss the defining characteristics of Web 2.0 and how it has changed the way that users interact with the Internet.

2.11 Web Resources

www.deitel.com/ie7

www.deitel.com/firefox

The Deitel Internet Explorer and Firefox Resource Centers contain links to some of the best resources about these browsers on the web. There you'll find categorized links to Internet Explorer and Firefox downloads and add-ons, keyboard shortcuts, glossaries, code compatibility issues, blogs, forums, podcasts and more. Also check out the range of tutorials on different aspects of these browsers.

Summary

Section 2.1 Introduction to the Internet Explorer 7 and Firefox 2 Web Browsers

- Web browsers are software programs that allow users to access the web's rich multimedia content.
- The two most popular web browsers are Microsoft's *Internet Explorer* and Mozilla's *Firefox*.

Section 2.2 Connecting to the Internet

- A computer alone is not enough to access the Internet. In addition to web browser software, the computer needs specific hardware and a connection to an Internet Service Provider.
- A modem enables a computer to connect to the Internet. A modem converts data to audio tones and transmits the data over phone lines. A network card, also called a network interface card (NIC), allows a computer to connect to the Internet through a network or a high-speed Internet connection, such as a LAN, cable modem or a Digital Subscriber Line (DSL).
- Bandwidth refers to the amount of data that can be transferred through a communications medium in a fixed amount of time. Different ISPs offer different types of high-speed connections, called broadband connections.
- Broadband is a category of high-bandwidth Internet service that is most often provided by cable television and telephone companies to home users.
- DSL is a broadband service that allows computers to be connected at all times to the Internet over existing phone lines, without interfering with voice services.
- Cable modems enable the computer to be connected to the Internet at all times. Cable modems transmit data over the cables that bring television to homes and businesses.
- ISDN provides Internet service over either digital or standard telephone lines. ISDN requires specialized hardware, called a terminal adapter (TA), which is usually obtained from the ISP.
- Fiber optics are a cable alternative that provide greater bandwidth and a better signal, but their popularity is limited by high cost.

Section 2.3 Internet Explorer 7 and Firefox 2 Features

- A URL is the address of a web page. Each web page is associated with a unique URL. URLs usually begin with `http://`, which stands for Hypertext Transfer Protocol (HTTP), the industry standard protocol for transferring web documents over the Internet.
- URLs of websites that handle private information, such as credit card numbers, often begin with `https://`, the abbreviation for Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS), the standard for transferring encrypted data over the Internet.
- Several techniques are available for navigating between different URLs. A user can click the **Address** field and type a web page's URL. The user can then press *Enter* or click **Go** to request the web page located at that URL.
- Another way to navigate the web is via visual elements on web pages called hyperlinks that, when clicked, load a specified web document. Both images and text serve as hyperlinks.
- Hyperlinks can reference other web pages, e-mail addresses and files. If a hyperlink is a `mailto:e-mailaddress`, clicking the link loads your default e-mail program and opens a message window addressed to the specified recipient's e-mail address.
- Tabbed browsing allows you to browse multiple pages without cluttering the desktop with many windows.
- When a file is downloaded, it is copied onto the user's computer. Programs, documents, images and sound files are all downloadable files.

- IE7 and FF2 maintain a list of previously visited URLs in chronological order, called the history.
- The **History** window contains heading levels ordered chronologically. Within each time frame headings are alphabetized by site directory name. This window is useful for finding previously visited sites without having to remember the exact URL.
- URLs from the history are displayed in a drop-down list when a user types a URL into the **Address** bar. This feature is called AutoComplete. Any URL from this drop-down list can be selected with the mouse to load the web page at that URL into the browser.
- Web pages can be saved directly to the computer's hard drive for off-line browsing. Select **Save As...** (IE7) or **Save Page As...** (FF2) from the **File** menu at the top of the browser window to save a web page and all its components.
- Individual images from a website can be saved by clicking the image with the right mouse button and selecting **Save Picture As...** (IE7) or **Save Image As...** (FF2) from the displayed context menu.
- Plug-ins are specialized pieces of software that enable the browser to support additional types of content. Normally the browser prompts the user to download a plug-in when a plug-in is needed.
- Extensions enhance the preexisting functionality of the browser.
- Clicking the **View** menu followed by the **Source** option in IE7 or **Page Source** in FF2 allows you to view the source code, or the original code written to create the web page you are viewing.

Section 2.4 Customizing Browser Settings

- IE7 and FF2 have many settings that determine how sites are displayed, how security measures are applied and how outputs are rendered. Many of these can be accessed through the **Tools** menu, then **Internet Options** in IE7, or **Options** in FF2.
- The privacy level for IE7 can be set under the **Privacy** tab of the **Internet Options** dialog. There are six levels of privacy. The most lenient level permits downloading and cookies; the strictest level renders a constant flow of alerts and alarms about browsing security and might prevent certain websites from working correctly.
- Privacy settings for FF2 can be found under the **Privacy** tab of **Options**, which displays options about how data is remembered in the system and when cookies should be accepted.
- Other security options can be found for both browsers under the **Security** tab.
- A personal home page can be specified under the **General** tab of the **Internet Options** dialog in IE7, and **Main** under **Options** in FF2. The home page is the web page that loads when the browser is first opened and appears when the **Home** button at the top of the browser window is clicked.
- History and cache options may be adjusted in the **General** tab of the **Internet Options** dialog by clicking the **Settings...** button in IE7, and the cache can be adjusted in the **Network** option in the **Advanced** tab of FF2. The amount of disk space to be reserved for web-page cache can be set here.

Section 2.5 Searching the Internet

- Search engines explore the Internet and maintain searchable records containing information about websites.
- Metasearch engines do not maintain databases. Instead, they send the search criteria to other search engines and aggregate the results. IE7 and FF2 have built-in search boxes next to the **Address** bar with several different search engines which can be selected by the user.
- Search engines are helpful tools for finding solutions to programming problems.

Section 2.6 Keeping Track of Your Favorite Sites

- As users browse the web, they often visit certain sites repeatedly. Internet Explorer 7 provides a feature called **Favorites** for bookmarking such sites, and Firefox 2 has a similar tool called **Book-**

marks. Sites can be remembered and organized under the **Favorites** menu in IE7 and the **Bookmarks** menu in FF2.

Section 2.7 File Transfer Protocol (FTP)

- FTP (file transfer protocol) is an older protocol for transferring information, especially large files, over the Internet. An FTP site's URL begins with `ftp://` rather than `http://`, and can be accessed through the browser or by any software that supports FTP.
- FTP sites with anonymous access allow any user access. Many FTP sites have restricted access; only users with authorized usernames and passwords are permitted to access such sites.
- Transferring a file from the local machine to another location on the Internet is called uploading and can be accomplished using the FTP protocol.

Section 2.8 Online Help

- The **Help** menu in the browsers allows the user to search or browse for answers to common questions and solutions to problems with the software.

Section 2.9 Other Web Browsers

- Many other browsers are available for download, each with its own set of features and advantages. Two of these browsers are Opera and Safari.

Terminology

Address bar	History
Adobe Acrobat Reader	home page
Adobe Flash Player	HTML layout engine
anonymous FTP	hyperlink
applications	Hypertext Transfer Protocol (HTTP)
AutoComplete	Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Back button	Integrated Services Digital Network (ISDN)
bandwidth	Internet Explorer 7 (IE7)
bookmark	Internet Options
broadband connection	Internet Service Provider (ISP)
cable modem	metasearch engine
cache	modem
context menu	Mozilla Firefox
cookie	network
database	network card
dial-up connection	network interface card (NIC)
Digital Subscriber Line (DSL)	off-line browsing
download	Opera
extensions	plug-in
Favorites	Portable Document Format (PDF)
fiber optics	Privacy
file transfer	restricted access
File Transfer Protocol (FTP)	Safari
Filezilla	Search
Firefox 2 (FF2)	search engine
Flash Content	security level
Forward button	session
Gecko	sharing
Help menu	

Silverlight source code	uploading
terminal adaptor (TA)	web browser
Uniform Resource Locator (URL)	web editor

Windows Internet Explorer

Self-Review Exercises

- 2.1** Fill in the blanks in each of the following statements:
- The two most popular web browsers are _____ and _____.
 - A browser is used to view files on the _____.
 - The location of a file on the Internet is called its _____.
 - The element in a web page that, when clicked, causes a new web page to load is called a(n) _____; when your mouse passes over this element, the mouse pointer changes into a(n) _____ in IE7 and FF2.
 - IE7 and FF2 keep of a list of visited URLs called the _____.
 - You can save an image from a web page by right clicking the image and selecting _____ in IE7 or _____ in FF2.
 - The feature of IE7 and FF2 that provides options for completing URLs is called _____.
 - The feature that enables the user to save URLs of frequently visited sites is called _____ in IE7 or _____ in FF2.
- 2.2** State whether each of the following is *true* or *false*. If the statement is *false*, explain why.
- Fiber optics cables have a better signal than traditional metal cables, but an inferior bandwidth.
 - It is not possible to view web pages when not connected to the Internet.
 - Search engines can be used to help resolve programming errors.
 - The cache is an area on the hard drive that is used for saving web pages for rapid future access.
 - FTP is a popular Internet mechanism by which files are uploaded and downloaded.
 - You can access any FTP site by logging in with the user name **anonymous**.

Answers to Self-Review Exercises

- 2.1** a) Internet Explorer, Firefox. b) Internet and the web. c) URL. d) hyperlink, hand. e) history. f) **Save Picture As...., Save Image As....** g) AutoComplete. h) **Favorites, Bookmarks**.
- 2.2** a) False. Fiber optic cables have a greater bandwidth than metal ones. b) False. Web pages saved to the hard drive can be viewed using off-line browsing. c) True. d) True. e) True. f) False. Many FTP sites are restricted and do not admit the general public.

Exercises

- 2.3** Spell out the following acronyms, and include a brief description of each:
- HTTP
 - FTP
 - URL
 - DSL
 - PDF
 - ISP
- 2.4** Use your browser's FTP capability to access `ftp.cdrom.com` and `ftp.deitel.com`. Observe what happens in both cases, and, if the site can be accessed, list the directory output.

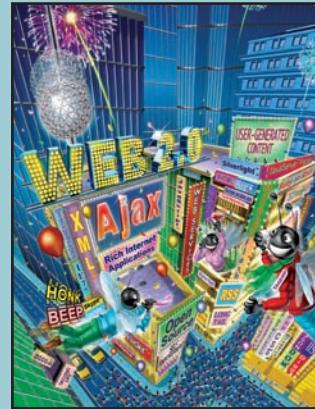
2.5 Go to www.ieaddons.com and browse the various extensions and plug-ins that can be installed into Internet Explorer. Choose one to install, and observe what capabilities are added as you browse the Internet.

2.6 Go to addons.mozilla.org and browse the various extensions and plug-ins that can be installed into Firefox. Choose one to install, and observe what capabilities are added as you browse the Internet.

2.7 Download and install the Opera (www.opera.com) and Safari (www.apple.com/safari) web browsers. Go to your favorite websites and try to observe any differences in speed, appearance and functionality.

3

Dive Into[®] Web 2.0



Network effects from user contributions are the key to market dominance in the Web 2.0 era.

—Tim O'Reilly

Link by link, click by click, search is building possibly the most lasting, ponderous, and significant cultural artifact in the history of humankind: the Database of Intentions.

—John Battelle, *The Search*

Web 2.0 is a massive social experiment...this is an opportunity to build a new kind of international understanding...citizen to citizen, person to person.

—Lev Grossman, *TIME*

One of the powerful things about networking technology like the Internet or the Web or the Semantic Web...is that the things we've just done with them far surpass the imagination of the people who invented them.

—Tim Berners-Lee, interviewed by Peter Moon, *IDG Now*

OBJECTIVES

In this chapter you will learn:

- The defining characteristics of Web 2.0.
- Why search is fundamental to Web 2.0.
- How Web 2.0 empowers the individual.
- The importance of collective intelligence and network effects.
- The significance and growth of blogging.
- Social networking, social media and social bookmarking.
- How tagging leads to folksonomies.
- How web services enable new applications to be quickly and easily “mashed up” from existing applications.
- Web 2.0 technologies.
- Web 2.0 Internet business and monetization models.
- The emerging Semantic Web (the “web of meaning”).

Outline

- 3.1 Introduction
- 3.2 What Is Web 2.0?
- 3.3 Search
- 3.4 Content Networks
- 3.5 User-Generated Content
- 3.6 Blogging
- 3.7 Social Networking
- 3.8 Social Media
- 3.9 Tagging
- 3.10 Social Bookmarking
- 3.11 Software Development
- 3.12 Rich Internet Applications (RIAs)
- 3.13 Web Services, Mashups, Widgets and Gadgets
- 3.14 Location-Based Services
- 3.15 XML, RSS, Atom, JSON and VoIP
- 3.16 Web 2.0 Monetization Models
- 3.17 Web 2.0 Business Models
- 3.18 Future of the Web
- 3.19 Wrap-Up
- 3.20 Where to Go for More Web 2.0 Information
- 3.21 Web 2.0 Bibliography
- 3.22 Web 2.0 Glossary

[Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

[Note: *Chapter 3, Dive Into® Web 2.0*, is also available as a free, frequently updated HTML-based e-book at <http://www.deitel.com/freeWeb20ebook/>. It is also available as a downloadable, fully-formatted PDF for a small fee. Check this site for the latest hyperlink-rich version. Many of the topics in this chapter are supplemented by extensive Resource Centers at <http://www.deitel.com/resourcecenters.html>. The e-book and the PDF link to the Resource Centers and other web resources for further study.]

3.1 Introduction

Chapter 1 presented basic computing concepts and the roles of several key technologies in developing distributed client/server applications for the Internet and the web. Chapter 2 discussed the capabilities of web browsers and how to use the latest versions of the two most popular browsers, Internet Explorer 7 and Firefox 2. Chapter 3 introduces the principles, applications, technologies, companies, business models and monetization strategies of Web 2.0.

When the Mosaic browser was introduced in 1993, the web exploded in popularity. It continued to experience tremendous growth throughout the 1990s—a period referred to as the “dot-com bubble”; that bubble burst in 2001. In 2003 there was a noticeable shift in how people and businesses were using the web and developing web-based applications.

The term **Web 2.0**—coined by **Dale Dougherty** of **O'Reilly Media**¹ in 2003 to describe this trend—became a major media buzzword, but few people really know what it means. Generally, Web 2.0 companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis, etc.). Web 2.0 was popularized by the annual O'Reilly Media Web 2.0 Summit (launched in 2004), in Tim O'Reilly's defining article on Web 2.0 entitled, "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software,"² and in John Musser and Tim O'Reilly's for-sale report, "Web 2.0 Principles and Best Practices."³

The growth of Web 2.0 can be attributed to some key factors. First, hardware keeps getting cheaper and faster, with memory capacities and speeds increasing at a rapid rate. Moore's Law states that the power of hardware doubles every two years, while the price remains essentially the same.⁴ This allows for development of applications with high demands that would have been previously unthinkable. Second, broadband Internet use has exploded—a Pew Internet study in March 2006 found 42% of American adults had high-speed Internet in their homes. Of the 35% of Internet users who had posted content online, 73% had broadband Internet.⁵ The abundance of digital media online would never have been possible without high-speed Internet. Third, the availability of abundant open source software (see Section 3.11) has resulted in cheaper (and often free) customizable software options. This makes it easier to start new Web 2.0 companies and greatly decreases the cost of failure. Fourth, unlike in **Web 1.0** (the state of the web through the 1990s and early 2000s), there are many easy-to-employ models available to monetize Web 2.0 businesses—immediately generating (modest amounts of) revenue allows for more stable growth of new companies.

Our information on the companies in this chapter comes from common knowledge, the company websites and the footnoted books and articles.

3.2 What Is Web 2.0?

In a sense, this entire chapter defines Web 2.0, but let's begin with a brief, one-section discussion. Web 1.0 was focused on a relatively small number of companies and advertisers producing content for users to access—some people called the web at the time the “brochure web.” Web 2.0 *involves* the user—not only is the content often created by users, but users help organize it, share it, remix it, critique it, update it, etc. One way to look at Web 1.0 is as a *lecture*, a small number of professors informing a large audience of students. In comparison, Web 2.0 is a *conversation*, with everyone having the opportunity to speak and share views.

-
1. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
 2. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
 3. Musser, J. and T. O'Reilly. *Web 2.0 Principles and Best Practices*. O'Reilly Media, Inc., 2006.
 4. Moore, G. "Cramming More Components onto Integrated Circuits." *Electronics*, April 1965 <ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf>.
 5. Horrigan, J. B. "Home Broadband Adoption 2006." *Pew Internet & American Life Project*, May 2006 <http://www.pewinternet.org/pdfs/PIP_Broadband_trends2006.pdf>.

Web 2.0 embraces an **architecture of participation**—a design that encourages user interaction and community contributions.⁶ You, the user, are the most important aspect of Web 2.0—so important, in fact, that in 2006, *TIME Magazine*'s “Person of the Year” was “you.”⁷ The article recognized the social phenomenon of Web 2.0—the shift away from a powerful few to an empowered many.

“We can't be device centric...we must be user centric.”

—Bill Gates, MIX06 conference⁸

Many Web 2.0 companies are built almost entirely on user-generated content and **harnessing collective intelligence**. The significance is not just in having user-generated content, but in how it is used. Google—the leading search engine and Internet advertising company—sends its users to user-generated websites by considering what users collectively have valued in the past. For websites like MySpace®, Flickr™, YouTube and Wikipedia®, users create the content, while the sites provide the platforms. These companies *trust their users*—without such trust, users cannot make significant contributions to the sites.

“A platform beats an application every time.”

—Tim O'Reilly⁹

The architecture of participation is seen in software development as well. Open source software is available for anyone to use and modify with few or no restrictions—this has played a major role in Web 2.0 development. Harnessing collective intelligence,¹⁰ communities collaborate to develop software that many people believe is better than proprietary software.

You, the user, are not only contributing content and developing open source software, but you are also directing how media is delivered, and deciding which news and information outlets you trust. Many popular blogs now compete with traditional media powerhouses. Social bookmarking sites such as del.icio.us and Ma.gnolia allow users to recommend their favorite sites to others. Social media sites such as Digg™ or Reddit enable the community to decide which news articles are the most significant. You are also changing the way we find the information on these sites by **tagging** (i.e., labeling) web content by subject or keyword in a way that helps anyone locate information more effectively. This is just one of the ways Web 2.0 helps users identify new meaning in already existing content. RSS feeds (Chapter 14, XML and RSS) enable you to receive new information as it is updated—pushing the content right to your desktop.

-
6. O'Reilly, T. “What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.” September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
 7. Grossman, L. “TIME's Person of the Year: You.” *TIME*, December 2006 <<http://www.time.com/time/magazine/article/0,9171,1569514,00.html>>.
 8. “Bill Gates: Microsoft MIX06 Conference.” *Microsoft*, March 2006 <<http://www.microsoft.com/presspass/exec/billg/speeches/2006/03-20MIX.mspx>>.
 9. O'Reilly, T. “What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.” September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
 10. O'Reilly, T. “What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.” September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.

The rise of **social networks** has changed the way we interact and network. MySpace—the largest social network—has rapidly become the world’s most popular website. Other popular social networking sites include Facebook, Bebo, LinkedIn, and Second Life—a 3D virtual world where you interact with others via your online persona called an **avatar**.

Many Web 2.0 businesses leverage the **Long Tail**.¹¹ Coined by Chris Anderson in an article in the October 2004 *WIRED* magazine, the Long Tail refers to the economic model in which the market for non-hits (typically large numbers of low-volume items) could be significant and sometimes even greater than the market for big hits (typically small numbers of high-volume items).¹² So an online company like Netflix—which has a catalog of over 80,000 movie titles for rent—typically rents a large volume of less popular movies in addition to the substantial business it does renting hits. A local movie store has limited shelf space and serves a small, local population; it cannot afford the space to carry the Long Tail movies in every store. However, Netflix serves millions of people and does not have the physical constraints of stores; it can keep a small inventory of many Long Tail movies to serve its entire customer base. The opportunity to leverage the Long Tail is made possible by the relative ease of running a Web 2.0 Internet business and is fueled by the social effects of Web 2.0 that increase exposure for lesser-known products.

In this chapter, we introduce some of the key technologies used to create Web 2.0 applications. Many of these technologies are discussed in detail in the programming chapters of *Internet & World Wide Web How to Program*, 4/e. You’ll learn web development technologies, such as Ajax (Chapter 15); its component technologies, including XHTML (Chapter 4), Cascading Style Sheets (CSS, Chapter 5), JavaScript (Chapters 6–11), the Document Object Model (DOM, Chapter 12), XML (Chapter 14) and the XMLHttpRequest object (Chapter 15); and the popular Ajax toolkits—Dojo (Chapter 15) and Script.aculo.us (Chapter 24).

You’ll learn how to build Rich Internet Applications (RIAs)—web applications that offer the responsiveness and rich GUI features of desktop applications. We discuss key tools for building RIAs, including Adobe’s Flex (Chapter 18), Microsoft’s Silverlight (Chapter 19), ASP.NET Ajax (Chapter 25) and Sun’s JavaServer Faces (Chapters 26–27). We present web development tools such as Adobe’s Dreamweaver and its Ajax-enabling capabilities (Chapter 20). We also discuss other popular development technologies including JSON (Chapter 15), the web servers IIS and Apache (Chapter 21), MySQL (Chapter 22), PHP (Chapter 23), and ASP.NET (Chapter 25).

We discuss the emergence of web services (Chapter 28), which allow you to incorporate functionality from existing applications into your own applications quickly and easily. For example, using Amazon Web Services™, you can create a specialty bookstore and earn revenues through the Amazon Associates program; or using Google™ Maps web services with eBay web services, you can build location-based mashup applications to find auction items in certain geographical areas. Web services, inexpensive computers, abundant high-speed Internet access, open source software and many other elements have inspired new, exciting **lightweight business models** that people can launch with only a small investment. Some websites with robust functionality that might have required hundreds of thousands

11. Anderson, C. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

12. Anderson, C. “The Long Tail.” *WIRED*, October 2004 <<http://www.wired.com/wired/archive/12.10/tail.html>>.

or even millions of dollars to build in the 1990s can now be built for nominal amounts of money.

Section 3.17 overviews key Web 2.0 business models, many of which are also explained in greater depth throughout the chapter. Fig. 3.1 includes a list of Web 2.0-related conferences. Some have a technology focus, while others have a business focus.

Web 2.0 and related conferences	
AdTech	Microsoft MIX
Affiliate Marketing Summit	Microsoft Tech Ed
AjaxWorld Expo	MySQL Conference and Expo
All Things Digital	Open Source (OSCON)
Always On	RailsConf
Blog Business Summit	Search Engine Strategies
eBay Live	Tools of Change for Publishing
Emerging Technology	Ubuntu Live
Emerging Telephony	Web 2.0 Expo
Future of Online Advertising	Web 2.0 Summit
JavaOne	Where 2.0

Fig. 3.1 | Web 2.0 and related conferences.

3.3 Search

“Google’s mission is to organize the world’s information and make it universally accessible and useful.”

—Google¹³

In Web 2.0, the saying “content is king” remains a prevailing theme. With seemingly endless content available online, the **findability** of content becomes key. **Search engines** are the primary tools people use to find information on the web. Today, you perform searches with keywords, but the future of web search will use natural language (see, for example, Powerset.com). Currently, when you enter a keyword or phrase, the search engine finds matching web pages and shows you a **search engine results page (SERP)** with recommended web pages listed and sorted by relevance. People-assisted search engines have also emerged, such as Mahalo, which pays people to develop search results.¹⁴ The popularity of vertical search engines—ones that focus on a specific topic or industry—is on the rise, though traffic to these search engines is still far behind the major (more generalized) search engines.

Traffic to the major search engines is growing rapidly—according to a recent comScore (a web analytics company) report, Americans conducted 8 billion search queries in June 2007, up 26% from the previous year. In the same report, the comScore analysis of

13. “Company Overview.” *Google* <<http://www.google.com/intl/en/corporate/index.html>>.

14. “Mahalo Greenhouse FAQ.” *Mahalo* <http://greenhouse.mahalo.com/Mahalo_Greenhouse_FAQ>.

U.S. market share across the most popular search engines reported Google at the top with 49.5% of the U.S. search market, followed by Yahoo! with 25.1%, Microsoft with 13.2%, Ask with 5.0% and Time Warner Network with 4.2%.¹⁵

John Battelle's book *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture* provides an extensive history of search engines and presents strong arguments for the importance of search in almost every aspect of our personal and business lives. John Battelle's *Searchblog* discusses search and technology issues (<http://battellemedia.com>).

Attention Economy

“Telecommunications bandwidth is not a problem, but human bandwidth is.”

—Thomas Davenport and John Beck, *The Attention Economy*¹⁶

The abundant amounts of information being produced and people's limited free time has led to an **attention economy**. More content is available than users can sort through on their own, especially given the demands on their time, such as responsibilities to children, parents, friends, employers, etc. *The Attention Economy*, by Thomas Davenport and John Beck, begins with the familiar story of a man whose attention is constantly demanded by work and family. The authors explain that the constant flow of information in today's world causes attention to continually be diverted.

Though it used to be difficult to obtain diverse content, there are now seemingly endless options competing for an audience's attention. As a result, search engines have gained popularity by helping users quickly find and filter the information they want.¹⁷

Google Search

Google is the leading search and online advertising company, founded by Larry Page and Sergey Brin while they were Ph.D. students at Stanford University. Google is so popular that its name has been added to the Oxford English Dictionary—the verb “Google” means to find something on the Internet using the Google search engine. (“google” with a lowercase “g” is a cricket term, whereas “googol” or 10^{100} is the mathematical term Google was named after.)¹⁸

Google's success in search is largely based on its **PageRank™** algorithm (patented by Stanford University and Larry Page) and its unique infrastructure of servers that uses linked PCs to achieve faster responses and increased scalability at lower costs.¹⁹ Estimates on the number of Google servers range from hundreds of thousands to over one million.²⁰ The PageRank algorithm considers the number of links into a web page and the quality of

-
- 15. “comScore Releases June U.S. Search Engine Rankings.” *CNNMoney*, 16 July 2007 <<http://money.cnn.com/news/newsfeeds/articles/prnewswire/AQM17916072007-1.htm>>.
 - 16. Davenport, T. and J. Beck. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business School Press, 2002, p.2.
 - 17. Thompson, C. “Media in the Age of the Swarm.” *cbc.ca* <http://www.cbc.ca/10th/columns/media_thompson.html>.
 - 18. Brin, S. and L. Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine.” <<http://infoLab.stanford.edu/~backrub/google.html>>.
 - 19. “Technology Overview.” *Google* <<http://www.google.com/corporate/tech.html>>.
 - 20. “Google: One Million and Counting.” *Pandia Search Engine News*, 2 July 2007 <<http://www.pandia.com/sew/481-gartner.html>>.

the linking sites (among other factors) to determine the importance of the page. Each inbound link is a vote saying that site is valuable to someone else; however, votes are given different weights depending on the “voter” site’s own value. So, two pages could have the same PageRank even if one has numerous links in from other pages and the other has fewer links in but from pages with higher PageRank. Google search also considers all of the content on the page, its fonts, its headers and the content of neighboring pages.²¹ Sites with the highest PageRank will appear at the top of the search results.

In addition to its regular search engine, Google offers specialty search engines for images, news, videos, blogs and more. Using Google web services, you can build Google Maps and other Google services into your applications (see Section 3.13, Web Services, Mashups, Widgets and Gadgets).

AdWords, Google’s pay-per-click (PPC) contextual advertising program (launched in 2000), is the company’s main source of revenue. AdWords ads appear next to search results on the Google site (and are related to the search query). Advertisers write their own ads, which are unobtrusive and uniform in appearance—each ad consists of a headline, limited text and a URL. Advertisers bid on search keywords related to their ads and pay based on the number of users who click on the ads.

AdSense is Google’s advertising program for publishers (sites like <http://www.deitel.com> that offer content), inspired by Susan Wojcicki, the vice president of product management. (In 1998, Wojcicki rented a spare room in her house to Larry Page and Sergey Brin where they founded Google.)²² AdSense is a fundamental and popular form of website monetization, particularly for Web 2.0 startup companies. Google text ads (as well as banner and rich-media ads) are placed on participating sites with related content. Click-through rates on contextual ads are often higher than on non-contextual ads because the ads reach people expressing interest in a related topic. As a result, contextual pay-per-click ads generally pay a higher eCPM (effective cost per thousand impressions).

Yahoo!

Yahoo! was started in 1994 by Jerry Yang and David Filo (also Stanford Ph.D. students) as a web directory rather than a search engine. The original site, “Jerry and David’s Guide to the World Wide Web,” consisted of their favorite websites manually added to a categorized directory.²³ As the web grew, maintaining the directory structure became increasingly difficult, and a search capability was created for better access to the data. Focusing more on search, Yahoo! also expanded into other areas, becoming a popular provider of e-mail, user groups and more. In 2003, Yahoo! acquired Overture (now Yahoo! Search Marketing), which was the first search engine to offer sponsored search results successfully.²⁴

MSN

MSN search was created in 1998, a year after Google was launched.²⁵ Over the past few years, Microsoft has made search engine technology development a top priority.²⁶ Mi-

-
- 21. “Technology Overview.” *Google* <<http://www.google.com/corporate/tech.html>>.
 - 22. Graham, J. “The House that Helped Build Google.” *USA TODAY*, 5 July 2007, 1B.
 - 23. “Company History.” *Yahoo!* <<http://yahooclient.shareholder.com/press/history.cfm>>.
 - 24. Mills, E. “Google Rises at Yahoo’s Expense.” *CNET*, 23 April 2007 <http://news.com.com/Google+rises+at+Yahoos+expense/2100-1038_3-6178164.html>.
 - 25. Underwood, L. “A Brief History of Search Engines.” *Web Ref* <http://www.webreference.com/authoring/search_history/>.

crosoft search query volume and its search market share grew rapidly in June 2007; analysis companies comScore and Compete attribute this boost largely to MSN's Live Search club, a program introduced in May 2007 to reward users of Live Search.^{27, 28} MSN's Live Search includes a new search engine, index and crawler.²⁹ It allows you to search the web, performing specialized searches (news, images, or local listings) or MSN content searches.³⁰ Another approach that Microsoft is taking to increase its search market share is buying vertical search sites such as MedStory, a health search engine.³¹ Microsoft is also looking to gain market share in the contextual advertising market through Microsoft ad-Center (similar to Google AdWords and Yahoo! Search Marketing).

Ask

Ask (formally known as AskJeeves.com) is owned by InterActiveCorp (IAC), which also owns Ticketmaster®, Match.com®, LendingTree.com®, RealEstate.com® and many other Internet properties. In June 2007, Ask launched a new search site, which includes a new design with a simple homepage default, customizable backgrounds, new video search (powered by Blinkx) and the ability to view video previews and listen to music clips. The search results are based on the searcher's location—Ask will report relevant local businesses and events. Searching for movies, for example, will show local show times.

Vertical Search

Vertical search engines are specialists (focusing on specific topics) in comparison to generalists (e.g., Google and Yahoo!).³² Vertical search engines enable you to search for resources in a specific area, with the goal of providing you with a smaller number of more relevant results. Popular vertical search engines include travel sites (such as Kayak or Expedia), real-estate sites (such as Zillow or Trulia), job search sites (such as Indeed or Monster) and shopping search engines (such as Shopzilla and MySimon).

Location-Based Search

Location-based search (offered by most major search engines as well as some smaller specialized ones) uses geographic information about the searcher to provide more relevant search results. For example, search engines can ask the user for a ZIP code or estimate the user's general location based on IP address. The engine can then use this information to give higher priority to search results physically located near the user. This is particularly useful when searching for businesses such as restaurants or car services. (See Section 3.14 for more information on location-based services.)

-
26. Olson, S. "MSN Launches Revamped Search Engine." *CNET*, 30 June 2004 <http://news.com.com/MSN+launches+revamped+search+engine/2100-1032_3-5254083.html>.
 27. "comScore Releases June U.S. Search Engine Rankings." *CNNMoney*, 16 July 2007 <<http://money.cnn.com/news/newsfeeds/articles/prnewswire/AQM17916072007-1.htm>>.
 28. Sullivan, D. "Compete: Microsoft Gaining Searches; Live Search Club Giveaway Working?" *Search Engine Land*, 10 July 2007 <<http://searchengineland.com/070710-105603.php>>.
 29. "MSN Live Search: About Us." *MSN* <<http://search.msn.com/docs/default.aspx>>.
 30. "Web Search: How to use MSN Search." *MSN* <http://search.msn.com/docs/help.aspx?t=SEARCH_CONC_WhatsNewWithMSNSearch.htm>.
 31. "Vertical Search-Engines: Know Your Subject." *Economist.com*, 12 July 2007 <http://www.economist.com/business/displaystory.cfm?story_id=9478224>.
 32. "Vertical Search-Engines: Know Your Subject." *Economist.com*, 12 July 2007 <http://www.economist.com/business/displaystory.cfm?story_id=9478224>.

Creating Customized Search Engines

Rollyo—a build-your-own customized search engine website—allows you to explore, create and personalize search engines (“searchrolls”) created by others. This helps you narrow your search to sites you already trust.³³ Other custom search sites include Gigablast and Google Custom Search Engine.

Search Engine Optimization (SEO)

Search Engine Optimization (SEO) is the process of designing and tuning your website to maximize your findability and improve your rankings in organic (non-paid) search engine results. To maximize traffic, you need to take into consideration how search engines work when you design your website. There are two ways of employing SEO. The first, **white hat SEO**, refers to methods that are approved by search engines, do not attempt to deceive the search engines, and produce quality, long-term results. Top white hat techniques for SEO include: offering quality content, using proper metadata and effective keywords, and having inbound links from relevant high-quality pages.³⁴ **Black hat** methods are used to deceive search engines. Although they may result in temporary improvement in search engine results, these tactics could get your site banned by the search engines. A “Googlebomb” (or link bomb) is an example of a black hat method—it attempts to trick the Google algorithm into promoting a certain page (generally for humorous reasons).³⁵

Link Building

Link building is the process of increasing search engine rankings and traffic by generating inbound links to a particular website. Search engine algorithms regard each link as a vote for the destination website’s content, so sites with the greatest **link popularity** (or number of high-quality inbound links) appear highest on search engine result pages (SERPs). The three most practiced methods of building links include reciprocal linking, link baiting and natural linking. **Reciprocal linking** is an exchange in which two related websites link to each other, increasing the link popularity of both sites and adding value for site users. **Link baiting** involves creating attention-grabbing web content specifically for viral (exponentially increasing) exposure through social media and social bookmarking websites. **Natural linking** is the process of building one-way inbound links by optimizing website content and user experience without the explicit solicitation of a backlink. Search algorithms are continuously updated to prevent black hat SEOs from deceiving search engines with automated linking software and links from directories or other low-quality websites. One-way links from websites with strong, related pages are given greater weight than reciprocal links, links from sites with unrelated content or links from sites with low PageRank.

Search Engine Marketing (SEM)

Search Engine Marketing (SEM) is the method of promoting your website to increase traffic and search results by raising the site’s visibility on search engine results pages. Danny Sullivan (founder of Search Engine Watch and, more recently, Search Engine Land) introduced the term “Search Engine Marketing” in 2001 to include SEO, manag-

33. “About Rollyo.” *Rollyo* <<http://www.rollyo.com/about.html>>.

34. Wilding, R. “Top 5 Black Hat and White Hat Search Engines Optimisation Techniques.” *PushON* <<http://www.pushon.co.uk/articles/top5-black-hat-white-hat-seo.htm>>.

35. Calore, M. “Remembering the First Google Bomb.” *Compiler (WIRED blog)*, 26 January 2007 <http://blog.wired.com/monkeybites/2007/01/earlier_today_m.html>.

ing paid listings, developing online marketing strategies and submitting sites to directories.³⁶ SEO is the most popular form of search engine marketing, which continues to take away business from other marketing channels (especially offline sources). According to the Search Engine Marketing Professional Organization's annual State of Search Engine Marketing survey, North American advertisers spent \$9.4 billion on search engine marketing in 2006, a 62% increase over 2005 spending.³⁷

Search Engine Watch and Search Engine Land

Search Engine Watch is a search engine marketing resource site. It includes articles, tutorials, conferences and more. The site, launched in 1997 by Danny Sullivan, was inspired by his 1996 release of "A Webmaster's Guide To Search Engines." Search Engine Watch incorporates Web 2.0 features (blogging and forums in addition to expert columnist articles). Other Search Engine Watch departments include search engine submission tips, web searching tips, popular search engines and search engine resources (numerous topics related to search engines). Danny Sullivan served as Search Engine Watch's editor-in-chief until November 2006, when he left the site and became the editor-in-chief for **Search Engine Land**. The site provides news and information on the major search engines—Google, Yahoo!, and Microsoft—as well as search engine marketing and searching issues. The site also informs users of upcoming related conferences and webcasts.

Search Engine Strategies Conferences

Search Engine Strategies is a global conference series focused on search engine advertising (including current SEO and SEM issues). Search Engine Strategies (hosted by Search Engine Watch) offers event information given by the top experts in the field as well as representatives from search engine companies.³⁸ Because traffic and advertising are so important to most Web 2.0 businesses, understanding the search process and making sure your site is easily found is vital.

Discovery

Rather than the traditional use of search engines (searching with a topic in mind), **discovery** refers to finding new content you would not have otherwise sought out. For example, Yahoo!'s original directory design allowed users to browse categories, and discover new interesting sites. StumbleUpon, a social bookmarking site, addresses discovery with its recommendation system that helps you discover and share websites based on your interests. Content networks also direct users to web content they would not necessarily have looked for otherwise.

3.4 Content Networks

Content networks are websites or collections of websites that provide information in various forms (such as articles, wikis, blogs, etc.). These provide another way of filtering the vast amounts of information on the Internet, by allowing users to go to a trusted site that

-
- 36. Sullivan, D. "Congratulations! You're a Search Engine Marketer!" *Search Engine Watch*, 5 November 2005 <<http://searchenginewatch.com/showPage.html?page=2164351>>.
 - 37. Sherman, C. "The State of Search Engine Marketing 2006." *Search Engine Land*, 8 February 2007 <<http://searchengineland.com/070208-095009.php>>.
 - 38. *Search Engine Strategies: Conference & Expos*, 2007 <<http://www.searchenginestrategies.com>>.

has already sorted through many sources to find the best content or has provided its own content. Figure 3.2 shows some examples of content networks.

Content networks	
About.com —Acquired by the <i>New York Times</i> , About is a collection of information on a wide variety of topics. About was founded in 1996 and provides over 500 guides written by topic experts. The guides include new content as well as links to other websites.	Gawker Media —A blog network that includes 14 blogs, such as Gizmodo, Gawker, Valleywag and Lifehacker. The blogs cover a range of topics including technology, gossip and more.
b5media —A blog network with over 200 blogs related to travel, entertainment, technology and more.	HowStuffWorks —HowStuffWorks offers articles explaining “how the world actually works.” Articles are written by freelance writers, and experts from <i>Consumer Guide</i> and <i>Mobil Travel Guide</i> .
Corante —A blog network authored by leading commentators in technology, business, law, science, and culture.	LifeTips —LifeTips provides short articles on both work and general life issues from hundreds of writers. Tips are voted on by readers (who can also mark their favorites for easy access).
Deitel —Deitel Resource Centers (currently about 80 sites and growing rapidly) include links to, and descriptions of, key tutorials, demos, free software tools, articles, e-books, whitepapers, videos, podcasts, blogs, RSS feeds and more. Resource Centers are grouped into major topic areas, including Web 2.0, Internet business, programming languages, software development and open source. See Fig. 2 in the Preface for a complete list of Resource Centers.	9rules —A blog network with a wide range of blog topics. The site also includes social networking aspects.
eHow —eHow claims over 35,000 articles explaining “how to do just about everything.” The articles are written by members, and the site also features a section of “how to” videos.	Suite101 —Suite101 offers thousands of articles on a variety of topics written by freelance writers. In addition to the articles, the site also provides discussion areas and free courses.
	Weblogs, Inc. —A blog network of 90 blogs, including Engadget, Autoblog and Joystiq. Users can apply to write for one of the blogs (and get paid) or suggest topics for potential new blogs.

Fig. 3.2 | Content networks.

3.5 User-Generated Content

User-generated content has been the key to success for many of today’s leading Web 2.0 companies, such as Amazon, eBay and Monster. The community adds value to these sites, which, in many cases, are almost entirely built on user-generated content. For example, **eBay** (an online auction site) relies on the community to buy and sell auction items, and **Monster** (a job search engine) connects job seekers with employers and recruiters.

User-generated content includes explicitly generated content such as articles, home videos and photos. It can also include implicitly generated content—information that is gathered from the users’ actions online. For example, every product you buy from Amazon and every video you watch on YouTube provides these sites with valuable information

about your interests. Companies like Amazon have developed massive databases of anonymous user data to understand how users interact with their site. For example, Amazon uses your purchase history and compares it to purchases made by other users with similar interests to make personalized recommendations (e.g., “customers who bought this item also bought...”). Implicitly generated content is often considered hidden content. For example, web links and tags are hidden content; every site you link to from your own site or bookmark on a social bookmarking site could be considered a vote for that site’s importance. Search engines such as Google (which uses the PageRank algorithm) use the number and quality of these links to a site to determine the importance of a site in search results.

Collective Intelligence

Collective intelligence is the concept that collaboration can result in smart ideas. Working together, users combine their knowledge for everyone’s benefit.

The first chapter of *Wikinomics*, by Don Tapscott and Anthony D. Williams, tells the Goldcorp story. Inspired by the community efforts in Linux, the CEO of Goldcorp released to the public proprietary geological information about the company’s land. Goldcorp offered cash rewards to people who could use this information to help the company locate gold on the land. The community helped his company find 8 million ounces of gold, catapulting Goldcorp from \$100 million in stock equity to \$9 billion.³⁹ Goldcorp reaped amazing benefits by sharing information and encouraging community participation.

User-generated content is significant to Web 2.0 companies because of the innovative ways companies are harnessing collective intelligence. We’ve already discussed Google’s PageRank (Section 3.3), which is a product of collective intelligence. Amazon’s and Last.fm’s personalized recommendations also result from collective intelligence, as algorithms evaluate user preferences to provide you with a better experience by helping you discover new products or music preferred by other people with similar interests. Wesabe is a web community where members share their decisions about money and savings—the site uses the collective financial experiences of the community to create recommendations.⁴⁰ Reputation systems (used by companies like eBay) also use collective intelligence to build trust between buyers and sellers by sharing user feedback with the community. Social bookmarking sites (Section 3.10), and social media sites (like Digg and Flickr) use collective intelligence to promote popular material, making it easier for others to find.

Wikis

Wikis, websites that allow users to edit existing content and add new information, are prime examples of user-generated content and collective intelligence. The most popular wiki is **Wikipedia**, a community-generated encyclopedia with articles available in over 200 languages. Wikipedia trusts its users to follow certain rules, such as not deleting accurate information and not adding biased information, while allowing community members to enforce the rules. The result has been a wealth of information growing much faster than could otherwise be produced. In 2005, an experiment comparing 42 entries from Wikipedia and *Britannica* (a popular printed traditional encyclopedia) showed only slightly

39. Tapscott, D. and A.D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover, 2006.

40. “FAQ.” *Wesabe* <<http://www.wesabe.com/page/faq>>.

more inaccuracies in the Wikipedia articles.⁴¹ The Wikipedia entries were promptly corrected, though, whereas errors in *Britannica* entries cannot be corrected until the book's next printing and will remain in already printed copies.

Wikipedia, [Wikia](#) (a site for specialized wiki communities about popular television shows, games, literature, shopping and more) and many other wikis use [MediaWiki](#) open source software (originally developed for Wikipedia). The software can be downloaded from MediaWiki's website (www.mediawiki.org), where you can also find descriptions, tutorials, suggestions and more to help navigate the software. Wikis are also used by many companies to provide product information, support and community resources. [SocialText](#), the first wiki company, provides corporate wiki services. Many companies have found that using wikis for project collaboration reduces e-mails and phone calls between employees, while allowing the ability to closely track a project's changes.⁴²

Collaborative Filtering

Though collaboration can result in a wealth of knowledge, some users might submit false or faulty information. For example, Wikipedia has experienced instances of people deliberately adding false information to entries. While **moderation** (monitoring of content by staff) is sometimes necessary, it is time consuming and costly. Many Web 2.0 companies rely on the community to help police their sites. This **collaborative filtering** lets users promote valuable material and flag offensive or inappropriate material. Users have the power to choose for themselves what is important. Examples of sites using collaborative filtering include Digg, a news site where users rate the stories (see Section 3.8), and social bookmarking sites such as del.icio.us, where users can easily find popular sites (see Section 3.10). Customer reviews on Amazon products also employ collaborative filtering—readers vote on the usefulness of each review (helping other readers to find the best reviews).

Craigslist

[Craigslist](#), founded by Craig Newmark, is a popular classified ads website that has radically changed the classified advertising market. Newspapers have experienced a decline in classified ad sales,⁴³ as revenues from help-wanted ads on Craigslist climbed to \$50 million in 2006.⁴⁴ Most ad postings on Craigslist are free, and it's easy for anyone to post ads. The site has gained popularity because of its job and housing postings. In 2005, a documentary, "24 Hours on Craigslist," showed the diverse postings that occur on the site in a single day.⁴⁵ Craigslist is built on user content, leveraging the Long Tail by connecting the unique (often unusual) needs of its users. The site also uses collaborative filtering—users are encouraged to flag inappropriate postings.

-
41. Cauchi, S. "Online Encyclopedias Put to the Test." *The Age*, 15 December 2005 <<http://www.theage.com.au/news/national/online-encyclopedias-put-to-the-test/2005/12/14/1134500913345.html>>.
 42. "SocialText is the Enterprise Wiki Trusted Most by Global 2000 Corporations." *SocialText* <<http://www.socialtext.com/products/overview>>.
 43. Steel, E. "Newspapers' Ad Sales Show Accelerating Drop." *The Wall Street Journal*, 18 July 2007, A4.
 44. "Leading Indicators." *FORTUNE*, 13 November 2006, p.40.
 45. "24 Hours on Craigslist." <<http://24hoursoncraigslist.com/>>.

Wisdom of Crowds

Wisdom of crowds (from the book of the same title written by James Surowiecki) is similar to collective intelligence—it suggests that a large diverse group of people (that does not necessarily include experts) can be smarter than a small group of specialists. The key difference between collective intelligence and the wisdom of crowds is that the latter is not meant to be a collaborative process—part of forming a reliable crowd is making sure people don't influence each other.⁴⁶ For example, Surowiecki describes how calculating the average of all submissions in a guessing contest (e.g., guessing the number of jelly beans in a jar) often results in nearly the correct answer, even though most individual estimates are incorrect and vary considerably. When the U.S. submarine *Scorpion* sank in 1968, the Navy asked various experts to work individually assessing what might have happened; their collective answers were then analyzed to determine the accurate location of the submarine.⁴⁷ Practical everyday applications of the wisdom of crowds can be seen in sites employing collaborative filtering.

3.6 Blogging

“The blog is the best relationship generator you’ve ever seen.”

—Robert Scoble, blogger⁴⁸

History of Blogging

Blogs are websites consisting of entries listed in reverse chronological order. They have existed since the mid-1990s; however, interest in blogging has grown exponentially in recent years because of easy-to-use blogging software and increasingly economical Internet access. The term “blog” evolved from **weblog**, a regularly updated list of interesting websites. These blogs consisted of short postings, in reverse chronological order, that contained links to other web pages and short commentaries or reactions. Blogging has since taken on a looser structure—some blogs still follow the traditional format of links and small amounts of text, while others consist of essays, sometimes not containing any links. Blogs can also now incorporate media, such as music or videos. Many people are familiar with personal journal blogs, like those on **Xanga** or **LiveJournal**. These sites include social networking features and are particularly popular with teenage bloggers, who often write about their day-to-day lives for friends.

Blogging has become a major social phenomenon, empowering users to participate in, rather than just view, the web. In July 2006 most **bloggers**, or blog authors, had not had a personal website before starting their blog.⁴⁹ The increased availability of user-friendly blogging software has allowed blogging to become accessible to more mainstream Internet users.

46. Jenkins, H. “Collective Intelligence vs. The Wisdom of Crowds.” *Confessions of an Aca-Fan*, 27 November 2006 <http://www.henryjenkins.org/2006/11/collective_intelligence_vs_the.html>.

47. Surowiecki, J. *The Wisdom of Crowds*. Anchor, 2005.

48. Kirkpatrick, D. “Why There’s No Escaping the Blog.” *FORTUNE*, 10 January 2005 <http://money.cnn.com/magazines/fortune/fortune_archive/2005/01/10/8230982/index.htm>.

49. Lenhart, A. and S. Fox. “Bloggers: A Portrait of the Internet’s New Storytellers.” *Pew Internet & American Life Project*, July 2006 <<http://www.pewinternet.org/pdfs/PIP%20Bloggers%20Report%20July%2019%202006.pdf>>.

Blog Components

Reader comments create an interactive experience, allowing readers to react to blog entries. According to a Pew Internet study, 87% of blogs allow reader comments.⁵⁰ Successful bloggers pay attention to their readers and respond, often sparking interesting discussions and debates. However, allowing comments increases the possibility of spam (including irrelevant comments, inappropriate language and link spam—where a user tries to increase an irrelevant site’s number of inbound links). By some estimates, over 90% of blog comments are spam.⁵¹

Permalinks provide blog readers with a way of linking to specific blog entries. Each blog post has a unique URL referring to that single post. Links stay relevant even after the blog entry moves off the homepage and into the archive.

Trackbacks tell bloggers who is linking to their posts. This enhances Internet content by making linking two-way. The blogger provides a trackback link, and sites that use the link are added to a list on the blog entry. For an example of a trackbacks section, visit <http://www.techcrunch.com/2006/08/08/web-20-the-24-minute-documentary/>. This is a permalink to a post on **TechCrunch**, a popular Internet technology blog, that features a Web 2.0 video from 2006.

A **blogroll** is a list of the blogger’s favorite blogs. Though not all blogs feature a blog-roll, it is common for the main page of a blog to contain links to several other blogs. For example, LiveJournal automatically incorporates a blogroll (consisting of users the blogger has marked as friends) into a user’s profile page.

Blogging and Journalism

“Freedom of the press is guaranteed only to those who own one.”

—A.J. Liebling⁵²

Blogging has encouraged **citizen journalism**, allowing anyone to be a journalist. Blogs have become a significant news resource, drawing traffic away from the mainstream media. Some argue that this form of “participatory journalism” holds less biases than mainstream media, or at least makes these biases clear and provides many different views. This **democratization of media** allows a larger group to take part in journalism.⁵³ Traditional journalists had previously been able to create a representative democracy (much like the political system of the United States) by speaking for the masses. However, blogging gives a voice to everyone with a computer and Internet access, creating a more direct democracy.

Many bloggers are recognized as members of the media. Just as television and radio increased the speed of news delivery over that of newspapers, blogs have become a fast and in-depth (and often “unwashed”) news medium. The mass media is embracing blogging; many TV news anchors suggest that viewers read their blogs after the show, and many newspaper websites feature blogs by reporters.

-
- 50. Lenhart, A. and S. Fox. “Bloggers: A Portrait of the Internet’s New Storytellers.” *Pew Internet & American Life Project*, July 2006 <<http://www.pewinternet.org/pdfs/PIP%20Bloggers%20Report%20July%202006.pdf>>.
 - 51. Akismet, 10 August 2007 <<http://akismet.com/stats/>>.
 - 52. “A.J. Liebling Quotes.” *ThinkExist.com Quotations* <http://thinkexist.com/quotation/free_dom_of_the_press_is_guaranteed_only_to_those/220714.html>.
 - 53. Bowman, S. and C. Willis. “We Media.” July 2003 <http://www.hypergene.net/wemedia/download/we_media.pdf>.

Though journalism is a large part of the blogging phenomenon, according to a Pew Internet study only one-third of bloggers consider their blogs a form of journalism. Eighty-four percent of bloggers consider it a hobby, and only 10% spend more than ten hours a week blogging.⁵⁴ Posting new content and responding to reader comments requires a substantial time commitment.

Growth of Blogging

The number of blogs has been doubling about twice a year.⁵⁵ However, there is also a large number of abandoned blogs. A Caslon Analytics study found that “66.0% of surveyed blogs had not been updated in two months.”⁵⁶

Companies are reaching out to the **blogosphere**, or blogging community, to keep in touch with consumer opinions. Many CEOs and top executives from large companies such as Sun Microsystems, Marriott International and General Motors are now regular bloggers. This helps build consumer trust and loyalty. The NewPR Wiki lists over 250 CEOs and upper-management bloggers.⁵⁷

Increased use of mobile devices has also lead to **moblogging**, or mobile blogging, as bloggers no longer need to be at their computer to update their blogs. Similarly, **vlogging**, or video blogging, has gained popularity. Rocketboom, for example, posts a three-minute video every day covering news and Internet stories.

Blogging and RSS Feeds

Many popular blogs provide RSS and Atom feeds to let readers know when new content is posted. Feeds, offered through blogging software or sites such as **Feedburner** (acquired by Google in 2007), help bloggers track and maintain a steady readership. The feeds (containing an entire post or just a selection with a link) can be automatically syndicated via the web and aggregated on a website or application designated by the user. Some sites (like Feedburner) provide an e-mail option, forwarding the day’s posts to subscribers. While the use of feeds is certainly growing, a Pew Internet study in July 2006 reported that only 18% of bloggers provide RSS feeds.⁵⁸ (See “RSS and Atom” in Section 3.15.)

Blogging Software

Bloggers now have many options for building blogs. Online hosted blog software options include **WordPress** (which also offers server software), **TypePad** and **Blogger**. Blog server software programs include **Movable Type** and **Textpattern**. These require users to have their own web server; however, they also allow for more customization. Some word pro-

- 54. Lenhart, A. and S. Fox. “Bloggers: A Portrait of the Internet’s New Storytellers.” *Pew Internet & American Life Project*, July 2006 <<http://www.pewinternet.org/pdfs/PIP%20Bloggers%20Report%20July%2019%202006.pdf>>.
- 55. Walsh, B. *Clear Blogging: How People Are Changing the World and How You Can Join Them*. Apress, 2007.
- 56. “Blog Statistics and Demographics.” *Caslon Analytics*. March 2007 <<http://www.caslon.com.au/weblogprofile1.htm>>.
- 57. “CEO Blog List.” *NewPR Wiki* <<http://www.thenewpr.com/wiki/pmwiki.php/Resources/CEOBlogsList?pagename=Resources.CEOBlogsList>>.
- 58. Lenhart, A. and S. Fox. “Bloggers: A Portrait of the Internet’s New Storytellers.” *Pew Internet & American Life Project*, July 2006 <<http://www.pewinternet.org/pdfs/PIP%20Bloggers%20Report%20July%2019%202006.pdf>>.

cessors (such as Microsoft Word 2007) also offer blog publishing features or are compatible with blog posting extensions.

Blog Networks

Blog networks are collections of blogs, often with several editors. Popular blog networks include Corante, Weblogs, Inc., 9rules, b5media and Gawker Media. Many of these networks, with multiple bloggers and daily postings, draw significant traffic and a broad audience. Blog networks help bloggers build reputations and loyal readers. Some social networking sites like MySpace and Facebook also enable blogging to a private network of friends.

Blog Search Engines

Blog search engines, such as [Technorati](#) and [Google Blog Search](#), monitor the blogosphere's constant changes. When dealing with blogs, search results cannot be based strictly on traditional factors such as reputations built over time (since the blogosphere is so dynamic). Technorati, which tracked over 93 million blogs in July 2007, addresses the unique needs of what they call the "World Live Web." Google Blog Search adjusts Google's search algorithms to specifically address the blogosphere. Other blog search engines include Feedster, IceRocket and Blogdigger.

3.7 Social Networking

Social networking sites, which allow users to keep track of their existing interpersonal relationships and form new ones, are experiencing extraordinary growth in Web 2.0. According to the "Hitwise US Consumer Generated Media Report," in September 2006 "one in every 20 Internet visits went to one of the top 20 social networks." A large portion of the traffic on shopping sites (and other Web 2.0 sites) comes from social networking websites such as MySpace.⁵⁹

Network Effects

"What distinguished 2.0 is the design of systems that harness network effects—a broader way of saying community—to get better the more people use them."

—Tim O'Reilly⁶⁰

The term **network effects** refers to the increased value of a network as its number of users grows. **Metcalfe's Law** states that the value of the network is proportional to the square of the number of users.⁶¹ Consider, for example, eBay—the more buyers and sellers that use the site, the more valuable the site becomes to its users. Google's AdSense advertising program also increases in value as the number of participating advertisers and publishers grows and ads can be better matched to site content (see Section 3.3). Social networking sites also rely heavily on network effects, often attracting users only if their friends are on the site.

A key part of building a successful network and creating an architecture of participation is setting the user preferences to **default to share** content so users will automatically

-
59. Prescott, L. "Hitwise US Consumer Generated Media Report." *Hitwise*, February 2007.
 60. Heiss, J. "Open Possibilities at the First CommunityOne Conference." *JavaOne*, 7 May 2007 <http://java.sun.com/javaone/sf/2007/articles/comm1_post.jsp>.
 61. "Metcalfe's Law." <<http://www-ec.njit.edu/~robertso/infosci/metcalf.html>>.

contribute to the value of the network.⁶² Most users do not think about sharing capabilities, let alone care to alter their preferences. If companies do not enable sharing automatically, few users will take the time to share their data. *Providing the option to disable sharing is an important privacy feature.*

Network effects also make it difficult (though not impossible) to break into markets already claimed by successful companies. User content often loses value when moved into a new network. For example, a photo's tags (created by the community) on Flickr are lost if the photo is taken to a different site. Competitors must then find a unique way of convincing users that it's worth the switch.

Friendster

Friendster was an early leader in social networking. Within a year of Friendster's founding in 2002, Google offered to buy the site (Friendster rejected the offer). Created as a dating site, Friendster experienced a boom in popularity that quickly overwhelmed its servers. Friendster's popularity declined as new sites like MySpace emerged.⁶³ Though Friendster has not been able to keep pace with competing social networking sites, it still claims over 45 million members worldwide. It was granted a patent in 2006 on a key part of social networking, specifically how networks of friends are developed (i.e., identifying mutual friends and degrees of separation).⁶⁴

MySpace

MySpace is the most popular social networking site. Hitwise reported it as the top website in May 2007 based on market share (beating Google by 1.5%).⁶⁵ Self-defined as "an online community that lets you meet your friends' friends," MySpace allows you to build a network of friends and identify mutual friends. Each user's page can contain general info, pictures, blog entries, a message board and more. Customization options, such as changing the background or adding music, give users an easy way to create their own unique web page. The site also features a private messaging system and special sections for film, music, videos, classifieds, etc.

MySpace plays an important role in the music scene, and even companies and politicians are creating accounts. MySpace reaches a younger audience than most conventional media outlets. Some political candidates have used MySpace to reach out to young voters and find new volunteers. Though candidates risk embarrassing connections (to inappropriate accounts) on these sites, they have often found the benefits to be worth it.⁶⁶ Businesses can also create profiles, which then become a form of free advertising. News Corp, which acquired MySpace in 2005 for \$580 million, recognizes its benefits for local busi-

-
- 62. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
 - 63. Rivlin, G. "Wallflower at the Web Party." *New York Times*, 15 October 2006 <<http://www.nytimes.com/2006/10/15/business/yourmoney/15friend.html>>.
 - 64. Kirkpatrick, M. "Friendster Awarded Patent on Social Networking." *TechCrunch*, 7 July 2006 <<http://www.techcrunch.com/2006/07/07/friendster-awarded-patent-on-social-networking/>>.
 - 65. "Top 20 Websites." *Hitwise*, May 2007. <<http://hitwise.com/datacenter/rankings.php>>.
 - 66. Jesdanun, A. "Candidates Seek Youths at MySpace." *ABC News*, 17 August 2006 <<http://abcnews.go.com/Technology/wireStory?id=2325325&page=1>>.

nesses that want to gain exposure.⁶⁷ Though many consider social networking sites to be more popular with teenagers and young adults, the largest user group on MySpace (and other large social networking sites) consists of 35–54 year olds.⁶⁸

Facebook

Hitwise named **Facebook** the “preferred network among college students. Because Facebook was closed to non-students, students felt safer than on MySpace, and Facebook became nearly a social necessity for students seeking to connect with peers.”⁶⁹ In July 2007, Facebook held an 85% market share of four-year U.S. universities and had over 31 million users.⁷⁰ Though Facebook has since allowed users without an .edu e-mail address to join, this elitism and idea of increased privacy drew a large enough crowd to compete with MySpace. A user can set privacy levels for networks or even individuals, but Facebook users (as well as users of other social networking sites) are warned about possible repercussions from information they post.

“Remember, unless you’re prepared to attach something in your profile to a resume or scholarship application, don’t post it.”

—Facebook⁷¹

The site has added many features over the past few years, including photo albums where you can tag your friends in pictures, recently updated profiles lists, events, groups, a marketplace for classified ads, and user status updates. In May 2007, the site introduced third-party applications that can be integrated directly into Facebook. Not all feature implementations have gone smoothly, though. In Fall 2006, Facebook experienced resistance from users concerned over privacy issues when it added a “News Feed” feature, which lists updates of friends’ Facebook activities in real time.⁷² Facebook increased privacy options in response, quieting most complaints.

LinkedIn

In June 2007, **LinkedIn** claimed a membership of “11 million experienced professionals.” The business-oriented social networking site allows users to stay in touch with professional contacts, network with new contacts, check references, and find a job or a potential employee. Its low-key design and feature implementations keep the site unobtrusive.⁷³ Because of its older, more mature audience, privacy concerns are more prevalent—some users worry that their professional contacts will be abused by other users or even their employers

-
67. “Businesses Find MySpace is a Place to Reach Youth.” *Trend Watching*, 11 July 2006 <http://www.trendwatching.com/about/inmedia/articles/youniversal_branding/businesses_find_myspace_is_a_p.html>.
 68. Arrington, M. “Bear Stearns: Yahoo Must Form A Social Networking Strategy.” *TechCrunch*, 3 August 2007 <<http://www.techcrunch.com/2007/08/03/bear-stearns-yahoo-must-form-a-social-networking-strategy>>.
 69. Prescott, L. “Hitwise US Consumer Generated Media Report.” *Hitwise*, February 2007.
 70. “Facebook Statistics.” *Facebook*, 17 July 2007 <http://static.ak.facebook.com/press/facebook_statist_ics.pdf?11:44617>.
 71. “Customer Support.” *Facebook* <<http://www.facebook.com/help.php?tab=safety>>.
 72. Schmidt, T. S. “Inside the Backlash Against Facebook.” *TIME*, 6 September 2006 <<http://www.time.com/time/nation/article/0,8599,1532225,00.html>>.
 73. Copeland, M. “The Missing Link.” *CNNMoney.com*, 28 February 2007 <http://money.cnn.com/magazines/business2/business2_archive/2006/12/01/8394967/index.htm>.

for marketing reasons.⁷⁴ However, the site has gained popularity as a convenient way of networking. Members can find other professionals through their mutual acquaintances and get introductions.

LinkedIn monetizes the site through advertising, premium accounts for power users (mostly recruiters), and groups for companies and organizations. Because of the growing size of its network, LinkedIn maintains a strong hold on the professional market.⁷⁵

Xing

Xing is a professional networking site based out of Germany. Xing is most popular in Europe and offers its services across many countries, industries, and languages—an important factor, given today's globalization of organizations. With its discovery capability and management tools, Xing helps members find professionals, search for job opportunities and locate other business prospects. In April 2007, Xing reached 2 million users.⁷⁶ Xing has also been acquiring other social networks in an attempt to increase its global reach.

Second Life

Second Life, developed by Linden Labs, is a **3D virtual world** with millions of inhabitants. Users create **avatars**, digital representations of themselves that they can use to meet other users with similar interests, conduct business, participate in group activities, take classes and more. Some users have created profitable businesses or continued their real-life professions in the virtual world. For example, lawyers have used Second Life to meet new clients (often software developers wanting to discuss patent laws).⁷⁷ Many large corporations, such as IBM and Hewlett-Packard, have created Second Life presences to connect with customers, hold meetings and even recruit and interview new hires.^{78, 79}

Users can create objects and add scripts (to animate the objects) in the virtual world. Because Second Life allows users to maintain rights to whatever they create, a dynamic marketplace has emerged that does millions of dollars in transactions monthly—the site has its own exchange, the LindeX.⁸⁰ Not only does this create monetization opportunities for users (one woman claims to have earned over \$1 million in Second Life assets⁸¹), but Second Life earns revenue from premium accounts, purchases of virtual land and more.

Gaia Online

Gaia Online is a popular teen virtual world. This online community allows teens to play games, make friends and express their creativity. Similar to Second Life, Gaia has its own marketplace where members can earn Gaia Gold for various actions they perform on the

-
- 74. "The CEO's Guide to Social Networks." *BusinessWeek*, 11 September 2006 <http://businessweek.com/mediacenter/qt/podcasts/guide_to_tech/guidetotech_09_11_06.mp3>.
 - 75. Copeland, M. "The Missing Link." *CNNMoney.com*, 28 February 2007 <http://money.cnn.com/magazines/business2/business2_archive/2006/12/01/8394967/index.htm>.
 - 76. "Xing Reaches 2 Million Users." *Mashable*, <<http://mashable.com/2007/04/16/xing/>>.
 - 77. "Second Life Lawyer." *Business 2.0*, May 2007, p.86.
 - 78. Athavely, A. "A Job Interview You Don't Have to Show Up For." *The Wall Street Journal Online*, 20 June 2007 <<http://online.wsj.com/article/SB118229876637841321.html>>.
 - 79. Bulkeley, W. "Playing Well With Others." *The Wall Street Journal Online*, 18 June 2007 <<http://online.wsj.com/article/SB118194536298737221.html>>.
 - 80. "What is Second Life." <<http://secondlife.com/whatis>>.
 - 81. Lawson, S. "Second Life Creates a Millionaire." *IDG News*, 30 November 2006 <<http://www.itworld.com/App/4201/061130secondlife/>>.

site (e.g., playing games or posting), and use their earnings at the virtual stores or for creating their own shops. Nearly 300,000 members login daily and about 2 million unique visitors login to Gaia every month.⁸²

Mobile Social Networking

Many social networking sites have found innovative ways of connecting people through the Internet and their mobile devices (such as cell phones and PDAs). Mobile users can send instant messages, check e-mail, and post content to the web from Internet-enabled mobile devices. The new Apple iPhone further realizes the dream of having the **Internet in your pocket** by allowing the full Internet (not a simplified mobile one) to be accessed wherever wireless Internet access is available.

Google's [Dodgeball.com](#) provides users with mobile access to a network of friends in many cities. GPS chips in mobile devices allow Dodgeball users to update their location and be notified of nearby friends or "crushes." Dodgeball also provides an easy way of sending messages to groups of friends to plan get-togethers. (See Section 3.14, Location-Based Services.)

Other sites such as [Twitter](#) provide similar services, accessible by text message, IM or a web client. Twitter users can message groups of friends at once and automatically receive their friends' updates on a cell phone or through a chat window. The site is considered to be a [microblogging](#) service (since users are limited to a small number of characters for each update). Twitter offers a web services API, which allows developers to integrate Twitter into other applications. (See Section 3.13, Web Services, Mashups, Widgets and Gadgets, for more information on web services APIs.)

3.8 Social Media

Social media refers to any media shared online (e.g., videos, music, photos, news, etc.). Hitwise reported that "increased broadband penetration, combined with the rise of consumer generated content and the proliferation of webcams and cell phone and home video cameras have firmly entrenched online video viewing into the habits of entertainment seekers in the United States."⁸³

YouTube

[YouTube](#), launched in late 2005, is the leading Internet video site. In true Web 2.0 fashion, the entire site is based on user-generated content. Users upload videos, and rate and comment on videos posted by other users. YouTube's Quick Capture Flash software makes it easy to upload content directly from a webcam. Users can browse videos by category, tag, or by following "related video" links. Highly rated videos are featured on YouTube's homepage. While many professionals and film students post content on the site, the most popular submissions are often simple spoofs or home videos. Because of the viral network effects of YouTube, these amateur videos can quickly gain worldwide attention.

Users can subscribe to other users' content, share videos with friends by e-mail, or embed videos directly into their blogs or other websites. YouTube addresses privacy and

82. "About Us." *Gaia Online*, <<http://www.gaiaonline.com/info/about.php?>>>.

83. Prescott, L. "Hitwise US Consumer Generated Media Report." *Hitwise*, February 2007.

spam concerns by allowing users to set videos as “public” or “private” and flag inappropriate material for review by YouTube’s staff.

Less than a year after its official launch, YouTube was acquired by Google (which had its own less popular Google Video site) for \$1.65 billion. Less than six months after the acquisition, Viacom sued YouTube for \$1 billion for copyright infringement.⁸⁴ The Digital Millennium Copyright Act of 1998 protects companies from prosecution due to user actions if they work in “good faith” to remove offending content.⁸⁵ However, interpretations of this act vary, and it has become a point of contention for many companies. YouTube is developing a mechanism that automatically detects copyrighted material. Currently, illegal content is removed from the site manually.

Internet TV

Many mass-media companies now offer full-length episodes of popular television shows on their websites to tap into the increasingly popular Internet television market. The average American watches 4.5 hours of television a day, not including Internet television.⁸⁶ Sites, such as **Joost**, Veoh and MobiTV, have emerged as a new way of watching television. Joost, for example, uses semantic technologies to help users find programs that interest them. (See Section 3.18, Future of the Web.)

Limited by copyright issues, Internet TV sites must make deals with mainstream networks to offer their content online. Viacom made a deal with Joost, allowing the site to include some shows from networks such as MTV, VH1 and Comedy Central.⁸⁷ As users take back the power to choose what they watch and when, networks may find themselves making more deals with Internet TV companies. As technologies continue to improve, Internet TV has the potential to radically change the television industry. Already, smaller content creators are able to gain access to worldwide audiences. In late June 2007, MySpace joined the market with its MySpaceTV. With MySpace’s enormous membership, it could rapidly become a direct competitor to YouTube and Internet TV websites.

Internet TV allows advertisers to target their markets more precisely than with broadcast television. Advertisers can use demographic information, such as location, gender and age, to serve appropriate ads.

Digg

Digg features news, videos and podcasts, all posted and rated by users. It has gained popularity by allowing users to “digg” or “bury” posts and user comments. Valuable sites, marked by large numbers of diggs, are moved to the Digg front page where other users can easily find them. Formulas were adjusted to make sure the “wisdom of crowds” was not being hijacked by users trying to promote their own posts.⁸⁸ Sites that are “dugg” and fea-

-
- 84. Mills, E. “Copyright Quagmire for Google and YouTube.” *ZDNet*, 14 March 2007 <http://news.zdnet.com/2100-9588_22-6167281.html>.
 - 85. “Conference Report Filed in House.” *Library of Congress*, 8 October 1998 <<http://thomas.loc.gov/cgi-bin/bdquery/z?d105:HR02281:@@@D&sum2=m&>>.
 - 86. Colvin, G. “TV Is Dying? Long Live TV!” *FORTUNE*, 5 February 2007, p.43.
 - 87. O’Hear, S. “Viacom to Partner with Joost.” *ZDNet*, 20 February 2007 <<http://blogs.zdnet.com/social/?p=96>>.
 - 88. Maney, K. “Techies Hot on Concept of ‘Wisdom of Crowds,’ But It Has Some Pitfalls.” *USA Today*, 12 September 2006 <http://www.usatoday.com/tech/columnist/kevinmaney/2006-09-12-wisdom-of-crowds_x.htm>.

tured on the homepage typically experience a traffic surge. Bloggers can add Digg buttons to their sites, making it easy for readers to “digg” their posts.

Digg uses collaborative filtering to help reduce spam by “burying” it (users can vote against posts they don’t like). Users can also set the threshold of diggs to automatically filter out content with low ratings. The site was criticized for removing popular posts of HD DVD security cracks (on the advice of lawyers); however, Kevin Rose (Digg’s founder) decided to support the crowds and “deal with whatever the consequences might be.”⁸⁹ Digg has additional social networking capabilities; users can view their friends’ Digg activities and the Diggs of other users with similar interests. Some Digg-like sites include Netscape, Reddit and Newsvine.

Last.fm

Last.fm is an Internet radio website that uses Web 2.0 concepts to make music recommendations and build communities. The site provides open source desktop software that can be integrated into most popular music players. Its **scrobbling** feature tracks the music users listen to so that Last.fm can provide users with personalized recommendations. A streamable radio with “discovery mode” and a network of like-minded listeners help users find new music. Groups and an events section add social value. The site also offers tagging and wiki pages for artists and record labels.

Digital Rights Management (DRM)

Digital Rights Management (DRM) systems add software to media files to prevent them from being misused, but these systems restrict compatibility with many media players. Companies want to protect their digital products from illegal distribution; however, users want unrestricted access to media they’ve purchased.

iTunes, Apple’s music store, has been criticized for restricting users’ access to their own music by allowing only up to five computers to be authorized to play any given file. However, Apple’s Steve Jobs advocated a DRM-free music world in February 2007, arguing the greater risk for piracy is in DRM-free CDs, which make up the majority of music sales.⁹⁰ CDs remain DRM-free because many CD players are not compatible with DRM systems. In June 2007, Amazon offered DRM-free downloads from more than 12,000 record labels, and both iTunes and Amazon sell DRM-free music from EMI (one of the four major record companies).⁹¹

Podcasting

Podcasting was popularized by Apple’s iPod portable media player. A **podcast** is a digital audio file (e.g., an .mp3) that often takes on the characteristics of a radio talk show (though without live callers).⁹² Much as blogging has made journalism accessible to everyone, pod-

-
89. Hefflinger, M. “Digg Users Revolt Over Deleted Posts of HD DVD Security Hack.” *digitalmediawire*, 2 May 2007 <<http://www.dmwmedia.com/news/2007/05/02/digg-users-revolt-over-deleted-posts-of-hd-dvd-security-hack>>.
 90. Jobs, S. “Thoughts on Music.” 6 February 2007 <<http://www.apple.com/hotnews/thoughtsonmusic/>>.
 91. “Amazon.com to Launch DRM-Free MP3 Music Download Store with Songs and Albums from EMI Music and More Than 12,000 Other Labels.” 16 May 2007 <<http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1003003>>.
 92. Torrone, P. “What Is Podcasting?” *O'Reilly Media*, 20 July 2005 <<http://digitalmedia.oreilly.com/2005/07/20/WhatIsPodcasting.html>>.

casting has introduced a more democratic form of radio broadcasting. Podcasts are easily created with audio software and can be played on a computer or portable media player. The files are posted online at individual websites or distributed via programs like Apple's iTunes. Listeners can often subscribe to podcasts via RSS feeds. Forrester Research predicted 12 million households will be regularly subscribing to podcasts by 2010.⁹³

3.9 Tagging

History of Tagging

Tagging, or labeling content, is part of the collaborative nature of Web 2.0. A tag is any user-generated word or phrase that helps organize web content and label it in a more human way. Though standard sets of labels allow users to mark content in a general way, tagging items with self-chosen labels creates a stronger identification of the content. In an interview by the Pew Internet & American Life Project, David Weinberger (author of *Everything is Miscellaneous*) said:

“Maybe the most interesting thing about tagging is that we now have millions and millions of people who are saying, in public, what they think pages and images are about.”

—David Weinberger

As part of the same December 2006 report, 28% of Internet users had reportedly “tagged” content online.⁹⁴

Tag Clouds

Tag Clouds are visual displays of tags weighted by popularity. Many Web 2.0 sites include a graphical representation of popular tags (the popularity of the tag marked by the size of its text). There are many ways of forming tag clouds—terms often appear in alphabetical order. However, tag clouds show only how the majority (or the crowd) thinks and disregard many individual unique points of view.⁹⁵ Figure 3.3 is an example of a “text cloud” that we created manually from the major terms in this chapter. (To build your own text cloud try ArtViper’s TextTagCloud tool at <http://www.artviper.net/texttagcloud/>.)

Folksonomies

Folksonomies are classifications based on tags. The term is generally attributed to Thomas Vander Wal, who combined the words “taxonomy” and “folk” to create a new term for this Internet phenomenon.⁹⁶ Folksonomies are formed on sites such as Flickr, Technorati and del.icio.us. Users can search content by tags, which identify content in different (and sometimes more meaningful) ways than traditional keywords used by search engines.

An example of Web 2.0’s reach outside of traditional technology fields can be seen in the [steve.museum project](#), an experiment in tagging and folksonomies regarding museum

93. D’Agostino, D. “Security in the World of Web 2.0.” *Innovations*, Winter 2006, p.15.

94. Rainie, L. “Tagging.” *Pew Internet & American Life Project*, 31 January 2007. <http://www.pewinternet.org/pdfs/PIP_Tagging.pdf>.

95. Rainie, L. “Tagging.” *Pew Internet & American Life Project*, 31 January 2007. <http://www.pewinternet.org/pdfs/PIP_Tagging.pdf>.

96. Vander Wal, T. “Folksonomy Coinage and Definition.” *Vanderwal.net*, 2 February 2007 <<http://www.vanderwal.net/folksonomy.html>>.



Fig. 3.3 | Text cloud of major Web 2.0 terms from this chapter.

collections. In 2005, The Metropolitan Museum of Art and the Guggenheim Museum organized a retreat to plan the project.⁹⁷ In 2007 they posted various collections of art online and asked the community for help tagging them.

Flickr

Flickr—a popular photo-sharing site—was launched in February 2004 and acquired by Yahoo! in 2005. The Flickr development team was originally working on “The Game

97. Chun, S., R. Cherry, D. Hiwiller, J. Trant and B. Wyman. “Steve.museum: An Ongoing Experiment in Social Tagging, Folksonomy, and Museums.” *Archives & Museum Informatics*, 1 March 2006 <<http://www.archimuse.com/mw2006/papers/wyman/wyman.html>>.

Neverending”—a multiplayer Flash game based on IM (instant message) and chat interfaces.⁹⁸ However, the team listened to its users and developed real-time photo sharing (Flickr Live) and more traditional web pages where users could view uploaded pictures. The Game Neverending and Flickr Live were later retired as the popularity of photo sharing and commenting on the web pages grew.⁹⁹

Flickr is a key content-tagging site. Intended as a way of organizing personal photo collections, tagging on the site gained popularity as the community became interested in “a global view of the **tagscape**” (how other people are tagging photos).¹⁰⁰ Users can search for photos by meaningful tags. The tags also encourage loyalty to the site, since the tags are lost if photos are moved to another site.

Technorati

Technorati, a social media search engine, uses tags to find relevant blogs and other forms of social media. To become searchable by Technorati, bloggers can add tags to their posts with a simple line of HTML or use the automated category system offered by some blogging software packages.¹⁰¹ Technorati tag searches return results from the blogosphere, YouTube videos and Flickr photos. Technorati features a tag cloud on its homepage and a “where’s the fire” section to promote the most popular tags and search results.

3.10 Social Bookmarking

Social bookmarking sites let you share your Internet bookmarks (e.g., your favorite websites, blogs, and articles) through a website. Users can access these bookmarks from any computer and discover new sites by searching popular bookmarks and tags. Some of the most popular social bookmarking sites are del.icio.us, Ma.gnolia, Blue Dot, StumbleUpon, Simpy and Furl.

del.icio.us

del.icio.us, a self-described “collection of favorites,” reported its two-millionth user registration in March 2007.¹⁰² Users can add a bookmark by going to the site or by using the del.icio.us downloadable browser buttons. Some sites post clickable badges—a button provided by del.icio.us to “save this page”—that make it easy for users to bookmark the site using del.icio.us.

del.icio.us is a great example of a Web 2.0 company that uses tagging, social networking and user-generated content. When bookmarking a website, users can add notes and tags to describe the site. These tags are searchable and help organize sites, making it easier for users to find the content they want based on what other users have recommended (by bookmarking). Users can also add descriptions to tags, which can help clear up what

-
98. Schonfeld, E. “The Flickrization of Yahoo!” *CNN Money.com*, 1 December 2005 <http://money.cnn.com/magazines/business2/business2_archive/2005/12/01/8364623/>.
99. Garrett, J.J. “An Interview with Flickr’s Eric Costello.” *Adaptive Path*, 4 August 2005 <<http://www.adaptivepath.com/publications/essays/archives/000519.php>>.
100. Garrett, J.J. “An Interview with Flickr’s Eric Costello.” *Adaptive Path*, 4 August 2005 <<http://www.adaptivepath.com/publications/essays/archives/000519.php>>.
101. “Using Technorati Tags.” *Technorati* <<http://support.technorati.com/support/siteguide/tags>>.
102. “That was Fast.” del.icio.us blog, 29 March 2007 <http://blog.del.icio.us/blog/2007/03/that_was_fast.html>.

a certain tag might mean to different people. Thus, searching for content on del.icio.us is based on collaborative filtering rather than search engine algorithms. The site also offers a fully searchable podcasting section.

Third parties can use the del.icio.us web services API to build tools and incorporate social bookmarking functionality into their applications (see Section 3.13, Web Services, Mashups, Widgets and Gadgets). For example, Adobe Illustrator uses the del.icio.us technology to organize bookmarks in the program's documentation.¹⁰³

Ma.gnolia

"If searching was the first day of the web, people helping each other find what they want must be the second."

—Ma.gnolia¹⁰⁴

Ma.gnolia is another social bookmarking site offering tagging and convenient bookmark accessibility through the site. Bookmarked pages are saved (when possible) so users need not worry about losing content if a page goes offline. The site also provides browser buttons (bookmarklets) for posting sites to Ma.gnolia, and a “roots” feature, which lets you see what other users have said about a site while surfing the Internet. Ma.gnolia encourages social networking through user groups and a private messaging feature. To deal with spam, Ma.gnolia trusts handpicked moderators, called “gardeners.”¹⁰⁵

3.11 Software Development

A key to Web 2.0 software development is to KIS (keep it simple; keep it small). At the 2006 Emerging Technology Conference, Rael Dornfest (now CEO of the company “values of n” and former O’Reilly CTO) explained, “great businesses will be built on giving you less.”¹⁰⁶ This is particularly important given the “attention economy” (too much information, too little time)—the theme of the 2006 conference.

The Webtop

The web has now become an application, development, delivery, and execution platform. The **webtop**, or web desktop, allows you to run web applications in a desktop-like environment in a web browser. Using the web as a platform is part of a movement toward operating-system-independent applications. The removal of OS barriers allows the potential audience for any single product to become larger. An example of a popular webtop is the Laszlo Webtop (built on the OpenLaszlo framework), which runs applications written in OpenLaszlo as well as those written in other frameworks using XML requests.¹⁰⁷ Exam-

103. “Know How Adobe and del.icio.us Work Together?” del.icio.us blog, 30 May 2007 <http://blog.del.icio.us/blog/2007/05/knowhow_adobe_a.html>.

104. “About Ma.gnolia.” <<http://ma.gnolia.com/about>>.

105. “Gardeners.” Ma.gnolia Community Wiki, 29 March 2007 <<http://wiki.ma.gnolia.com/Gardeners>>.

106. Farber, D. “ETech: Attenuation, Web 2.0 and spimes.” ZDNet, 7 March 2006 <<http://blogs.zdnet.com/BTL/?p=2667>>.

107. “The RIA Desktop in a Browser.” *LaszloSystems* <<http://www.laszlosystems.com/software/webtop>>.

ples of Laszlo Webtop applications can be seen at <http://www.laszlosystems.com/showcase/samples>. Other webtops include eyeOS and StartForce.

Software as a Service (SaaS)

Software as a Service (SaaS), application software that runs on a web server rather than being installed on the client computer, has gained popularity, particularly with businesses. It provides many benefits, including fewer demands on internal IT departments, increased accessibility for out-of-the-office use, and an easy way to maintain software on a large scale.¹⁰⁸ Instead of being installed on the local machine, software is installed on the provider's web server and accessed by customers "as a service" over the Internet. Updates applied on the server impact every computer. This change from local to server machine makes it easier for large corporations to keep software updates uniform throughout the organization. Most Google software is offered as SaaS. Microsoft now offers SaaS products, Windows Live and Office Live.

Collaborating on projects with co-workers across the world is easier, since information is stored on a web server instead of on a single desktop. 37Signals has developed several SaaS products, including Basecamp (a product management and collaboration tool), Campfire (a group chat tool), Backpack (a personal organization tool), Ta-da (a "to-do" list tool), Highrise (a customer relations tool), and Writeboard (a collaborative word-processing tool). Salesforce.com, which specializes in Customer Relationship Management (CRM) software, is a key SaaS company—they provide popular business applications for sales, marketing, customer support, analytics and more.

Perpetual Beta and Agile Development

Due to the increased use of web applications there has been a shift away from the traditional software release cycle. Historically, companies would spend months or even years developing major new software releases. Because releases came so infrequently, each one had to go through extensive testing and beta periods to create a "final" release each time. There is now a greater focus on **agile software development**, which refers to development of fewer features at a time with more frequent releases. This "**perpetual beta**" of frequent smaller releases is made possible by using the web as a platform.¹⁰⁹ A new CD cannot be distributed to all customers every day; however, updates to web servers delivering the application can be easily made.

37Signals' *Getting Real*, an e-book that discusses agile techniques for building web applications, warns against the temptation to overuse "betas." The Internet is a dynamic medium—there will always be flaws and possible upgrades. Companies must decide how long it's really necessary to remain in a beta period, before it becomes just an excuse for a weak application. *Getting Real*, comprised of 91 short essays and numerous quotes and anecdotes, is a must read, providing an informative, insightful and entertaining walk through the software development process. The e-book can be read for free on their site or downloaded as a PDF for a fee.¹¹⁰

108. Peiris, M. "The Pros and Cons of Hosted Software." *SmartBiz*, March 2006 <<http://www.smartbiz.com/article/articleview/1118/1/42>>.

109. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.

110. 37Signals. *Getting Real*. 2006 <<http://gettingreal.37signals.com>>.

Open Source

The open source movement continues to gain momentum. The idea behind it is not new (it was popularized in 1998 with O'Reilly's Freeware Open Source Summit, now known as OSCON).¹¹¹ Historically, programs had been distributed by sharing the source code, before selling compiled programs became the norm. Though open source software is not always free, the source code is available (under license) to developers, who can customize it to meet their unique needs.

“Business-technology managers know all too well the adage about open source: It’s free, as in a free puppy. The work and expense start once you get it home.”

—Larry Greenemeier, *InformationWeek*¹¹²

Using open source projects, such as the popular Linux operating systems Red Hat or Ubuntu, may require more work and technical knowledge than using the Microsoft Windows or Apple Macintosh operating systems. However, advanced users are able to customize the software to fit their needs. Benefits to using an open source program include the possibility of reduced cost (if you have the skills to work with it) and the worldwide support networks where users help each other. Because the source code is available to everyone, users can look to the community for bug fixes and plug-ins (program extensions that add functionality), instead of waiting for the software vendor to address each issue. The Ubuntu forums, for example, contain a wealth of information created by users helping other users. In addition to the free support that springs up around open source projects, businesses have been built from developing project extensions and consulting. IBM invested \$1 billion in Linux in 2001.

“Linux can do for business applications what the Internet did for networking and communications.”

—Louis Gerstner, former CEO of IBM¹¹³

At <http://www.SourceForge.net> over 150,000 open source projects are under development. Other sites with open source downloads include freshmeat.net and Tucows. The popular Firefox web browser from the Mozilla Foundation, the Apache web server from the Apache Software Foundation, and the MySQL database system are all open source. DotNetNuke and PHPNuke offer open source frameworks for developing rich Internet portals, making it easy and economical to develop sophisticated websites. (<http://www.deitel.com> is a DotNetNuke site.)

Licensing: GNU Licenses and Creative Commons

Open source project licenses vary—many projects use the GNU General Public License (GPL), which allows redistribution of the project provided the source code is included and the copyright information is left intact. The Free Software Foundation provides other versions as well, including the GNU Lesser General Public License and the GNU Free Docs

111. Van Rossum, G. “Open Source Summit Trip Report.” *Linux Gazette*, 10 April 1998 <<http://linuxgazette.net/issue28/rossum.html>>.

112. Greenemeier, L. “Open-Source Exuberance.” *InformationWeek*, 11 July 2005 <<http://www.informationweek.com/story/showArticle.jhtml?articleID=165700923>>.

113. Wilcox, J. “IBM to Spend \$1 Billion on Linux in 2001.” *CNET*, 12 December 2000 <http://news.com.com/IBM+to+spend+1+billion+on+Linux+in+2001/2100-1001_3-249750.html>.

umentation License. The Open Source Initiative also lists over 50 licenses available to open source software developers, including the BSD license and the MIT license.¹¹⁴

Creative Commons (creativecommons.org) deals with licensing issues for all types of digital media. The organization offers a variety of options to support **remixing** (extending existing content), commercial issues and attribution. By allowing users access to general licenses through Creative Commons or the Free Software Foundation, developers can worry less about the complicated issues of licensing and instead focus on developing.

3.12 Rich Internet Applications (RIAs)

Rich Internet Applications (RIAs) are web applications that offer the responsiveness, “rich” features and functionality approaching that of desktop applications. Early Internet applications supported only a basic HTML **graphical user interface (GUI)**. Though they could serve simple functions, these applications did not have the look or feel of a desktop application. The relatively slow Internet connections these applications relied on led to the term “World Wide Wait.” RIAs are a result of today’s more advanced technologies that allow greater responsiveness and advanced GUIs.

Ajax

The term **Ajax (Asynchronous JavaScript and XML)** was coined by Adaptive Path’s Jesse James Garrett in February 2005. Ajax (see Chapter 15, Ajax-Enabled Rich Internet Applications) allows partial page updates—meaning updates of individual pieces of a web page without having to reload the entire page. This creates a more responsive GUI, allowing users to continue interacting with the page as the server processes requests.

The technologies that make up Ajax—XHTML, CSS, JavaScript, the DOM, XML, and the XMLHttpRequest object—are not new. In fact, in the 1990s, Netscape used asynchronous page updates in LiveScript, which evolved into JavaScript. However, the popularity of Ajax has dramatically increased since its naming. Ajax performs a vital role in Web 2.0, particularly in building webtop applications and enhancing the user’s overall experience. The following toolkits and **frameworks** (environments with standard components that make development faster and easier) provide libraries and tools for convenient Ajax-enabled application development.

Dojo

Dojo is an open source JavaScript toolkit—it is a library, not a framework. Dojo development began in late 2004.¹¹⁵ Dojo helps standardize JavaScript by providing a variety of packages for cross-browser compatibility, rich GUI controls, event handling and more. (See the Dojo section in Chapter 15.)

Flex

Adobe **Flex** (see Chapter 18) is an RIA framework that allows you to build scalable, cross-platform, multimedia-rich applications that can be delivered over the Internet. It uses the Flash Player 9 runtime environment, which is installed on over 97% of computers, allow-

114. Tiemann, M. “Licenses by Name.” *Open Source Initiative*, 18 September 2006 <<http://www.opensource.org/licenses/alphabetical>>.

115. “History.” *The Dojo Toolkit*, 10 April 2007 <<http://dojotoolkit.org/book/dojo-book-0-9/introduction/history>>.

ing for almost universal compatibility.¹¹⁶ Flash Player 9 is backed by ActionScript 3, Adobe's object-oriented scripting language—this uses an asynchronous programming model, which allows for partial page updates similar to Ajax. Flash CS3 (the development tool for creating Flash movies) is discussed in Chapters 16–17.

Silverlight

Microsoft's **Silverlight** (see Chapter 19), formerly known as Windows Presentation Foundation Everywhere (WPF/E) and released in May 2007, is Microsoft's new competitor to Flex and Flash. Silverlight 1.1 uses a compact version of the .NET framework. Silverlight applications have user interfaces built in Extensible Application Markup Language (XAML)—Microsoft's XML-based format for describing user interfaces. The new framework allows quick and easy development of RIAs and is designed to run on major browsers and operating systems.¹¹⁷ **Moonlight**, an open source version of Silverlight for Linux operating systems, is being developed.

JavaFX

JavaFX is Sun Microsystems' counterpart to Flex and Silverlight, also designed for building Rich Internet Applications. It consists of the JavaFX Script and JavaFX Mobile (for mobile devices). The JavaFX Script, which takes advantage of the fact Java is installed on most computers, will be available under open source licences (see <https://openjfx.dev.java.net/>).¹¹⁸

Ruby on Rails

Ruby on Rails (see Chapter 24), developed by 37Signals' David Heinemeier Hansson, is an open source framework based on the Ruby scripting language that allows you to build database-intensive applications quickly, easily, and with less code. Ruby on Rails was designed to build 37Signals' Basecamp (a project management and collaboration tool) and other SaaS products.

Script.aculo.us

The **Script.aculo.us** library for creating “eye candy” effects is built on the Prototype JavaScript framework. Prototype encapsulates the DOM (Document Object Model, Chapter 12) and provides cross-browser processing capabilities.¹¹⁹ Script.aculo.us uses this framework and adds capabilities for rich user interfaces. Its core effects include opacity, scale, morph, move, highlight and parallel (for combining multiple effects).¹²⁰ Script.aculo.us is used on many popular websites and is incorporated into other frameworks (such as Ruby on Rails). We discuss Script.aculo.us and present examples in Chapter 24, Ruby on Rails.

116. “Adobe Flex 2.” *Adobe* <http://www.adobe.com/products/flex/whitepapers/pdfs/flex2wp_technicaloverview.pdf>.

117. Cubrilovic, N. “Silverlight: The Web Just Got Richer.” *TechCrunch*, 30 April 2007 <<http://www.techcrunch.com/2007/04/30/silverlight-the-web-just-got-richer>>.

118. “Sun Radically Simplifies Content Authoring—Previews JavaFX Script.” *Sun Microsystems*, 8 May 2007 <<http://www.sun.com/aboutsun/pr/2007-05/sunflash.20070508.2.xml>>.

119. “Prototype Tips and Tutorials.” *Prototype JavaScript* <<http://prototypejs.org/learn>>.

120. “Core Effects.” *Script.aculo.us Wiki* <<http://wiki.script.aculo.us/scriptaculous/show/CoreEffects>>.

JavaServer Faces

JavaServer Faces (JSF) is a Java-based web application framework. JSF separates design elements from business logic and provides a set of user-interface components (JSF components) that make developing RIAs simple. One of the **Java BluePrints** projects provides additional resources and libraries for building Ajax-enabled applications. We build RIAs with JSF in Chapters 26–27.

ASP.NET Ajax

ASP.NET Ajax (Chapter 25) is an extension of the .NET framework for creating Ajax-enabled applications. It includes an open source Ajax Control Toolkit for implementing asynchronous functionality. ASP.NET Ajax is easily used in Microsoft Visual Web Developer or Microsoft Visual Studio to quickly create Rich Internet Applications.

Adobe Integrated Runtime and Google Gears

Though web application use has been increasing, many feel these programs cannot truly compete with desktop applications until the “Offline Problem” (not being able to access web applications and data when not connected to the Internet) has been solved.¹²¹ Businesses can lose valuable time and money when Internet issues occur such as a slow or broken Internet connection.

Adobe released its **Adobe Integrated Runtime (AIR)**; previously called Apollo) in beta form in June 2007. AIR allows users to run Flex web applications on their desktops even when they are *not* connected to the Internet, thus allowing users to remain efficient when they are unable to access the Internet or when an SaaS application server goes down. Users can continue their work and synchronize it with the servers again later.

Google Gears, also in beta, is a similar product, allowing use of web applications while offline. Google Gears was created out of a Google engineer’s 20% project, inspired by wanting to use Google Reader on a bus with “flaky” Internet access.¹²² (Google engineers devote 20% of their time to projects other than their usual work and 10% of their time to projects that are “truly new.”)¹²³ **Dojo Offline** (using the Dojo library) is built on top of Google Gears, creating an easy-to-use interface for using web applications offline.¹²⁴

3.13 Web Services, Mashups, Widgets and Gadgets

“Design for ‘hackability’ and remixability.”

—Tim O'Reilly¹²⁵

Instead of reinventing the wheel with every new project, developers can use existing companies’ **web services** to create feature-rich applications. Incorporating web services into new programs allows people to develop new applications quickly.

121. Berlind, D. “Google Gears Vies to be De Facto Tech for Offline Web Apps.” *ZDNet*, 31 May 2007 <<http://blogs.zdnet.com/Berlind/?p=504>>.

122. Mills, E. “Google Gears Churns Toward Microsoft.” *CNET*, 31 May 2007 <http://news.com.com/2100-1012_3-6187942.html>.

123. “The 70 Percent Solution.” *Business 2.0*, 28 November 2005 <http://money.cnn.com/2005/11/28/news/newsmakers/schmidt_biz20_1205>.

124. “The Dojo Offline Toolkit.” *The Dojo Toolkit* <<http://dojotoolkit.org/offline>>.

125. O'Reilly, T. “What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.” September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.

APIs

APIs (**A**pplication **P**rogramming **I**nterfaces) provide applications with access to external services and databases. For example, a traditional programming API, like the Sun's Java API, allows programmers to use already-written methods and functions in their programs. Web services APIs are now offered by some websites as ways of sharing some of their functionality and information across the Internet.

Unique databases are central to Web 2.0; "data is the next Intel Inside."¹²⁶ Whether data is obtained from a proprietary source or collected over time from users, much of a site's value is in its databases. Many major Web 2.0 companies (e.g., eBay, Amazon, Google, Yahoo! and Flickr) provide APIs to encourage use of their services and data in the development of mashups, widgets and gadgets.

Mashups

Mashups combine content or functionality from existing web services, websites and RSS feeds to serve a new purpose. For example, [Housingmaps.com](http://www.housingmaps.com) is a mashup of Google Maps and real-estate listings from Craigslist. Mashups with maps are particularly popular, as are mashups using RSS feeds (see "RSS and Atom" in Section 3.15) created by using services such as Yahoo! Pipes™—a tool that enables you to aggregate and manipulate many data sources.

Using APIs can save time and money (some great mashups have been built in an afternoon); however, the mashup is then reliant on one or more third parties. If the API provider experiences downtime, the mashup will be unavailable as well (unless the mashup is programmed to avoid sites that are down). Always check the "terms of service" for using each company's web services. Many API providers charge usage fees based on the mashup's number of calls made to the server. Some sites require you to ask permission before using their APIs for commercial purposes, and others (e.g., Google) require that mashups based on their web services be free. Also, while mashups add value to data, there is always the question of who owns the data, and thus who should profit from the mashup.

Figure 3.4 lists some popular mashups. The site Programmable Web catalogs APIs and mashups and offers a "Mashup Matrix" (<http://www.programmableweb.com/matrix>) detailing which APIs have been combined to form each mashup. As more companies offer APIs, the only limitation on mashups (and the businesses built on them) is the developer's creativity. More complex mashups, using programs like Google Earth and Second Life, could be coming soon.¹²⁷

Mashup	Combines
http://www.housingmaps.com	Google Maps and Craigslist real-estate listings to create a map marked with available housing listings.

Fig. 3.4 | Mashup examples. (Part 1 of 2.)

126. O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.

127. Roush, W. "Second Earth." *Technology Review*, July/August 2007, p.38.

Mashup	Combines
http://www.chicagocrime.org	Google Maps and crime data from Citizen ICAM to create a map of Chicago marked with crime locations.
http://www.feedmashr.com	RSS feeds from Digg, ClipMarks, the <i>New York Times</i> , del.icio.us, Reddit and Slashdot to create a listing of the most popular stories from all sources.
http://www.secretprices.com	Amazon, Epinions.com and Shopping.com to create a comparison shopping site.
http://paul.kedrosky.com/publicloos/	Google Maps and Bathroom Diaries to create a map of San Francisco marked with the locations of public restrooms.

Fig. 3.4 | Mashup examples. (Part 2 of 2.)

Widgets and Gadgets

Widgets, also referred to as **gadgets**, are mini applications designed to run either as stand-alone applications or as add-on features in web pages. *Newsweek* called 2007 the “Year of the Widget” because of the huge increase in popularity of these applications.¹²⁸ Widgets can be used to personalize your Internet experience by displaying real-time weather conditions, aggregating RSS feeds, viewing maps, receiving event reminders, providing easy access to search engines and more. The availability of web services, APIs and various tools makes it easy even for beginner programmers to develop widgets. There are many catalogs of widgets online—one of the most all-inclusive is **Widgipedia**, which provides an extensive catalog of widgets and gadgets for a variety of platforms.

Amazon Web Services

Amazon is a leading provider of web services. The site provides historical pricing data and E-Commerce Services (ECS), which enable companies to use Amazon’s systems to sell their own products. Amazon also offers hardware and communications infrastructure web services that are particularly popular with companies, providing economical **web-scale computing**. Amazon’s Elastic Compute Cloud (EC2), Simple Storage Service (S3) and Simple Queue Service (SQS) enable businesses to pay for only the processing or storage space needed during any given period. This makes it possible for companies to save money (by not having to buy and maintain new hardware, software and communications equipment) while still being able to scale their storage and computing power to handle traffic surges (or reduce loss when the site’s popularity declines). This is extremely significant in the Internet world, where a site’s traffic can explode or crash overnight.

128. Braiker, B. “Tech: Welcome, Year of the Widget.” *Newsweek*, 30 December 2006 <<http://www.msnbc.msn.com/id/16329739/site/newsweek/>>.

Amazon also provides “artificial artificial intelligence” with its unique **Mechanical Turk**. This web service allows applications to call on people to perform tasks (such as identifying pictures) that are easier for humans to do than computers. People can sign up to become part of the Mechanical Turk web service and bid on jobs (called Human Intelligence Tasks or HITs). This creates a competitive market, driving down developer costs, creating opportunities for people worldwide and allowing more applications to become feasible.

REST (Representational State Transfer)-Based Web Services

Representational State Transfer (REST) (originally proposed in Roy Thomas Fielding’s doctoral dissertation¹²⁹) refers to an architectural style for implementing web services. Though REST is not a standard, RESTful web services are implemented using web standards. Each operation in a RESTful web service is easily identified by a unique URL. So, when the server receives a request, it immediately knows what operation to perform. Such web services can be used in a program or directly from a web browser. In some cases, the results of a particular operation may be cached locally by the browser. This can make subsequent requests for the same operation faster by loading the result directly from the browser’s cache.¹³⁰ Amazon’s S3 is RESTful, and many other Web 2.0 web services provide RESTful interfaces.¹³¹

RESTful web services are alternatives to those implemented with SOAP (Simple Object Access Protocol). (We discuss both REST-based and SOAP-based web services in Chapter 28, Web Services.) With SOAP-based web services, the request and response are hidden (in entities known as a SOAP “envelopes”). SOAP requests must be deciphered as they are received at the server to determine the operation to perform and the arguments required to perform that operation. Similarly, the responses are encoded and deciphered on the client to obtain the result of the operation. SOAP does not currently provide a mechanism for caching results.

3.14 Location-Based Services

Location-Based Services (LBS) are applications that take your geographic location (city, state, location of your mobile device, etc.) into consideration. While the term generally refers to services accessed on mobile devices using the Global Positioning System (GPS), it can also be used to describe web applications that take your location into account. Search engines including Yahoo! Local and Google Maps use **localization** to provide you with geographically relevant content. Local search is particularly useful when you want to find a nearby business (e.g., plumbers, taxis, etc.). Location-based services are becoming increasingly popular in Web 2.0 applications. Conferences related to LBS include O’Reilly’s Where 2.0 and the Location Intelligence Conference.

Global Positioning System (GPS)

The **Global Positioning System (GPS)**, developed by the United States Department of Defense, uses numerous satellites that send signals to a GPS receiver to determine its exact

129. Fielding, R. T. “Architectural Styles and the Design of Network-Based Software Architectures.” <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

130. Costello, R. “REST Tutorial.” *xFront*, 26 June 2002 <http://www.xfront.com/REST.html>.

131. Richardson, L. and S. Ruby. *RESTful Web Services*. O’Reilly, 2007.

location. (A Russian system called GLONASS also exists, and a new system named Galileo is under development in Europe.) In the 1980s, the US Department of Defense opened GPS for civilian use to encourage satellite technology development.¹³² Numerous location-based services are now available using GPS technology, such as GPS mapping devices used in cars or on mobile devices. GPS is also being used for safety. The US Federal Communications Commission (FCC) now requires wireless carriers to provide the locations of wireless customers calling 911 so emergency services can find them faster. To meet this requirement, wireless carriers have developed GPS-enabled cell phones.¹³³ These phones also provide premium services, such as driving directions and local information. The Disney Family Locator service uses GPS-enabled phones to help parents keep track of their children (as long as the child is carrying the special cell phone).¹³⁴

Mapping Services

Google Maps is one of the most popular mapping applications available online. You can use Google Maps to locate businesses in your area, get driving directions and live traffic information, create custom maps with images and more. You can even get the information by using your mobile device. Google's local search allows you to locate a business in a geographic area and get its address, phone number, driving directions and even user reviews. **Google Earth** provides satellite images of virtually any location on the planet. In some areas, you can even get a panoramic view of a neighborhood at street level. You can use the Google Maps API to add mapping capabilities to your websites and web applications.

MapQuest, owned by AOL, provides similar mapping services. Use it to get directions and maps on your desktop or mobile device. The MapQuest OpenAPI allows you to add location-based services to your web applications. Additional mapping services include Yahoo! Local Maps and MSN Live Search. Both services offer maps, driving directions, traffic information and local search.

Companies such as **NAVTEQ** and **Tele Atlas** provide digital map data for in-vehicle and portable navigation devices, websites, location-based services and more. Developers building commercial location-based services can license the robust mapping products from these companies to build richly functional web applications.

GeoRSS and Geotagging

GeoRSS, based on the RSS standards, is a set of standards for representing geographical information in a feed. Location and geographical information in a GeoRSS feed can be used in GPS devices, mapping applications and other location-based services. For example, a blog post about a vacation could map the locations mentioned.¹³⁵

Geotagging can be used to add location information (longitude, latitude, etc.) to websites, images, RSS feeds, videos and more. Websites can often determine a user's location by their IP address. Geotagging a website provides the user with location information about the site.¹³⁶ Geographic information can be used to add value to search results. Geotagging

132. Schiller, J. and A. Voisard. *Location-Based Services*. Morgan Kaufmann, 2004.

133. Malykhina, E. "Nokia Wants Your Cell Phone To Tell You Where You Are." *InformationWeek*, 9 October 2006 <<http://www.informationweek.com/showArticle.jhtml?articleID=193105219>>.

134. Magid, L. "Global Positioning by Cellphone." *New York Times*, 19 July 2007, C7.

135. "GeoRSS: Geographically Encoded Objects for RSS Feeds." *GeoRSS* <<http://georss.org/>>.

136. Turner, A. "Geotagging Web Pages and RSS Feeds." *Linux Journal*, 11 January 2005 <<http://interactive.linuxjournal.com/node/8025>>.

could also be mashed up with existing visualization systems, such as Google Earth or MSN Virtual Earth, which provide advanced satellite images for anywhere on the planet.

3.15 XML, RSS, Atom, JSON and VoIP

For more information on any of the following technologies, visit the corresponding Resource Centers at <http://www.deitel.com/resourcecenters.html> (see Fig. 2 in the Preface for a complete list of Deitel Resource Centers).

XML

XML (Extensible Markup Language, Chapter 14), developed in 1996 by the World Wide Web Consortium (W3C), is a markup language that allows you to label data based on its meaning. XML describes data in a way that is meaningful to both humans and computers.

XML documents are text files with a .xml extension; they can be created in text editors. These documents can reference a **Document Type Definition (DTD)** or a **schema**, which defines the structure for the document. This allows the information in the document to be verified by validating parsers, meaning they will check to make sure that no elements are missing (e.g., a last-name element in a document listing full names) and that the elements occur in the proper order. This makes XML data more reliable than data prepared with some other data-describing options. XML also can be used to create customized markup languages (e.g., XHTML for web content, CML for chemistry, MathML for mathematical content and formulas, and XBRL for financial data)—these are referred to as **XML vocabularies**. XHTML is described in Chapter 4, Introduction to XHTML. Chapter 14, XML and RSS, presents several examples that use MathML to render mathematical expressions.

RSS and Atom

Sites that offer RSS (Chapter 14) and Atom feeds can maintain an “open connection” with their readers. Users no longer have to regularly visit sites for updates—by subscribing to a site’s feed, users receive updates as new information is posted to the site. The difference between RSS and Atom is subtle and unimportant to most users—many tools support both formats. Versions of RSS (an XML-based web content syndication format) have existed since the late 1990s; Atom dates to 2003.

Most major web browsers support RSS and Atom feeds, and many **aggregators** (or feed readers) are available to help users organize their subscriptions. Feedburner (acquired by Google) is used by many blogs to provide their readers with new posts by e-mail. This service also helps bloggers get a better idea of the size of their audience (by allowing them to see the number of subscribers).

JSON

JavaScript Object Notation (JSON) was developed in 1999 as an alternative to XML. JSON (discussed in Chapter 15, Ajax-Enabled Rich Internet Applications) is a text-based data interchange format used to represent JavaScript objects as strings and transmit them over a network. It is commonly used in Ajax applications. JSON text is easy to produce and read—it is also faster to parse (or extract) than XML.

VoIP

Voice over Internet Protocol (VoIP) is the technology used to make free or inexpensive phone calls over the Internet. Some businesses and individuals have switched completely

to VoIP and eliminated traditional phone lines to cut costs. There are many VoIP services, such as Vonage, Packet8 or Lingo; Skype is the most popular. Acquired by eBay to integrate buyer and seller voice communication into auctions,¹³⁷ Skype offers free and fee-based services (such as calling non-Skype phones). VoIP is an enabling technology that can be layered into Web 2.0 companies and websites.

3.16 Web 2.0 Monetization Models

“The advertising model has come along; we underestimated how big that would be.”

—Bill Gates, MIX06

Many Web 1.0 businesses discovered that popularity (“eyeballs”) was not the same as financial success. Web 2.0 companies are paying more attention to **monetizing** their traffic. Starting an Internet business is cheaper than ever, and the cost of failure is lower. Anyone can start earning modest amounts of money almost immediately, using the monetization models described in Fig. 3.5.

Web 2.0 monetization is heavily reliant on advertising. Using Google’s AdSense contextual advertising program is one of the fastest and most popular ways of monetizing a new Internet business. For more information see Deitel’s Google AdSense and Website Monetization Resource Centers at <http://www.deitel.com/resourcecenters.html>.

Web 2.0 monetization models	
affiliate network —A business (such as Commission Junction and LinkShare) that connects web publishers with cost-per-action affiliate programs. See affiliate program .	banner ad —An ad that consists of an image, often placed at the top of a page.
affiliate program —A deal offered by a company to share a portion of the revenues earned from traffic coming from web publisher websites. Affiliates provide text and image ads to post on the publishers’ sites. If a user clicks through to the affiliate site and takes a specified action (e.g., makes a purchase, fills out a registration form, etc.) the publisher is paid a portion of the revenue or a flat fee. Companies offering affiliate programs include Amazon (the Amazon Associates program), Indeed, ClickBank, eBay and thousands more.	blog advertising —Advertising specifically designed for display on blog sites. Companies include Federated Media and Blogads.
	contextual advertising —Advertising that is targeted to the content on a web page. Contextual ad programs include Google AdSense, Yahoo! Publisher Network, Vibrant Media, Kontera and Tribal Fusion.
	cost-per-action (CPA) —Advertising that is billed to the advertiser per user action (e.g., purchasing a product or filling out a mortgage application). Companies include Amazon and Indeed. See also performance-based advertising .

Fig. 3.5 | Web 2.0 monetization models. (Part I of 2.)

137. Broache, A. “eBay to Nab Skype for \$2.6 Billion.” *CNET*, 12 September 2005 <http://news.com.com/eBay+to+nab+Skype+for+2.6+billion/2100-1030_3-5860055.html>.

Web 2.0 monetization models

cost-per-click (CPC)—Advertising that is billed by user click. The web publisher receives revenue each time a user clicks an ad on the publisher's site, regardless of whether the user makes a subsequent purchase. Companies include Google AdSense and Yahoo! Publisher Network.

cost-per-thousand impressions (CPM)—Advertising (usually banner advertising) that is billed per thousand impressions, regardless of whether the user clicks on the ad. Companies include DoubleClick, ValueClick and many more.

e-commerce—Selling products and/or services directly through a website. Companies include Amazon, Dell, CafePress.com and thousands more.

interstitial ad—An ad that plays between page loads. Companies include Tribal Fusion, DoubleClick, and many more.

in-text contextual advertising—Advertising that is marked by double-underlined keywords or phrases in the content of a web page. When a reader hovers the mouse cursor over a double-underlined word or phrase, a text ad pops up. By clicking on an ad, readers are taken to the advertiser's page. Companies providing in-text contextual advertising include Vibrant Media, Text Link Ads, Kontera and Tribal Fusion.

lead generation—Leads are generated when a visitor fills out an inquiry form so that a salesperson can follow through and potentially convert the lead to a sale. Lead generation is a subset of cost-per-action advertising. See cost-per-action (CPA).

paid blog post—A blog post (often a product review) that an advertiser pays a blogger to write. Some argue the ethics of this practice, and bloggers are encouraged to disclose that they are being paid for the posts. Companies that match bloggers and advertisers include PayPerPost, SponsoredReviews and ReviewMe.

performance-based advertising—Advertising that pays based on user action, such as making a purchase, filling out a registration form, etc. These are also often part of affiliate programs such as Amazon and ClickBank. See cost-per-action (CPA).

premium content—Content on a website that is available for an extra fee (e.g., e-books, articles, etc.). Companies that offer premium content include *The Wall Street Journal Online* and Search Engine Watch.

RSS ad—An ad included in RSS feeds. Companies include Feedster, Feedburner and Yahoo! Search Marketing.

tagging for profit—A site that buys inbound links or tags from other sites to help increase traffic, and thus increase potential advertising revenue. High-traffic sites can sell tags or links to other websites for a profit. (Caution: Search engines may lower the ranking of sites with paid links.) An example is 1000tags.com.

virtual worlds monetization—Selling products, premium services, virtual land and more in an online virtual world website. Virtual worlds include Second Life, IMVU, Habbo, Gaia Online and There.

Fig. 3.5 | Web 2.0 monetization models. (Part 2 of 2.)

3.17 Web 2.0 Business Models

The technologies and collaborative nature of Web 2.0 have opened up new business models. Some of these would not have been feasible even ten years ago, but because of Moore's Law they are not only possible but thriving. At the moment, there is no foreseeable end to the advancements attributed to Moore's Law, so fantastic ideas that are impossible today may become possible within just a few years. Figure 3.6 outlines many popular Internet

business models and lists some companies that use each one. In just about every case, there are many more companies using that business model.

Web 2.0 business models

advertising exchange—An online marketplace where web publishers can sell their advertising inventory (ad space) to advertisers. Companies include DoubleClick Advertising Exchange and Right Media Exchange.

affiliate network—A business that connects web publishers with cost-per-action affiliate programs, which are a form of cost-per-action advertising. Companies include Commission Junction and LinkShare. (See Fig. 3.5 for more information on affiliate programs.)

blog—A website with a series of posts in reverse chronological order. Many blogs attract significant traffic and monetize with advertising and affiliate programs. Popular blogs include BoingBoing, Gizmodo, TechCrunch, John Battelle's Searchblog, Problogger and Scobleizer.

blog search engine—A search engine devoted to the blogosphere. Companies include Technorati, Feedster, IceRocket and Google Blog Search.

blog network—A collection of blogs with multiple editors. Popular blog networks include Corante, 9rules, Gawker Media and Weblogs, Inc.

buying and selling domain names—A company purchases domain names with the intent of selling them in the future as Internet real estate becomes more valuable. Companies include Afternic.com and GreatDomains.

competitive intelligence—A company that analyzes Internet usage for use by client websites. Companies include Hitwise and Compete, Inc.

content network—A site (or collection of sites) that provides content including articles, wikis, blogs and more. Companies

include About.com, Deitel, LifeTips and Suite101.

discovery—A site that introduces users to valuable content they would not have looked for otherwise. Sites include StumbleUpon, Aggregate Knowledge, MOG and Deitel.

domain registrar—A site that sells domain names. Companies include Register.com, GoDaddy and Network Solutions.

encyclopedia and reference source—An online reference encyclopedia, dictionary, thesaurus, etc. Sites include Wikipedia, Reference.com and Citizendium.

feed aggregator—An application that combines RSS or Atom feeds so the user can view all subscriptions in a single location. Applications include NetNewsWire, Google Reader and Bloglines.

file sharing—An application where users can share files, music, software and more. Companies include BitTorrent, LimeWire, Kazaa, AllPeers and Shareaza.

infrastructure for distributing open source projects—A site that hosts collaborative open source software projects. Sites include SourceForge, freshmeat.net and Tucows.

Internet and web conference organizer—A company that organizes conferences on Internet and web topics. Companies include O'Reilly Media, CMP and Jupiter.

Internet radio—A site that distributes music and radio shows over the Internet. Companies include Last.fm and Pandora.

Internet TV—A site that distributes television shows (or allows you to distribute your own shows) over the Internet. Companies include Joost and Brightcove.

Internet video—A video sharing site where users upload and share content. Companies include YouTube and Yahoo! Video.

Fig. 3.6 | Web 2.0 business models. (Part 1 of 4.)

Web 2.0 business models

job boards and job search—A site that connects job seekers with employers and/or job search engines. Job boards include Monster, CareerBuilder and Dice. Job search engines include Indeed, Jobster and SimplyHired.

mashup—A combination of two or more existing web services and feeds to create a new application. For example, <http://www.housingmaps.com> combines real estate listings from Craigslist with Google Maps so you can view the listings on a map. For a list of popular mashups, see <http://www.programmableweb.com/popular>.

massively multiplayer online game—An online role playing or strategy game where Internet users interact with one another. Games include World of Warcraft, Guild Wars and Lineage.

mobile social networking—A social network oriented towards mobile devices (such as cell phones). Companies include Twitter, Dodgeball and MocoSpace.

music distribution site—An online music site where you can purchase electronic versions (e.g., .mp3) of single songs or entire albums. Companies include iTunes, Rhapsody and Amie Street.

online advertising—An online advertising company that offers contextual advertising, banner advertising, in-text contextual advertising and more. Companies include Google, Yahoo!, Microsoft, DoubleClick, Vibrant Media, Tribal Fusion, Kontera, Quigo, ValueClick, Federated Media and many more.

online auction—A marketplace where visitors bid for products (and services) over the Internet. Companies include eBay, Overstock.com and Amazon Auctions.

online classifieds—A classifieds “advertising” site where users can post jobs, real-estate listings, personal ads, etc. Companies include Craigslist, Yahoo! Classifieds and Google Base.

online survey site—A site that offers survey services to other companies. A popular example is Survey Monkey.

open source—Software that is available (under license) for anyone to use and modify with few or no restrictions. Many Web 2.0 companies use open source software to power their sites and offer open source products and content. Companies include the Free Software Foundation, Apache, Mozilla, Zend and many more.

outsourcing marketplaces—An online marketplace where contractors and freelancers can connect with potential clients for short-term work. Companies include Elance and Guru.com.

payments—A site that handles secure payments for e-commerce sites. Companies include PayPal and Google Checkout.

people-assisted search—A search engine or search-driven content site that is filtered and organized by people to provide users with more relevant search results. Companies include Mahalo and Deitel.

personalized start page—A site that allows you to customize a start page with weather, news, etc. Companies include Netvibes, iGoogle, Pageflakes and Protopage.

photo sharing site—A site where users can post and share their photos with other users. Companies include Flickr and Photobucket.

real estate—A site that offers online real estate listings and information. Companies include Redfin, Trulia and Zillow.

recommender system—A system that collects data using collaborative filtering systems to determine users’ tastes and interests. Sites can gather information about your personal interests, compare you to other users with similar interests and make recommendations. Popular examples of sites using recommender systems include Pandora, Netflix, CleverSet, ChoiceStream, MyStrands, StumbleUpon, Last.fm, and MovieLens.

Fig. 3.6 | Web 2.0 business models. (Part 2 of 4.)

Web 2.0 business models

reputation system—A system used by businesses like eBay and Amazon to encourage trust. For example, after each eBay transaction, the buyer and the seller can each leave positive or negative comments about the other party.

search engine—The primary tool people use to find information on the web. Companies include Google, Yahoo!, MSN, Ask and many more.

selling digital content—An e-commerce site that sells digital media (e.g., e-books). Companies include ClickBank, Blish, Lulu and more.

social bookmarking site—A site that allows users to share their bookmarks with others. Users bookmark their favorite sites, articles, blogs and more, and tag them by keyword. Companies include del.icio.us, Ma.gnolia and Blue Dot.

social media site—A site that allows digital media (text, photos, videos, music, etc.) to be shared online. Companies include Digg, YouTube, Flickr, Reddit, Wikipedia and more.

social networking site—A site that helps users organize their existing relationships and establish new ones. Companies include MySpace, Facebook, Bebo, LinkedIn, Second Life, Gaia Online and more.

Software as a Service (SaaS)—Software that runs on a web server rather than being installed on a local client computer. By modifying the version of the software on the server, a company can simultaneously update all users to the latest version. SaaS applications include Salesforce.com, Microsoft Office Live, Microsoft Windows Live, Zoho Office Suite and many Google and 37Signals products.

subscription site—A site that offers member-only areas and premium content (additional content for a fee). Examples include Safari Books Online and the *Wall Street Journal*.

travel site—An online travel resource site that allows you to find and book hotels, air travel, rental cars and more. Companies include Expedia, Travelocity and Orbitz.

vertical search engine—A search engine that allows you to focus your search on a narrow topic. For example, travel search engines include Yahoo! Fare Finder, SideStep and Kayak; source-code search engines include Krugle and Koders.

virtual world—A social networking site (or program) where users create an avatar (their online image and persona) that they use to meet other users with similar interests, conduct business, participate in group activities, take classes and more. Companies include Second Life, Habbo, Gaia Online and There.

Voice over Internet Protocol (VoIP) site—A site that offers inexpensive or free telephone services over the Internet. Companies include Skype, Packet8, Lingo and Vonage.

Web 2.0 software—Software designed to build Web 2.0 sites and applications (e.g., blogging software). Companies include Six Apart, 37Signals, Adobe and Microsoft.

web analytics—Software (desktop and SaaS) and companies that analyze Internet traffic, demographics, navigation and more. Companies include Alexa, WebTrends, Click-Tracks, Google Analytics and WebSideStory.

web and mobile messaging—A service that allows you to chat with your contacts from various Internet messaging services (AIM, Yahoo! Messenger, MSN Messenger, Google Talk). Companies include Meebo and eBuddy.

web conferencing—An application that enables users to collaborate remotely. This often includes chat, VoIP and desktop sharing. Companies include WebEx, GoToMeeting and DimDim (open source).

Fig. 3.6 | Web 2.0 business models. (Part 3 of 4.)

Web 2.0 business models

webmail—A web-based e-mail system that allows you to send and receive e-mail using a standard browser. Popular webmail services include Google gmail, .Mac, Yahoo! Mail and MSN Hotmail.

wiki—A site that offers collaborative, editable documents online. Companies include Wikipedia, Wikia and SocialText.

Fig. 3.6 | Web 2.0 business models. (Part 4 of 4.)

3.18 Future of the Web

“Web 2.0 will make the cover of Time magazine, and thus its moment in the sun will have passed. However, the story that drives Web 2.0 will only strengthen, and folks will cast about for the next best name for the phenomenon.”

—John Battelle¹³⁸

“We’re a long way from the full realization of the potential of intelligent systems, and there will no doubt be a tipping point where the systems get smart enough that we’ll be ready to say, ‘this is qualitatively different. Let’s call it Web 3.0.’”

—Tim O'Reilly¹³⁹

The XHTML coding on websites defines their structure and layout, specifying colors, fonts, sizes, use of bold and italic, paragraphs, tables and the like, but *not* specifying the meaning of the data on the page. Web 1.0 servers sent mostly static web pages coded in HTML or XHTML to browsers that rendered the pages on the screen. Web 2.0 applications are more dynamic, generally enabling significant interaction between the user (the client) and the computer (the server), and among communities of users.

Computers have a hard time deciphering meaning from XHTML content. The web today involves *users'* interpretations of what pages and images mean, but the future entails a shift from XHTML to a more sophisticated system based on XML, enabling *computers* to better understand meaning.

Web 2.0 companies use “data mining” to extract as much meaning as they can from XHTML-encoded pages. For example, Google’s AdSense contextual advertising program does a remarkable job placing relevant ads next to content based on some interpretation of the meaning of that content. XHTML-encoded content does not explicitly convey meaning, but XML-encoded content does. So if we can encode in XML (and derivative technologies) much or all of the content on the web, we’ll take a great leap forward towards realizing the Semantic Web.

It is unlikely that web developers and users will directly encode all web content in XML—it’s simply too tedious and probably too complex for most web designers. Rather, the XML encoding will occur naturally as a by-product of using various content creation tools. For example, to submit a resume on a website, there may be a tool that enables the

138. Battelle, John. “2006 Predictions, How Did I Do?” *John Battelle Searchblog*, <<http://battellemedia.com/archives/003216.php>>.

139. O'Reilly, Tim. “Web 3.0 Maybe when we get there.” *O'Reilly Radar*, 13 November 2006 <http://radar.oreilly.com/archives/2006/11/web_30_maybe_wh.html>.

user to fill out a form (with first name, last name, phone number, career goal, etc.). When the resume is submitted, the tool could create a computer readable microformat that could easily be found and read by applications that process resumes. Such tools might help a company find qualified potential employees, or help a job seeker who wants to write a resume find resumes of people with similar qualifications).

Tagging and Folksonomies

Tagging and folksonomies are early hints of a “web of meaning.” Without tagging, searching for a picture on Flickr would be like searching for a needle in a giant haystack. Flickr’s tagging system allows users to subjectively tag pictures with meaning, making photos findable by search engines. Tagging is a “loose” classification system, quite different, for example, from using the Dewey Decimal System for cataloging books, which follows a rigid taxonomy system, limiting your choices to a set of predetermined categories. Tagging is a more “democratic” labeling system that allows people, for example, to associate whatever meanings they choose with a picture (e.g. who is in the picture, where it was taken, what is going on, the colors, the mood, etc.).

Semantic Web

“People keep asking what Web 3.0 is. I think maybe when you’ve got an overlay of scalable vector graphics—everything rippling and folding and looking misty—on Web 2.0 and access to a semantic Web integrated across a huge space of data, you’ll have access to an unbelievable data resource.”

—Tim Berners-Lee¹⁴⁰

“The Holy Grail for developers of the semantic Web is to build a system that can give a reasonable and complete response to a simple question like: I’m looking for a warm place to vacation and I have a budget of \$3,000. Oh, and I have an 11-year-old child... Under Web 3.0, the same search would ideally call up a complete vacation package that was planned as meticulously as if it had been assembled by a human travel agent.”

—John Markoff¹⁴¹

Many people consider the **Semantic Web** to be the next generation in web development, one that helps to realize the full potential of the web. This is Tim Berners-Lee’s original vision of the web, also known as the “[web of meaning](#).¹⁴² Though Web 2.0 applications are finding meaning in content, the Semantic Web will attempt to make those meanings clear to computers as well as humans. It will be a web able to answer complex and subtle questions.

Realization of the Semantic Web depends heavily on XML and XML-based technologies (see Chapter 14), which help make web content more understandable to computers. Currently, computers “understand” data on basic levels, but are progressing to find mean-

140. Shannon, V. “A ‘More Revolutionary’ Web.” May 2006 <<http://www.iht.com/articles/2006/05/23/business/web.php>>.

141. Markoff, John. “Entrepreneurs See a Web Guided by Common Sense.” The New York Times, November 2006 <<http://www.nytimes.com/2006/11/12/business/12web.html?ex=1320987600&en=254d697964cedc62&ei=5088>>.

142. Berners-Lee, T. *Weaving the Web*. Harper-Collins, 2000.

ingful connections and links between data points. The emerging Semantic Web technologies highlight new relationships among web data. Some experiments that emphasize this are Flickr and FOAF (Friend of a Friend), a research project that “is creating a Web of machine-readable pages describing people, the links between them and the things they create and do.”¹⁴³ Programming in both instances involves links between databases—ultimately allowing users to share, transfer, and use each other’s information (photos, blogs, etc.).¹⁴⁴

Preparations for the Semantic Web have been going on for years. XML is already widely used in both online and offline applications, but still only a minute portion of the web is coded in XML or derivative technologies. Many companies, including **Zepheira**, an information management company, and Joost, an Internet TV provider, already use semantic technologies in working with data. Deterring Semantic Web development are concerns about the consequences of false information and the abuse of data. Since the Semantic Web will rely on computers having greater access to information and will yield a deeper understanding of its significance, some people worry about the potentially increased consequences of security breaches. The **Policy Aware Web Project** is an early attempt at developing standards to encourage data sharing by providing access policies that can sufficiently protect individuals’ privacy concerns.¹⁴⁵

Microformats

“We need microformats that people agree on.”

— Bill Gates, MIX06 conference¹⁴⁶

Some people look at the web and see lots of “loose” information. Others see logical aggregates, such as business cards, resumes, events and so forth. **Microformats** are standard formats for representing information aggregates that can be understood by computers, enabling better search results and new types of applications. The key is for developers to use standard microformats, rather than developing customized, non-standard data aggregations. Microformat standards encourage sites to similarly organize their information, thus increasing interoperability. For example, if you want to create an event or an events calendar, you could use the hCalendar microformat. Some other microformats are adr for address information, hresume for resumes, and xfolk for collections of bookmarks.¹⁴⁷

Resource Description Framework (RDF)

The **Resource Description Framework (RDF)**, developed by the World Wide Web Consortium (W3C), is based on XML and used to describe content in a way that is understood by computers. RDF helps connect isolated databases across the web with consistent semantics.¹⁴⁸ The structure of any expression in RDF is a collection of triples.¹⁴⁹ **RDF triples** consist of two pieces of information (subject and object) and a linking fact (predicate).

143. Friend of a Friend Project homepage. <<http://www.foaf-project.org/>>.

144. Shannon, Victoria. “A ‘More Revolutionary’ Web.” *International Herald Tribune*. May 24 2006 <<http://www.iht.com/articles/2006/05/23/business/web.php>>.

145. Weitzner, D., J. Hendler, T. Berners-Lee, and D. Connolly. “Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web.” October 2004 <<http://www.w3.org/2004/09/Policy-Aware-Web-ac1.pdf>>.

146. “Bill Gates: Microsoft MIX06 Conference.” *Microsoft*, March 2006.

147. “Microformats Wiki.” *microformats.org* <http://microformats.org/wiki/Main_Page>.

Let's create a simple RDF triple. "Chapter 3, Dive Into® Web 2.0" is the title of this document and one property (the document's subject) that we'll use in our RDF triple. Another property of this chapter is "Deitel" as the author. So the sentence "Chapter 3, Dive Into® Web 2.0 is written by Deitel" is an RDF triple, containing two properties and a linking fact ("is written by").

[DBpedia.org](#) is currently transferring content into RDF from Wikipedia, one of the largest and most popular resources of online information. Using [SPARQL](#) (SPARQL Protocol and RDF Query Language), DBpedia.org is converting data from Wikipedia entries into RDF triples. In June 2007, they claimed to have over 91 million triples—this will allow the information (from Wikipedia) to be accessed by more advanced search queries.¹⁵⁰

Ontologies

Ontologies are ways of organizing and describing related items, and are used to represent semantics. This is another means of cataloging Internet content in a way that can be understood by computers.¹⁵¹ RDF is designed for formatting ontologies. [OWL \(Web Ontology Language\)](#), also designed for formatting ontologies in XML, extends beyond the basic semantics of RDF ontologies to enable even deeper machine understanding of content.¹⁵²

Closing Comment

This book will get you up to speed on Web 2.0 applications development. Building a "web of meaning" will ultimately open a floodgate of opportunities for web developers and entrepreneurs to write new applications, create new kinds of businesses, etc. We don't know exactly what the "web of meaning" will look like, but it's starting to take shape. If it helps accomplish what many leaders in the web community believe is possible, the future of the web will be exciting indeed.

3.19 Wrap-Up

In this chapter, you learned how Web 2.0 embraces an architecture of participation, encouraging user interaction and community contributions. User-generated content is the key to success for many leading Web 2.0 companies. Harnessing collective intelligence can result in smart ideas. Collaborative filtering lets users promote valuable content, and flag offensive or inappropriate material. The wisdom of crowds suggests that a large diverse group of people can be smarter than a small group of specialists.

We presented several popular Web 2.0 business models. You learned how you, the user, are deciding which news and information outlets you trust, enabling popular blogs and social media networks to compete with traditional media powerhouses. People are using social networks to interact and network, personally and professionally. We discussed

-
148. Miller, E. "An Introduction to the Resource Description Framework." *D-Lib Magazine*, May 1998 <<http://dlib.org/dlib/may98/miller/05miller.html>>.
 149. "Resource Description Framework (RDF) Concepts and Abstract Syntax." *w3.org* <<http://www.w3.org/TR/rdf-concepts/#section-Concepts>>.
 150. [DBpedia.org](#). <<http://dbpedia.org/docs/>>.
 151. Heflin, J. "OWL Web Ontology Language Use Cases and Requirements." *W3C*, 10 February 2004 <<http://www.w3.org/TR/webont-req/>>.
 152. "Introduction to OWL." *W3Schools* <http://www.w3schools.com/rdf/rdf_owl.asp>.

popular social bookmarking sites that let you share your favorite websites, blogs, and articles with other users.

You learned about the Long Tail economic model and how Web 2.0 Internet businesses are increasing exposure for lesser-known products in a way that traditional businesses cannot. Web 2.0 companies are monetizing their content with advertising, affiliate programs and more.

We discussed how the explosion of content combined with people's increasing demands on time has led to an attention economy, increasing the importance of search engines used to find content online. SEO, link building and SEM can help you maximize your website's findability and improve search engine results. Many Web 2.0 sites enable discovery, pointing you to valuable new content that you might not have otherwise sought. Tagging and folksonomies help you locate content on the web more effectively, especially content that computers have a hard time identifying, such as photos and videos. Search engines are using localization to provide you with geographically relevant content.

You learned how Software as a Service (SaaS) applications offer companies (and users) many benefits, including fewer demands on internal IT departments, increased accessibility for out-of-the-office use, an easy way to maintain software across a diversity of platforms on a large scale and more. Rich Internet Applications offer responsiveness, "rich" features and functionality similar to desktop applications. Web services are used to create feature-rich mashup applications, combining content or functionality from existing web services, websites and RSS feeds. Many people believe that the Semantic Web—the "web of meaning"—will be the next generation of the web, enabling exciting new kinds of applications.

This chapter concludes our introduction to computers, the Internet, browsers and Web 2.0. The remainder of the book is devoted to building web applications—you'll learn how to program the client side and the server side, including interacting with databases. We'll focus on building Ajax-enabled Rich Internet Applications. We begin in Chapter 4 by discussing how to use XHTML (the Extensible HyperText Markup Language) to create web pages to be rendered by web browsers. You'll use XHTML to incorporate images into your web pages, add internal linking for page navigation, create forms for collecting information from a user, create tables and more.

3.20 Where to Go for More Web 2.0 Information

Figure 3.4 lists some popular resources for Web 2.0 news and analysis.

Resource	Description
TechCrunch http://www.techcrunch.com/	Edited by Michael Arrington, this blog is the leading Web 2.0 news resource that profiles innovative and important Internet companies and products.
Mashable http://www.mashable.com/	A social networking news blog, edited by Pete Cashmore. The site includes sections devoted to MySpace, YouTube, Bebo, Facebook and Xanga.

Fig. 3.7 | Web 2.0 news, analysis, technology and business resources. (Part 1 of 2.)

Resource	Description
ReadWriteWeb http://www.readwriteweb.com/	Edited by Richard MacManus, this blog provides web technology news, reviews and analysis.
GigaOM http://www.gigaom.com/	Technology news and analysis blog, edited by Om Malik—founder of GigaOmniMedia and a former writer for Business 2.0, Red Herring and Forbes.com.
Dion Hinchcliffe's Web 2.0 Blog http://web2.socialcomputingmagazine.com/	Web 2.0 news and analysis blog by Dion Hinchcliffe, Editor-in-Chief of <i>Social Computing Magazine</i> .
Matt Cutts' Blog http://www.mattcutts.com/blog/	Matt Cutts, a software engineer at Google, blogs about gadgets, Google and SEO.
O'Reilly Radar http://radar.oreilly.com/	O'Reilly Media's blog about Web 2.0, open source, emerging technology and more.
SearchEngineLand http://www.searchengineland.com/	Search engine news blog, edited by Danny Sullivan—a leading search engine expert.
SearchEngineWatch http://searchenginewatch.com/	News and analysis of the search engine industry. Includes blogs, tutorials, forums and more.
Deitel Resource Centers http://www.deitel.com/resourcecenters.html (See Fig. 2 in the Preface for a list of Resource Centers.)	Numerous Web 2.0 technology and Internet business Resource Centers that include links to, and descriptions of tutorials, demos, free software tools, articles, e-books, whitepapers, videos, podcasts, blogs, RSS feeds and more.

Fig. 3.7 | Web 2.0 news, analysis, technology and business resources. (Part 2 of 2.)

3.21 Web 2.0 Bibliography

General Web 2.0

- Anderson, C. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.
- Anderson, C. “The Long Tail.” *WIRED*, October 2004 <<http://www.wired.com/wired/archive/12.10/tail.html>>.
- Battelle, J. *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Portfolio, 2005.
- “Bill Gates: Microsoft MIX06 Conference.” *Microsoft*, March 2006 <<http://www.microsoft.com/presspass/exec/billg/speeches/2006/03-20MIX.mspx>>.
- Brin, S. and L. Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine.” <<http://infolab.stanford.edu/~backrub/google.html>>.
- Farber, D. “ETech: Attenuation, Web 2.0 and spimes.” *ZDNet*, March 2006 <<http://blogs.zdnet.com/BTL/?p=2667>>.
- Graham, P. “Web 2.0.” November 2005 <<http://www.paulgraham.com/web20.html>>.

- Grossman, L. "TIME's Person of the Year: You." *TIME*, December 2006 <<http://www.time.com/time/magazine/article/0,9171,1569514,00.html>>.
- Hinchcliffe, D. "The State of Web 2.0." April 2006 <http://web2.socialcomputingmagazine.com/the_state_of_web_20.htm>.
- Horrigan, J. B. "Home Broadband Adoption 2006." *Pew Internet & American Life Project*, May 2006 <http://www.pewinternet.org/pdfs/PIP_Broadband_trends2006.pdf>.
- Madden, M. and S. Fox. "Riding the Waves of 'Web 2.0'." *Pew Internet & American Life Project*, October 2006 <http://www.pewinternet.org/pdfs/PIP_Web_2.0.pdf>.
- Miller, P. "Thinking About This Web 2.0 Thing." August 2005 <http://paulmiller.typepad.com/thinking_about_the_future/2005/08/thinking_about_.html>.
- Moore, G. "Cramming More Components onto Integrated Circuits." *Electronics*, April 1965 <http://ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf>.
- Musser, J. and T. O'Reilly. *Web 2.0 Principles and Best Practices*. O'Reilly Media, Inc., 2006.
- O'Reilly, T. "Web 2.0: Compact Definition?" October 2005 <http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html>.
- O'Reilly, T. "What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." September 2005 <<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>>.
- "Tim O'Reilly's seven principles of web 2.0 make a lot more sense if you change the order." *Open Gardens*, April 2006 <http://opengardensblog.futuretext.com/archives/2006/04/tim_o_reillys_s.html>.

Search

- "About Rollyo." *Rollyo* <<http://www.rollyo.com/about.html>>.
- "About Us." *Microsoft adCenter Labs* <<http://adlab.msn.com/AboutUs.aspx>>.
- Brin, S. and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." <<http://infolab.stanford.edu/~backrub/google.html>>.
- Calore, M. "Remembering the First Google Bomb." *Compiler (WIRED blog)*, 26 January 2007 <http://blog.wired.com/monkeybites/2007/01/earlier_today_m.html>.
- "Company History." *Yahoo!* <<http://yhoo.client.shareholder.com/press/history.cfm>>.
- "Company Overview." *Google* <<http://www.google.com/intl/en/corporate/index.html>>.
- "comScore Search Rankings." *CNN*, 16 July 2007 <<http://money.cnn.com/news/newsfeeds/articles/prnewswire/AQM17916072007-1.htm>>.
- Davenport, T. and J. Beck. *The Attention Economy: Understanding the New Currency of Business*. Harvard Business School Press, 2002.
- "Google: One Million and Counting." *Pandia Search Engine News*, 2 July 2007 <<http://www.pandia.com/sew/481-gartner.html>>.
- "Mahalo FAQ." *Mahalo* <http://www.mahalo.com/Mahalo_FAQ>.
- "Microsoft Digital Advertising Solutions." *Microsoft*. <<http://advertising.microsoft.com/advertising-online>>.
- Mills, E. "Google Rises at Yahoo's expense." *CNET*, 23 April 2007 <http://news.com.com/Google+rises+at+Yahoos+expense/2100-1038_3-6178164.html>.
- Morville, P. *Ambient Findability: What We Find Changes Who We Become*. O'Reilly Media Inc., 2005.

- “MSN Live Search: About Us.” *MSN* <<http://search.msn.com/docs/default.aspx?FORM=HLHP2>>.
- Olson, S. “MSN Launches Revamped Search Engine.” *CNET*, 30 June 2004 <http://news.com.com/MSN+launches+revamped+search+engine/2100-1032_3-5254083.html>.
- Search Engine Strategies: Conference & Expos*, 2007 <<http://www.searchenginestrategies.com>>.
- Sherman, C. “The State of Search Engine Marketing 2006.” *Search Engine Land*, 8 February 2007 <<http://searchengineland.com/070208-095009.php>>.
- Sullivan, D. “Congratulations! You’re a Search Engine Marketer!” *Search Engine Watch*, 5 November 2005 <<http://searchenginewatch.com/showPage.html?page=2164351>>.
- “Technology Overview.” *Google* <<http://www.google.com/corporate/tech.html>>.
- Thompson, C. “Media in the Age of the Swarm.” *cbc.ca* <http://www.cbc.ca/10th/columns/media_thompson.html>.
- Underwood, L. “A Brief History of Search Engines.” *Web Ref*. <http://www.webreference.com/authoring/search_history/>.
- “Vertical Search-Engines: Know Your Subject.” *Economist.com*, 12 July 2007 <http://www.economist.com/business/displaystory.cfm?story_id=9478224>.
- “Web Search: How to use MSN Search.” *MSN* <http://search.msn.com/docs/help.aspx?t=SEARCH_CONC_WhatsNewWithMSNSearch.htm#2D>.
- Wilding, R. “Top 5 Black Hat and White Hat Search Engines Optimisation Techniques.” *PushON* <<http://www.pushon.co.uk/articles/top5-black-hat-white-hat-seo.htm>>.

User-Generated Content

- Cauchi, S. “Online Encyclopedias Put to the Test.” *The Age*, December 2005 <<http://www.theage.com.au/news/national/online-encyclopedias-put-to-the-test/2005/12/14/1134500913345.html>>.
- “Leading Indicators.” *FORTUNE*, 13 November 2006, p.40.
- Maney, K. “Techies Hot on Concept of ‘Wisdom of Crowds,’ But It Has Some Pitfalls.” *USA Today*, September 2006 <http://www.usatoday.com/tech/columnist/kevinmaney/2006-09-12-wisdom-of-crowds_x.htm>.
- O'Reilly, T. “The Architecture of Participation.” June 2004 <<http://oreillynet.com/lpt/a/5994>>.
- Seigenthaler, J. “A False Wikipedia ‘Biography.’” *USA Today*, November 2005 <http://www.usatoday.com/news/opinion/editorials/2005-11-29-wikipedia-edit_x.htm>.
- “SocialText is the Enterprise Wiki Trusted Most by Global 2000 Corporations.” *SocialText* <<http://www.socialtext.com/products/overview>>.
- Steel, E. “Newspapers’ Ad Sales Show Accelerating Drop.” *The Wall Street Journal*, 18 July 2007, A4.
- Suworecki, J. *The Wisdom of Crowds*. Anchor, 2005.
- Tapscott, D. and A. D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover, 2006.
- Vara, V. “Wikis at Work.” *The Wall Street Journal*, 18 June 2007, R11.

Blogging

- Akismet* <<http://http://akismet.com/stats/>>.
- “Blog Statistics and Demographics.” *Caslon Analytics*, March 2007 <<http://www.caslon.com.au/weblogprofile1.htm>>.

- Blood, R. "Weblogs: A History and Perspective." *Rebecca's Pocket*, September 2000 <http://www.rebeccablood.net/essays/weblog_history.html>.
- Bowman, S. and C. Willis. "We Media." July 2003 <http://www.hypergene.net/wemedia/download/we_media.pdf>.
- "CEO Blog List." *NewPR Wiki* <<http://www.thenewpr.com/wiki/pmwiki.php/Resources/CEOBlogsList?pagename=Resources.CEOBlogsList>>.
- Kirkpatrick, D. "Why There's No Escaping the Blog." *FORTUNE*, 10 January 2005 <http://money.cnn.com/magazines/fortune/fortune_archive/2005/01/10/8230982/index.htm>.
- Lenhart, A. and S. Fox. "Bloggers: A Portrait of the Internet's New Storytellers." *Pew Internet & American Life Project*, July 2006 <<http://www.pewinternet.org/pdfs/PIP%20Bloggers%20Report%20July%2019%202006.pdf>>.
- Thompson, C. "A Timeline of the History of Blogging." *New York Magazine*, February 2006 <<http://nymag.com/news/media/15971/>>.
- Walsh, B. *Clear Blogging: How People Are Changing the World and How You Can Join Them*. APress, 2007.

Social Networking

- Arrington, M. "Bear Stearns: Yahoo Must Form A Social Networking Strategy." *TechCrunch*, 3 August 2007 <<http://www.techcrunch.com/2007/08/03/bear-stearns-yahoo-must-form-a-social-networking-strategy>>.
- Athavely, A. "A Job Interview You Don't Have to Show Up For." *The Wall Street Journal Online*, June 2007 <http://online.wsj.com/article/SB118229876637841321.html?mod=tff_main_tff_top>.
- Baker, S. "IBM on Second Life: More than PR." *BusinessWeek*, 15 November 2006 <http://www.businessweek.com/the_thread/blogspotting/archives/2006/11/ibm_on_second_1.html>.
- Bulkeley, W. "Playing Well With Others." *The Wall Street Journal Online*, 18 June 2007 <<http://online.wsj.com/article/SB118194536298737221.html>>.
- "Businesses Find MySpace is a Place to Reach Youth." *Trend Watching*, July 2006 <http://www.trendwatching.com/about/inmedia/articles/youniversal_branding/businesses_find_myplace_is_a_p.html>.
- Copeland, M. "The Missing Link." *CNNMoney.com*, February 2007 <http://money.cnn.com/magazines/business2/business2_archive/2006/12/01/8394967/index.htm>.
- "Customer Support." *Facebook* <<http://www.facebook.com/help.php?tab=safety>>.
- "Facebook Statistics." *Facebook*, 17 July 2007 <http://static.ak.facebook.com/press/facebook_statistics.pdf?11:44617>.
- Jesdanun, A. "Candidates Seek Youths at MySpace." *ABC News*, August 2006 <<http://abcnews.go.com/Technology/wireStory?id=2325325&page=1>>.
- Kirkpatrick, M. "Friendster Awarded Patent on Social Networking." *TechCrunch*, July 2006 <<http://www.techcrunch.com/2006/07/07/friendster-awarded-patent-on-social-networking/>>.
- Lawson, S. "Second Life Creates a Millionaire." *IDG News*, 30 November 2006 <<http://www.itworld.com/App/4201/061130secondlife/>>.
- "Metcalf's Law." <<http://www-ec.njit.edu/~robertso/infosci/metcalf.html>>.
- Prescott, L. "Hitwise US Consumer Generated Media Report." *Hitwise*, February 2007.

- Rivlin, G. "Wallflower at the Web Party." *New York Times*, October 2006 <<http://www.nytimes.com/2006/10/15/business/yourmoney/15friend.html>>.
- "Second Life Lawyer." *Business 2.0*, May 2007, p.86.
- "The CEO's Guide to Social Networks." *BusinessWeek*, September 2006 <http://businessweek.com/mediacenter/qt/podcasts/guide_to_tech/guidetotech_09_11_06.mp3>.
- "Top 20 Websites." *Hitwise*, May 2007 <<http://hitwise.com/datacenter/rankings.php>>.
- "What is Second Life." <<http://secondlife.com/whatis>>.

Social Media

- "Amazon.com to Launch DRM-Free MP3 Music Download Store with Songs and Albums from EMI Music and More Than 12,000 Other Labels." May 2007 <<http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1003003>>.
- Colvin, G. "TV Is Dying? Long Live TV!" *FORTUNE*, 5 February 2007, p.43.
- "Conference Report Filed in House." *Library of Congress*, 8 October 1998 <<http://thomas.loc.gov/cgi-bin/bdquery/z?d105:HR02281:@@D&summ2=m&>>.
- D'Agostino, D. "Security in the World of Web 2.0." *Innovations*, Winter 2006, p.15.
- Hefflinger, M. "Digg Users Revolt Over Deleted Posts of HD DVD Security Hack." *digitalmediawire*, 2 May 2007 <<http://www.dmwmedia.com/news/2007/05/02/digg-users-revolt-over-deleted-posts-of-hd-dvd-security-hack>>.
- Jobs, S. "Thoughts on Music." February 2007 <http://www.apple.com/hotnews/thoughtson_music>.
- "Leading Indicators." *FORTUNE*, 13 November 2006, p.40.
- Maney, K. "Techies Hot on Concept of 'Wisdom of Crowds,' but it has some pitfalls." *USA Today*, September 2006 <http://www.usatoday.com/tech/columnist/kevinmaney/2006-09-12-wisdom-of-crowds_x.htm>.
- Mills, E. "Copyright Quagmire for Google and YouTube." *ZDNet*, March 2007 <http://news.zdnet.com/2100-9588_22-6167281.html>.
- O'Hear, S. "Viacom to Partner with Joost." *ZDNet*, 20 February 2007 <<http://blogs.zdnet.com/social/?p=96>>.
- Prescott, L. "Hitwise US Consumer Generated Media Report." *Hitwise*, February 2007.
- Steel, E. "Newspapers' Ad Sales Show Accelerating Drop." *The Wall Street Journal*, 18 July 2007, A4.
- Smith E. and K. Delaney. "Vivendi's Universal Sues Two Web Sites Over Video Sharing." *The Wall Street Journal*, 18 October 2006, B3.
- Torrone, P. "What Is Podcasting?" *O'Reilly Media*, 20 July 2005 <<http://digitalmedia.oreilly.com/2005/07/20/WhatIsPodcasting.html>>.

Tagging

- Chun, S., R. Cherry, D. Hiwiller, J. Trant and B. Wyman. "Steve.museum: An Ongoing Experiment in Social Tagging, Folksonomy, and Museums." *Archives & Museum Informatics*, March 2006 <<http://www.archimuse.com/mw2006/papers/wyman/wyman.html>>.
- Garrett, J. J. "An Interview with Flickr's Eric Costello." *Adaptive Path*, August 2005 <<http://www.adaptivepath.com/publications/essays/archives/000519.php>>.
- Masters, C. "Programming Provocateurs." *TIME*, 19 March 2007, p. 84.
- Mayfield, R. "Buy Side Publishing." *Ross Mayfield's Weblog*, April 2006 <http://ross.typepad.com/blog/2006/04/buy_side_publis.html>.

- Rainie, L. "Tagging." *Pew Internet & American Life Project*, December 2006 <http://www.pewinternet.org/pdfs/PIP_Tagging.pdf>.
- Schonfeld, E. "The Flickrization of Yahoo!" *CNN Money.com*, December 2005 <http://money.cnn.com/magazines/business2/business2_archive/2005/12/01/8364623/>.
- "Using Technorati Tags." *Technorati* <<http://support.technorati.com/support/siteguide/tags>>.
- Vander Wal, T. "Folksonomy Coinage and Definition." *Vanderwal.net*, February 2007 <<http://www.vanderwal.net/folksonomy.html>>.

Social Bookmarking

- "About Ma.gnolia" <<http://ma.gnolia.com/about>>.
- "Gardeners." *Ma.gnolia Community Wiki*, March 2007 <<http://wiki.ma.gnolia.com/Gardeners>>.
- "Know How Adobe and del.icio.us Work Together?" *del.icio.us blog*, May 2007 <http://blog.del.icio.us/blog/2007/05/knowhow_adobe_a.html>.
- "That was Fast." *del.icio.us blog*, March 2007 <http://blog.del.icio.us/blog/2007/03/that_was_fast.html>.

Software Development

- 37Signals. *Getting Real*, 2006 <<http://gettingreal.37signals.com>>.
- Fallows, D. "China's Online Population Explosion." *Pew Internet & American Life Project*, July 2007 <http://www.pewinternet.org/pdfs/China_Internet_July_2007.pdf>.
- Farber, D. "ETech: Attenuation, Web 2.0 and Spimes." *ZDNet*, March 2006 <<http://blogs.zdnet.com/BTL/?p=2667>>.
- Greenemeier, L. "Open-Source Exuberance." *InformationWeek*, July 2005 <<http://www.informationweek.com/story/showArticle.jhtml?articleID=165700923>>.
- Peiris, M. "The Pros and Cons of Hosted Software." *SmartBiz*, March 2006 <<http://www.smartbiz.com/article/articleview/1118/1/42>>.
- Pink, D. H. "Why the World Is Flat." *Wired*, May 2005 <<http://www.wired.com/wired/archive/13.05/friedman.html>>.

Tapscott, D. and A. D. Williams. *Wikinomics: How Mass Collaboration Changes Everything*, Portfolio Hardcover, 2006.

"The RIA Desktop in a Browser." *LaszloSystems* <<http://www.laszlosystems.com/software/webtop>>.

Tiemann, M. "Licenses by Name." *Open Source Initiative*, 18 September 2006 <<http://www.opensource.org/licenses/alphabetical>>.

Van Rossum, G. "Open Source Summit Trip Report." *Linux Gazette*, 10 April 1998 <<http://linuxgazette.net/issue28/rossum.html>>.

Wilcox, J. "IBM to Spend \$1 Billion on Linux in 2001." *CNET*, January 2002 <http://news.com.com/IBM+to+spend+1+billion+on+Linux+in+2001/2100-1001_3-249750.html>.

Rich Internet Applications

- "Adobe Flex 2." *Adobe* <http://www.adobe.com/products/flex/whitepapers/pdfs/flex2wp_technicaloverview.pdf>.
- Berlind, D. "Google Gears Vies to be De Facto Tech for Offline Web Apps." *ZDNet*, 31 May 2007 <<http://blogs.zdnet.com/Berlind/?p=504>>.
- "Core Effects." *Script.aculo.us Wiki* <http://wiki.script.aculo.us/scriptaculous/show/Core_Effects>.

- Cubrilovic, N. "Silverlight: The Web Just Got Richer." *TechCrunch*, April 2007 <<http://www.techcrunch.com/2007/04/30/silverlight-the-web-just-got-richer>>.
- Mills, E. "Google Gears Churns Toward Microsoft." *CNET*, 31 May 2007 <http://news.com.com/2100-1012_3-6187942.html>.
- "Prototype Tips and Tutorials." *Prototype JavaScript* <<http://prototypejs.org/learn>>.
- "Sun Radically Simplifies Content Authoring—Previews JavaFX Script." *Sun Microsystems*, 8 May 2007 <<http://www.sun.com/aboutsun/pr/2007-05/sunflash.20070508.2.xml>>.
- Taft, E. "Google Gears Allows Offline Web Development." *eWeek*, 30 May 2007 <<http://www.eweek.com/article2/0,1895,2139083,00.asp>>.
- "The 70 Percent Solution." *Business 2.0*, 28 November 2005 <http://money.cnn.com/2005/11/28/news/newsmakers/schmidt_biz20_1205/>.
- "The Dojo Offline Toolkit." *The Dojo Toolkit* <<http://dojotoolkit.org/offline>>.

Web Services and Mashups

- Amazon Web Services* <<http://aws.amazon.com>>.
- Braiker, B. "Tech: Welcome, Year of the Widget." *Newsweek*, December 2006 <<http://www.msnbc.msn.com/id/16329739/site/newsweek/>>.
- Costello, R. "REST Tutorial." *xFront*, June 2002 <<http://www.xfront.com/REST.html>>.
- Fielding, R. T. "Architectural Styles and the Design of Network-Based Software Architectures." <<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>.
- Richardson, L. and S. Ruby. *RESTful Web Services*. O'Reilly, 2007.
- Roush, W. "Second Earth." *Technology Review*, July/August 2007, p.38.

Location-Based Services

- Magid, L. "Global Positioning by Cellphone." *New York Times*, 19 July 2007, C7.
- Malykhina, E. "Nokia Wants Your Cell Phone To Tell You Where You Are." *InformationWeek*, 9 October 2006 <<http://www.informationweek.com/showArticle.jhtml?articleID=193105219>>.
- Schiller, J. and A. Voisard. *Location-Based Services*. Morgan Kaufmann, 2004.

XML, RSS, Atom, JSON and VoIP

- Broache, A. "eBay to Nab Skype for \$2.6 billion." *CNET*, 12 September 2005 <http://news.com.com/eBay+to+nab+Skype+for+2.6+billion/2100-1030_3-5860055.html>.
- "Introducing JSON" <<http://www.json.org>>.

Internet Business

- Anderson, C. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.
- Copeland, M. "The Missing Link." *CNNMoney.com*, February 2007 <http://money.cnn.com/magazines/business2/business2_archive/2006/12/01/8394967/index.htm>.
- Deutschman, A. "Inside the Mind of Jeff Bezos." *Fast Company*, August 2004 <http://www.fastcompany.com/magazine/85/bezos_1.html>.
- Graham, J. "The House that Helped Build Google." *USA TODAY*, 5 July 2007, 1B.
- Helft, M. "eBay Spawns a Marketplace for the Bizarre." *CNN*, 20 September 1999 <<http://www.cnn.com/TECH/computing/9909/20/ebay.side.show.idg/>>.
- Newman, A. A. "Google to Expand Program, Giving Newspapers a Lift." *New York Times*, 18 July 2007, C8.

O'Connell, P. "Pierre Omidyar on 'Connecting People.'" *BusinessWeek*, 20 June 2005 <http://www.businessweek.com/magazine/content/05_25/b3938900.htm>.
Second Life <<http://www.secondlife.com/whatis/economy.php>>.

Semantic Web

Berners-Lee, T. *Weaving the Web*. Harper-Collins, 2000.

"Bill Gates: Microsoft MIX06 Conference." *Microsoft*, March 2006 <<http://www.microsoft.com/presspass/exec/billg/speeches/2006/03-20MIX.mspx>>.

Copeland, M. V. "Weaving The [Semantic] Web." *Business 2.0*, July 2007, p.89-93.

Heflin, J. "OWL Web Ontology Language Use Cases and Requirements." *W3C*, 10 February 2004 <<http://www.w3.org/TR/webont-req/>>.

"Introduction to OWL." *W3Schools* <<http://www.w3schools.com/rdf/rdf.owl.asp>>.

"Microformats FAQs for RDF fans." *Microformat.org Wiki* <<http://microformats.org/wiki/faqs-for-RDF>>.

"Microformats Wiki." *microformats.org* <http://microformats.org/wiki/Main_Page>.

Miller, E. "An Introduction to the Resource Description Framework." *D-Lib Magazine*, May 1998 <<http://dlib.org/dlib/may98/miller/05miller.html>>.

Rapoza, J. "Semantic Web Technology Gains Steam." *eWEEK*, June 2007 <http://etech.eweek.com/content/web_technology/semantic_web_technology_gains_steam.html>.

Rapoza, J. "SPARQL Will Make the Web Shine." *eWEEK*, May 2006 <<http://www.eweek.com/article2/0,1759,1965980,00.asp>>.

Rapoza, J. "Weaving the Semantic Web." *eWEEK*, May 2007 <http://etech.eweek.com/content/web_technology/weaving_the_semantic_web.html>.

Tauberer, J. "What Is RDF?" *xml.com*, July 2006 <<http://www.xml.com/pub/a/2001/01/24/rdf.html>>.

Weitzner, D., J. Hendl, T. Berners-Lee, and D. Connolly. "Creating a Policy-Aware Web: Discretionary, Rule-based Access for the World Wide Web." October 2004 <<http://www.w3.org/2004/09/Policy-Aware-Web-acl.pdf>>.

3.22 Web 2.0 Glossary

Adaptive Path—A strategy and design consulting company that helps companies build products that improve the users' experiences. Founder Jesse James Garrett coined the term Ajax and is a major proponent of the technology.

Adobe Integrated Runtime (AIR)—Allows offline access to web applications (i.e., when an Internet connection is not available).

AdSense—Google's search advertising program for web publishers. This is a fundamental and popular form of monetization, particularly for Web 2.0 startup companies.

AdWords—Google's search advertising program for advertisers.

affiliate network—A company (see Commission Junction and LinkShare) that connects web publishers with cost-per-action affiliate programs. (See affiliate program.)

affiliate program—A program that allows publishers to post text and image ads on their sites. If a user clicks through to the affiliate site and takes a specified action (e.g., makes a purchase, fills out a registration form, etc.) the publisher is paid a portion of the sale or a flat fee.

agile software development—A process that focuses on developing small pieces and upgrades to a program quickly and continuously throughout the life cycle of the product.

Amazon—An online retailer and web services provider. The Amazon Associates affiliate program allows web publishers to monetize their sites by recommending Amazon products.

Apache—An open source software foundation, responsible for the Apache Web Server and many other open source products.

API (Application Programming Interface)—An interface called by a program. Web services are often referred to as APIs and are accessible over the Internet.

Apple—A leading computer company, responsible for Macintosh computers, the iPod, iTunes and the iPhone.

architecture of participation—A design that encourages user interaction, where the community contributes content and/or participates in the design and development process. Creating websites that encourage participation is a key part of Web 2.0.

Ajax (Asynchronous JavaScript and XML)—Allows pieces of a web page to be refreshed separately, while the user continues to work on the page. Ajax improves the user's experience by making webtop applications approach the responsiveness of desktop applications.

Atom—Used for web feeds; similar to RSS.

attention economy—The result of the abundant amounts of information being produced and people's limited free time. More content is available than users can sort through on their own.

avatar—A person's digital representation in a 3D world such as Second Life.

Basecamp—An SaaS project management and collaboration tool from 37Signals.

Blinkx—A video search engine with over 12 million hours of video indexed (which makes the content searchable).

blog—A website consisting of posts in reverse chronological order. For common blog components see: blogroll, permalink, reader comment, and trackback.

blog search engine—A search engine devoted to the blogosphere. Some of the top blog search engines include Technorati, Feedster, IceRocket and Google Blog Search.

Blogger—A popular blogging platform now owned by Google.

blogger—The author of a blog.

blogging—The act of writing a blog.

blog network—A collection of blogs with multiple editors. Popular blog networks include Corante, 9rules, Gawker and Weblogs, Inc.

blogosphere—The blogging community. In mid 2007 there were over 90 million blogs.

blogroll—A list of links to a blogger's favorite blogs.

broadband Internet—High-speed Internet, often offered by cable companies and satellite companies.

collaborative filtering—The act of working together to promote valuable content and remove spam or offensive content.

collective intelligence—The idea that collaboration and competition among large groups results in grand and intelligent ideas.

Commission Junction—A popular affiliate network with member advertisers including eBay, Best Buy, Hewlett-Packard and hundreds more.

community-generated content—Content (writing, videos, etc.) that is created by Internet users.

contextual advertising—Advertising that is targeted to web page content. Because these ads are relevant to the nearby content, contextual ads often enhance the value of that content and generate higher clickthrough rates.

Corante—A blog network whose blogs are written by leading commentators in their field. Categories include law, policy, business, management, media, the Internet, technology and science.

cost-per-action (CPA)—Advertising that is billed by user action (e.g., purchasing a product or filling out a form).

cost-per-click (CPC)—Advertising that is billed by user click. The publisher receives revenue each time the user clicks an ad on the publisher's site, regardless of whether the user makes a subsequent purchase.

cost-per-thousand impressions (CPM)—Advertising that is billed per thousand impressions, regardless of whether the user clicks the ad or makes a subsequent purchase.

Craigslist—A popular classifieds and social networking website that fits the Web 2.0 lightweight business model. The company has few employees and all of the content is user generated. Craigslist was originally free; however, it is now transitioning to charging for certain services such as real-estate and job postings. A portion of the company is owned by eBay.

DBpedia.org—A website working on converting Wikipedia articles into RDF triples. This is a step toward the Semantic Web.

Deitel—A content-creation, corporate training and Web 2.0 business development organization. Deitel has a rapidly growing content network (currently about 80 Resource Centers) specializing in topic categories, including Web 2.0, Internet business, programming languages, software development and more.

del.icio.us—A social bookmarking site.

Digg—A social media site where users submit news stories and the community votes on the stories. The most popular stories are featured on the site's front page.

Digital Rights Management (DRM)—Technology used to prevent piracy and misuse of digital media. Several high-profile executives, including Apple CEO Steve Jobs, have recently started an anti-DRM movement.

discovery—The future of search; the idea of introducing users to valuable content they might not have looked for otherwise. For example, social bookmarking sites and Deitel Resource Centers suggest valuable resources.

Dodgeball.com—A social networking site designed for use on mobile devices, owned by Google. Allows users to locate friends and “crushes” who are nearby so they can meet up.

DotNetNuke—An open source web application framework based on Microsoft's .NET framework. DotNetNuke allows users to build dynamic websites quickly and easily. For more information, visit the DotNetNuke Resource Center (<http://www.deitel.com/ResourceCenters/Software/DotNetNukeDNN/tabcid/1217/Default.aspx>).

DoubleClick—An Internet advertising company acquired by Google in 2007 for \$3.1 billion. Their advertising exchange connects advertisers with online publishers.

Dougherty, Dale—Coined the term “Web 2.0.”

eBay—The leading online auction site.

Extensible Markup Language (XML)—A widely supported open (i.e., nonproprietary) technology for electronic data exchange and storage, which is fundamental to Web 2.0 and the Semantic Web. It can be used to create other markup languages to describe data in a structured manner.

Facebook—A social networking site. Though it is now open to the public, Facebook was originally designed for and is especially popular with college students.

Federated Media—A company that connects bloggers with advertisers. Founded by John Battelle, the chair for the annual Web 2.0 Summit Conference, and the author of *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*.

Feedburner—Provides RSS feeds for blogs, feed monetization, podcast tracking and more. Acquired by Google.

Firefox—Open source web browser (based on the Netscape Navigator browser introduced in 1994) developed by the Mozilla Foundation. For more information, visit the Firefox Resource Center (<http://www.deitel.com/ResourceCenters/Software/Firefox/tabid/1213/Default.aspx>).

Flex—A Rich Internet Application (RIA) framework developed by Adobe. For more information, visit the Flex Resource Center (<http://www.deitel.com/ResourceCenters/Programming/AdobeFlex/tabid/1682/Default.aspx>).

Flickr—A photo-sharing website often credited as one of the best examples of tagging content.

folksonomy—A classification based on tagging content. Users tag the web content (web pages, photos, etc.), making it easier to find the content online. Folksonomies are formed on sites such as Flickr, Technorati and del.icio.us. Users can search tags for content that is identified in different (and sometimes more meaningful) ways than by traditional search engines.

Friendster—A social networking site; an early leader in the category of social networking. In 2006, Friendster was awarded a patent for a method and tool called the “Web of Friends,” which gathers descriptive and relationship information for users in the network. The combined data is used to show all of the social relationships connecting two users in the social network. It also allows users to find people connected to their friends, their friends' friends, etc.

Garrett, Jesse James—Coined the term “Ajax” and founded Adaptive Path.

Gawker Media—A blog network that includes Gawker.com (New York City gossip), Gizmodo.com (technology and gadgets) and more.

Google—A Web 2.0 search and online advertising company founded by Larry Page and Sergey Brin when they were Ph.D. students at Stanford University. It is the most widely used search engine, commanding almost 50% market share. In addition to its regular search engine, Google offers specialty search engines for images, news, videos, blogs and more. Google provides web services that allow you to build Google Maps and other Google services into your applications.

Google Gears—An open source web browser extension that enables developers to provide offline usage of their web applications. The program can easily be resynchronized when an Internet connection becomes available.

Google Maps—Google’s mapping web service, hugely popular in mashups.

Hitwise—An Internet competitive intelligence service provider. Hitwise collects and sells usage information from over one million websites in numerous industries. Clients use the information to find market share, keyword, web traffic flow and demographic data.

housingmaps.com—A mashup of Google Maps and Craigslist apartment and real-estate listings; often credited as being the first mashup.

IceRocket—A blog search engine.

Intel—The computer hardware company that creates the processors that power most of the world’s computers.

interstitial ad—Ad that plays between page loads.

in-text contextual advertising—Advertising that is marked by double-underlined keywords in the content of a web page. When a reader hovers the mouse over a double-underlined word, a text

ad pops up. By clicking on the ad, the reader is taken to the advertiser's page. Companies providing in-text contextual advertising include Vibrant Media, Kontera, Text Link Ads and Tribal Fusion.

iPhone—Apple's mobile phone, released June 2007. The iPhone is designed to run a full version of the Internet.

iPod—Apple's portable media player.

iTunes—Apple's online music and video store; designed to sync with the iPod.

John Battelle's Searchblog—A blog in which John Battelle discusses search, media, technology, and more. (See Federated Media.)

JSON (JavaScript Object Notation)—A text-based data interchange format used to represent data structures and objects, and transmit them over a network. JSON is most often used with JavaScript in Ajax applications, but it can also be used with other programming languages.

Joost—An Internet TV company using semantic technologies to provide high-quality video with time shifting (recording for later viewing) and social networking capabilities. Joost allows advertisers to target their markets precisely. Advertisers can use demographic information such as location, gender, age, and more, to serve appropriate ads and to get a better return on investment from their advertising campaigns.

Last.fm—A popular social music website that uses the wisdom of crowds to recommend music to its users. The Last.fm Audioscrobbler music engine automatically sends the name of every song the user plays to the server. It then uses the information to recommend songs and artists, connect users with similar tastes and more.

Laszlo Webtop—A desktop-like environment for running web applications built on the OpenLaszlo framework.

lead generation—A monetization model for many sites that send traffic to another site and typically collect a fee when the visitor fills out an inquiry form so a salesperson can follow through and potentially convert the lead into a sale.

lightweight business model—A plan that allows a company to start quickly and with little capital investment. This is facilitated by powerful yet economical computers, the wide availability of inexpensive broadband Internet, robust open source software offerings, and well-developed, easy-to-employ monetization models—especially advertising and affiliate programs.

link baiting—Attracting other sites to link to your site, but without specifically asking for links. Providing quality content is considered the best form of link baiting.

link building—Using various methods to encourage other sites to link to your site. It is widely believed that increasing the number of inbound links to your site will encourage search engines to send you more traffic.

LinkedIn—A social networking site for business professionals. It can be used to stay in touch with professional contacts or make new contacts in your extended network.

LinkShare—A popular affiliate network with over 600 member companies (including American Express, Office Depot and Walmart).

Livejournal—A website where you can create your own blog.

Long Tail—Coined by Chris Anderson in an article in the October 2004 *WIRED* magazine (<http://www.wired.com/wired/archive/12.10/tail.html>). Refers to the continuous sales of numerous items with low sales volume that can add up to a significant part of a company's total sales. Amazon and Netflix are classic Long Tail companies.

mashup—A combination of two or more existing web services, RSS feeds or other sources of data to create a new application. For example, <http://www.housingmaps.com> combines real estate

listings from Craigslist with Google Maps so you can view the listings on a map. For a list of popular mashups, see <http://www.programmableweb.com/popular>.

Mechanical Turk—Amazon’s “artificial artificial intelligence,” which uses people in a web service to perform tasks that are difficult for computers to perform, such as identifying the subject of a picture and transcribing dictation recordings. Users can post a HIT (Human Intelligence Task). Anyone interested in completing the task can submit a response. If the response is approved by the person who posted the HIT, the responder is paid a predetermined fee for each task completed. The key is that the human task is interwoven with the execution of the web service, creating applications that mix computing power with human intelligence accessed worldwide.

MediaWiki—Open source software written originally for Wikipedia and now used by many popular wikis.

Metcalfe's Law—The value of a network is proportional to the square of the number of its users.

Metcalfe's Law was authored by Robert Metcalfe, the creator of Ethernet. (See also network effects.)

microformat—A common standard for identifying information in a way that can be understood by computers. Some current microformats include adr (for address information), hresume (for resumes and CVs), and xfolk (for collections of bookmarks). See <http://microformats.org> for more information.

mobile technology—Devices such as cell phones and PDAs. An increasing number now offer web access, which has opened up new web application possibilities. (See also iPhone.)

moblogging—Blogging from mobile devices.

moderation—Monitoring and censoring inappropriate content and comments in blog or forum postings. The potential need for moderation is a drawback to allowing user-generated content.

monetization—Generating money through your website (e.g., using contextual advertising, affiliate programs, e-commerce and other revenue-generating models).

Moonlight—An open source version of Microsoft's Silverlight for Linux operating systems.

Movable Type—A blogging software package from the company Six Apart that is installed on the blogger's server.

Mozilla Foundation—Creates and maintains open source software including the Mozilla Firefox web browser and the Mozilla Thunderbird e-mail client.

MySpace—The most popular social networking site, and the most popular site on the Internet.

network effects—The increased value of a network as its number of users grows. For example, as the number of people with Internet connections grows worldwide, the value and benefit to all users of the Internet grows (individuals can communicate with more people, companies can reach more customers, etc.). (See Metcalfe's Law.)

9Rules—A blog network.

ontology—A way of organizing and relating things. Ontologies are a key technology in the Semantic Web.

open source software—Software that is available for anyone to use and modify with few or no restrictions. Users can modify source code to meet their unique needs, or collaborate with others to enhance the software. Many Web 2.0 companies use open source software to power their sites, and offer open source products and content.

O'Reilly Media—The company that introduced and promoted the term Web 2.0 (coined by company executive Dale Dougherty). O'Reilly Media publishes technology books and

websites, and hosts several conferences, including the Web 2.0 Summit, Web 2.0 Expo, OSCON™ (the Open Source Convention), Emerging Technology, Emerging Telephony, Where 2.0, RailsConf, MySQL, Ubuntu Live and more. See the O'Reilly Radar (<http://radar.oreilly.com/>) to keep up-to-date on emerging technology trends.

outsourcing—A company's hiring of independent contractors or other companies to perform various tasks. Outsourcing is often cheaper for the company.

performance-based advertising—Advertising that pays based on user actions, such as making a purchase, filling out a registration form, etc. (See also cost-per-action.)

permalink—A URL that links to a specific blog entry instead of the blog's homepage. Links stay relevant even after the blog entry moves off the home page and into the archive.

perpetual beta—The idea of continually releasing new features even if they aren't "final." This allows software companies to constantly fix bugs and improve the software by getting continuous feedback from real users.

Pew Internet & American Life Project—A non-profit Internet research company. The project is funded by the Pew Charitable Trusts, which has also initiated other research and cultural projects.

PHPNuke—An open source content-management system and web publishing tool based on PHP and MySQL.

podcast—A media file designed for syndicated distribution online. It can be played on a personal computer or mobile media player (such as an iPod or MP3 player).

Policy Aware Web Project—A site devoted to developing policies regarding Internet data. This is an attempt to deal with Semantic Web security concerns.

premium content—Website content that is available for a fee (e.g., e-books, articles, etc.). It is a way for publishers to monetize their sites. Sites offering premium content typically offer free content as well.

Prologger—A blog about blogging. It teaches bloggers how to monetize their sites with Google AdSense and other programs.

Programmable Web—A website with extensive directories of web services APIs and mashups.

publisher—See "web publisher."

RDF (Resource Description Framework)—An XML-based language used to describe content attributes such as the page title, author, etc.

RDF triples—Composed of two pieces of information and a linking fact. They are used to help computers understand data, a vital part of the Semantic Web.

reader comment—Feedback left by readers on a blog.

recommender systems—Systems that collect data using collaborative filtering to determine users' tastes and interests as they search the Internet. For example, Amazon's "customers who bought this item also bought..."

Red Hat—A popular version of the Linux operating system. The company is a leader in the open source movement.

remixing—Combining existing applications and/or content into something new; this is fundamental to Web 2.0.

reputation systems—Systems used by businesses like eBay and Amazon to encourage trust. For example, after each eBay transaction, the buyer and the seller can each leave positive or negative comments about the other party.

REST (Representational State Transfer)—A simple alternative to SOAP for implementing web services. Many developers prefer REST-based web services to SOAP-based web services for their simplicity, their ability to be cached and more. Amazon offers some REST-based web services. (See also SOAP.)

Rich Internet Applications (RIAs)—Web applications that have the responsiveness and the rich GUI normally associated with desktop applications. Related technologies for building RIAs include Ajax, Dojo, Silverlight, Flex and more.

RSS—An XML-based web-content syndication format. Syndicated RSS feeds are used to publish frequently updated content such as news, blog entries, podcasts, and more. Some RSS feeds include the full text, but most contain only a portion of the document, encouraging the reader to visit the content site.

Ruby on Rails—An open source, web application development scripting language and framework that increases the speed at which you can create typical database-driven web applications.

Salesforce.com—An SaaS company that specializes in Customer Relationship Management (CRM) software; a leader in the SaaS movement.

scrobbling—Last.fm’s music tracking and analysis feature that provides you with recommendations based on the music you listen to through the site or on your iPod. (See also recommender systems.)

search engine marketing (SEM)—Promoting your website to increase traffic and search results. This includes paid search, online advertising and more.

search engine optimization (SEO)—Designing your website to maximize your findability and improve your rankings in organic search engine results.

search engine result page (SERP)—The page shown to a user by a search engine with a listing of web pages matching the search query sorted by relevance.

SearchEngineLand.com—Danny Sullivan’s search engine news blog.

SearchEngineWatch.com—A search engine marketing resource site. Includes articles, tutorials, conferences and more.

Second Life—A 3D virtual world social networking program developed by Linden Labs. Users create an avatar (their online image and persona) that they use to meet other users with similar interests, conduct business, participate in group activities, take classes and more.

Semantic Web—The “web of meaning.” What some believe will be the next evolution of the web in which web content can be read and understood by software applications.

Silverlight—A Rich Internet Application (RIA) framework developed by Microsoft; competes with Adobe Flash and Flex.

Six Apart—The company responsible for several blogging software applications and websites, including Movable Type, TypePad and Vox.

Skype—The most popular VoIP company. Users can place free calls to other Skype users around the world over their Internet connection. They also offer fee-based services that allow you to call non-Skype phone numbers. Skype was purchased by eBay in 2005 for \$2.6 billion. Its founders recently launched Joost (an Internet TV site).

SOAP (Simple Object Access Protocol)—A protocol for exchanging XML-based information over a network. SOAP is used as a messaging framework in web services.

social bookmarking—The act of sharing your bookmarks with others through a website such as del.icio.us or Ma.gnolia. Users bookmark their favorites sites, articles, blogs and more, and tag them by keyword.

social media—Any media (e.g., photos, videos, music, etc.) shared online. Social media sites, such as Digg, YouTube and Flickr, often include features for user comments, collaborative filtering and tagging.

social networking—Sites designed to organize users' existing relationships and help users establish new ones. Popular social networking sites include MySpace, Facebook, LinkedIn, Second Life and more.

SocialText—The first wiki company; provides wiki services to corporations. (See also wiki.)

Software as a Service (SaaS)—Software that runs on a web server. It does not need to be installed on your local computer, and companies can instantly update all users to the latest version. Salesforce.com, Google, 37Signals and Microsoft all have extensive SaaS offerings.

spam—Junk e-mail messages, blog comments and forum postings.

SPARQL Protocol and RDF Query Language (SPARQL)—An RDF query language for the Semantic Web.

tag—An author- and/or user-submitted label for web content used to classify it by subject or keyword. For example, a picture of the Statue of Liberty posted on Flickr might be tagged with “newyorkcity,” “statueofliberty,” “usa,” etc. Users can search for content on a site by tags. For examples of tag usage, see Technorati and Flickr.

tag cloud—A weighted list of content tags on a website. A tag cloud is usually in alphabetical order, with the most popular tags often appearing in a larger or bold font. Each tag links to a page where you'll find all of the content on that site that has been “tagged” (by publishers and/or users) with that term. Tag clouds are used by many Web 2.0 companies, including Technorati, Flickr, del.icio.us and more.

tagging—The act of adding tags to content.

tagscape—The tagging “landscape”; the patterns and trends that are seen in tagging and tag clouds.

TechCrunch—A popular Internet technology blog that focuses on the companies, products, people and events of Web 2.0.

Technorati—A popular blog search engine that uses tagging.

37Signals—The company that developed Ruby on Rails (<http://www.deitel.com/ResourceCenters/Programming/Ruby/tabid/715/Default.aspx>) and many SaaS applications, including Basecamp.

trackback—A method for web content authors to request notification when a website links to their content (articles, blog postings, etc.). It is a great way for authors to track links into their sites, measure the viral effects of their work, find related sites and more.

Twitter—A mobile web service that enables users to message groups of friends at once and automatically receive their friends' updates on a cell phone or through a chat window.

Ubuntu—A popular distribution of the Linux operating system.

user-generated content—Content that is created by users. User-generated content is central to Web 2.0.

ValueClick—An Internet advertising company.

vlogging—Video blogging.

VoIP (Voice over Internet Protocol)—Voice services over the Internet; used to build telephone services. The leading VoIP company is Skype, which offers free phone service among Skype users worldwide.

Vonage—A VoIP company. They provide broadband Internet telephone services that can be used with a standard phone (with adapter).

Web 1.0—The Internet experience previous to Web 2.0, focusing more on static content. Some people called it the “brochure web.”

Web 2.0—A term coined by Dale Dougherty of O'Reilly Media in 2003. It refers to the current state of the web, which has a strong emphasis on user participation and community. Web 2.0 sites include social networking, wikis, blogging, social media, tagging, collaborative filtering, and more.

web as a platform—Instead of viewing the operating system as the application platform and building “Windows-based applications” or “Linux-based applications,” developers now build “web-based applications.”

web of meaning—Another name for the “Semantic Web.”

Web Ontology Language (OWL)—A key Semantic Web technology, used for organizing data.

web publisher—A site that offers content. Advertisers place ads on web publisher sites.

web-scale computing—Refers to the ability to scale memory and processing power according to need, by using web-based processing power and memory, often provided by other companies. Amazon offers web-scale computing through web services such as Simple Storage Service (S3) and Elastic Compute Cloud (EC2).

web service—A service provided online that can be called by another program across the Internet.

Weblogsinc—A blog network.

webtop—A desktoplike environment (such as Laszlo Webtop) for running web applications in a web browser.

wiki—A collaborative, editable document online. The best known example of a wiki is Wikipedia, which has quickly become a leading web resource for virtually any topic.

Wikia—A site offering specialized wiki communities about popular television shows, games, literature, shopping and more.

Wikipedia—A community-generated encyclopedia using wiki technology.

wisdom of crowds—The concept that a large diverse group of individuals that does not necessarily include experts can provide more accurate answers than a small group of specialists working together.

WordPress—Popular blogging software.

World Wide Web Consortium (W3C)—An organization that develops web standards.

Xanga—A popular personal blogging site that includes community features.

XML (Extensible Markup Language)—A markup language developed in 1996 by the World Wide Web Consortium (W3C) that allows you to label data based on its meaning.

XML vocabularies—Customized XML-based markup languages, such as XHTML for web content, CML for chemistry, MathML for mathematical content and formulas, and XBRL for financial data.

Yahoo! Pipes—A mashup tool that enables you to aggregate and manipulate many data sources.

Yahoo! Publisher Network—Yahoo's contextual advertising program for publishers. This is a fundamental and popular form of monetization, particularly for Web 2.0 startup companies.

Yahoo! Search Marketing—Yahoo's advertising program for advertisers.

YouTube—An Internet video sharing site that has created a huge social phenomenon. Users upload and share videos. The company was bought by Google in 2006 for \$1.65 billion.

Zepheira—A company that provides Semantic Web knowledge management and enterprise data integration products and services.

Self-Review Exercises

3.1 Fill in the blanks in each of the following statements:

- _____ content refers to (legally) taking someone else's existing content and adding to it or changing it in some way.
- The term Web 2.0 was coined by _____ of O'Reilly® Media in 2003.
- _____ are user-generated labels used to categorize content.
- The major Ajax technologies are _____, _____, _____, _____, _____ and _____.
- _____ are webtop applications that have responsiveness and functionality approaching that of desktop applications.
- Amazon's hardware and communications infrastructure web services are examples of _____. They enable businesses to pay for only the processing or storage space needed during any given period.
- The increased value of a network as its number of users grows is referred to as _____.
- Two popular RIA frameworks are Adobe's _____ and Microsoft's _____.

3.2 State whether each of the following is *true* or *false*. If *false*, explain why.

- Tagging is for personal organization of content only.
- The user is at the center of Web 2.0.
- Location-based services always use GPS.
- Open source software is often called "free" because it does not cost money.
- Google's PageRank is determined by the number of page views a website receives.

Answers to Self-Review Exercises

3.1

- Remixing.
- Dale Dougherty
- Tags.
- XHTML, CSS, JavaScript, the DOM, XML, the XMLHttpRequest object.
- Rich Internet Applications (RIAs).
- web-scale computing.
- network effects (an aspect of Metcalfe's Law).
- Flex, Silverlight.

3.2

- False. User-generated tags are used by many websites to categorize content so that it is easily searchable by other users.
- True.
- False. Location-based services often use GPS; however, they often use other information to determine your location, such as your IP address.
- False. Open source software is free in terms of allowing access to the source code. It is not necessarily free of cost.
- False. The PageRank algorithm considers the number of links into a web page and the quality of the linking sites (among other factors) to determine the importance of the page. Google search also considers all of the content on the page, its fonts, its headers and the content of neighboring pages.

Exercises

3.3 Fill in the blanks in each of the following statements:

- _____ is an example of an agile development process.

- b) The _____ is a design that encourages user interaction and community contributions.
- c) Ruby on Rails was developed by _____.
- d) _____ systems add software to digital media to prevent piracy.
- e) _____ are attempts at consistent naming conventions.
- f) Wikis rely on the _____.

3.4

State whether each of the following is *true* or *false*. If *false*, explain why.

- a) Advertising is the most common Web 2.0 monetization model.
- b) Collaborative filtering is used by search engines.
- c) XML is an executable language.
- d) Most bloggers provide RSS feeds.
- e) Holding people's attention is difficult in today's society.

3.5

Define each of the following terms:

- a) collective intelligence.
- b) folksonomy.
- c) permalink.
- d) tag cloud.
- e) web service.
- f) monetization.

3.6

List some of the key factors that have attributed to the growth of Web 2.0.

3.7

Discuss some of the methods you can use to increase the findability of your website.

3.8

In Section 3.3 we discussed how many Web 2.0 sites are enabling discovery—helping you find new content you would not have otherwise sought out. Pick three Web 2.0 sites and describe how they are enabling you to discover new content through their sites.

3.9

Consider a picture of the Eiffel Tower. List 10 words you might use to tag this picture on a photosharing site such as Flickr so that others searching the site will find it.

2

PART

The Ajax Client

... 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

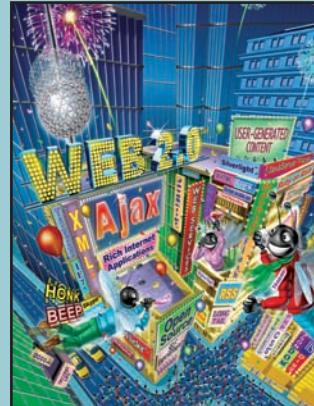
4

Introduction to XHTML

OBJECTIVES

In this chapter you will learn:

- To understand important components of XHTML documents.
- To use XHTML to create web pages.
- To add images to web pages.
- To create and use hyperlinks to navigate web pages.
- To mark up lists of information.
- To create tables with rows and columns of data and control table formatting.
- To create and use forms to get user input.
- To make web pages accessible to search engines using <meta> tags.



*To read between the lines
was easier than to follow the
text.*

—Henry James

*High thoughts must have
high language.*

—Aristophanes

*Yea, from the table of my
memory
I'll wipe away all trivial
fond records.*

—William Shakespeare

*He had a wonderful talent
for packing thought close,
and rendering it portable.*

—Thomas Babington
Macaulay

- 4.1 Introduction
- 4.2 Editing XHTML
- 4.3 First XHTML Example
- 4.4 W3C XHTML Validation Service
- 4.5 Headings
- 4.6 Linking
- 4.7 Images
- 4.8 Special Characters and Horizontal Rules
- 4.9 Lists
- 4.10 Tables
- 4.11 Forms
- 4.12 Internal Linking
- 4.13 meta Elements
- 4.14 Wrap-Up
- 4.15 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

4.1 Introduction

Welcome to the world of opportunity created by the World Wide Web. The Internet is almost four decades old, but it wasn't until the web's growth in popularity in the 1990s and the recent start of the Web 2.0 era that the explosion of opportunity we are experiencing began. Exciting new developments occur almost daily—the pace of innovation is unprecedented. In this chapter, you'll develop your own web pages. As the book proceeds, you'll create increasingly appealing and powerful web pages. Later in the book, you'll learn how to create complete web-based applications with databases and user interfaces.

This chapter begins unlocking the power of web-based application development with **XHTML**—the **Extensible HyperText Markup Language**. Later in the chapter, we introduce more sophisticated XHTML techniques such as **internal linking** for easier page navigation, **forms** for collecting information from a web-page visitor and **tables**, which are particularly useful for structuring information from **databases** (i.e., software that stores structured sets of data). In the next chapter, we discuss a technology called **Cascading Style SheetsTM (CSS)**, a technology that makes web pages more visually appealing.

Unlike procedural programming languages such as C, C++, or Java, XHTML is a **markup language** that specifies the format of the text that is displayed in a web browser such as Microsoft's Internet Explorer or Mozilla Firefox.

One key issue when using XHTML is the separation of the **presentation** of a document (i.e., the document's appearance when rendered by a browser) from the **structure** of the document's information. XHTML is based on HTML (HyperText Markup Language)—a legacy technology of the World Wide Web Consortium (W3C). In HTML, it was common to specify both the document's structure and its formatting. Formatting might specify where the browser placed an element in a web page or the fonts and colors used to display an element. The XHTML 1.0 Strict recommendation (the version of

XHTML that we use in this book) allows only a document's structure to appear in a valid XHTML document, and not its formatting. Normally, such formatting is specified with Cascading Style Sheets (Chapter 5). All our examples in this chapter are based upon the XHTML 1.0 Strict Recommendation.

4.2 Editing XHTML

In this chapter, we write XHTML in its **source-code form**. We create **XHTML documents** by typing them in a text editor (e.g., Notepad,TextEdit, vi, emacs) and saving them with either an `.html` or an `.htm` filename extension.



Good Programming Practice 4.1

Assign filenames to documents that describe their functionality. This practice can help you identify documents faster. It also helps people who want to link to a page, by giving them an easy-to-remember name. For example, if you are writing an XHTML document that contains product information, you might want to call it products.html.

Computers called **web servers** running specialized software store XHTML documents. Clients (e.g., web browsers) request specific **resources** such as the XHTML documents from web servers. For example, typing `www.deitel.com/books/downloads.html` into a web browser's address field requests `downloads.html` from the `books` directory on the web server running at `www.deitel.com`. We discuss web servers in detail in Chapter 21. For now, we simply place the XHTML documents on our computer and render them by opening them locally with a web browser such as Internet Explorer or Firefox.

4.3 First XHTML Example

This chapter presents XHTML markup and provides screen captures that show how a browser renders (i.e., displays) the XHTML. You can download the examples from `www.deitel.com/books/iw3htp4`. Every XHTML document we show has line numbers for the reader's convenience—these line numbers are not part of the XHTML documents. As you read this book, open each XHTML document in your web browser so you can view and interact with it as it was originally intended.

Figure 4.1 is an XHTML document named `main.html`. This first example displays the message “Welcome to XHTML!” in the browser. The key line in the program is line 13, which tells the browser to display “Welcome to XHTML!” Now let us consider each line of the program.

Lines 1–3 are required in XHTML documents to conform with proper XHTML syntax. For now, copy and paste these lines into each XHTML document you create. The meaning of these lines is discussed in detail in Chapter 14.

Lines 5–6 are **XHTML comments**. XHTML document creators insert comments to improve markup readability and describe the content of a document. Comments also help other people read and understand an XHTML document's markup and content. Comments do not cause the browser to perform any action when the user loads the XHTML document into the web browser to view it. XHTML comments always start with `<!--` and end with `-->`. Each of our XHTML examples includes comments that specify the figure number and filename and provide a brief description of the example's purpose. Subsequent examples include comments in the markup, especially to highlight new features.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.1: main.html -->
6 <!-- First XHTML example. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Welcome</title>
10   </head>
11
12   <body>
13     <p>Welcome to XHTML!</p>
14   </body>
15 </html>
```

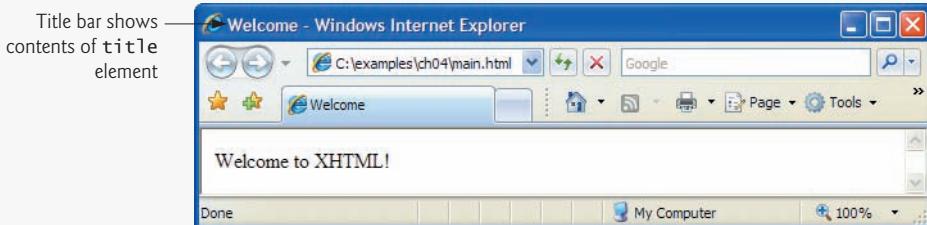


Fig. 4.1 | First XHTML example.



Good Programming Practice 4.2

Place comments throughout your markup. Comments help other programmers understand the markup, assist in debugging and list useful information that you do not want the browser to render. Comments also help you understand your own markup when you revisit a document to modify or update it in the future.

XHTML markup contains text that represents the content of a document and **elements** that specify a document's structure. Some important elements of an XHTML document are the **html** element, the **head** element and the **body** element. The **html** element encloses the **head section** (represented by the **head** element) and the **body section** (represented by the **body** element). The head section contains information about the XHTML document, such as its **title**. The head section also can contain special document formatting instructions called **style sheets** and client-side programs called **scripts** for creating dynamic web pages. (We introduce style sheets in Chapter 5 and scripting with JavaScript in Chapter 6.) The body section contains the page's content that the browser displays when the user visits the web page.

XHTML documents delimit an element with **start** and **end** tags. A start tag consists of the element name in angle brackets (e.g., `<html>`). An end tag consists of the element name preceded by a forward slash (/) in angle brackets (e.g., `</html>`). In this example, lines 7 and 15 define the start and end of the **html** element. Note that the end tag in line 15 has the same name as the start tag, but is preceded by a / inside the angle brackets. Many start tags have **attributes** that provide additional information about an element. Browsers can use this additional information to determine how to process the element.

Each attribute has a **name** and a **value** separated by an equals sign (=). Line 7 specifies a required attribute (`xmlns`) and value (`http://www.w3.org/1999/xhtml`) for the `html` element in an XHTML document. For now, simply copy and paste the `html` element start tag in line 7 into your XHTML documents. We discuss the details of the `xmlns` attribute in Chapter 14.



Common Programming Error 4.1

Not enclosing attribute values in either single or double quotes is a syntax error. However, some web browsers may still render the element correctly.



Common Programming Error 4.2

Using uppercase letters in an XHTML element or attribute name is a syntax error. However, some web browsers may still render the element correctly.

An XHTML document divides the `html` element into two sections—head and body. Lines 8–10 define the web page’s head section with a `head` element. Line 9 specifies a `title` element. This is called a **nested element** because it is enclosed in the `head` element’s start and end tags. The `head` element is also a nested element because it is enclosed in the `html` element’s start and end tags. The `title` element describes the web page. Titles usually appear in the **title bar** at the top of the browser window, in the browser tab that the page is displayed on, and also as the text identifying a page when users add the page to their list of **Favorites** or **Bookmarks** that enables them to return to their favorite sites. Search engines (i.e., sites that allow users to search the web) also use the `title` for indexing purposes.



Good Programming Practice 4.3

Indenting nested elements emphasizes a document’s structure and promotes readability.



Common Programming Error 4.3

XHTML does not permit tags to overlap—a nested element’s end tag must appear in the document before the enclosing element’s end tag. For example, the nested XHTML tags `<head><title>hello</head></title>` cause a syntax error, because the enclosing `head` element’s ending `</head>` tag appears before the nested `title` element’s ending `</title>` tag.



Good Programming Practice 4.4

Use a consistent title-naming convention for all pages on a site. For example, if a site is named “Bailey’s Website,” then the title of the contact page might be “Bailey’s Website - Contact.” This practice can help users better understand the website’s structure.

Line 12 begins the document’s body element. The body section of an XHTML document specifies the document’s content, which may include text and elements.

Some elements, such as the **paragraph element** (denoted with `<p>` and `</p>`) in line 13, mark up text for display in a browser. All the text placed between the `<p>` and `</p>` tags forms one paragraph. When the browser renders a paragraph, a blank line usually precedes and follows paragraph text.

This document ends with two end tags (lines 14–15). These tags close the body and `html` elements, respectively. The `</html>` tag in an XHTML document informs the browser that the XHTML markup is complete.

To open an XHTML example from this chapter, open the folder where you saved the book's examples, browse to the ch04 folder and double click the file to open it in your default web browser. At this point your browser window should appear similar to the sample screen capture shown in Fig. 4.1. (Note that we resized the browser window to save space in the book.)

4.4 W3C XHTML Validation Service

Programming web-based applications can be complex, and XHTML documents must be written correctly to ensure that browsers process them properly. To promote correctly written documents, the World Wide Web Consortium (W3C) provides a [validation service](http://validator.w3.org) (`validator.w3.org`) for checking a document's syntax. Documents can be validated by providing a URL that specifies the file's location, by uploading a file to `validator.w3.org/file-upload.html` or by pasting code directly into a text area. Uploading a file copies the file from the user's computer to another computer on the Internet. The W3C's web page indicates that the service name is **MarkUp Validation Service** and that the validation service is able to validate the syntax of XHTML documents. All the XHTML examples in this book have been validated successfully using `validator.w3.org`.

By clicking **Choose...**, users can select files on their own computers for upload. After selecting a file, clicking the **Check** button uploads and validates the file. If a document contains syntax errors, the validation service displays error messages describing the errors.



Error-Prevention Tip 4.1

Most current browsers attempt to render XHTML documents even if they are invalid. This often leads to unexpected and possibly undesirable results. Use a validation service, such as the W3C MarkUp Validation Service, to confirm that an XHTML document is syntactically correct.

4.5 Headings

Some text in an XHTML document may be more important than other text. For example, the text in this section is considered more important than a footnote. XHTML provides six [headings](#), called **heading elements**, for specifying the relative importance of information. Figure 4.2 demonstrates these elements (`h1` through `h6`). Heading element `h1` (line 13) is considered the most significant heading and is typically rendered in a larger font than the other five headings (lines 14–18). Each successive heading element (i.e., `h2`, `h3`, etc.) is typically rendered in a progressively smaller font.



Portability Tip 4.1

The text size used to display each heading element can vary significantly between browsers. In Chapter 5, we discuss how to control the text size and other text properties.



Look-and-Feel Observation 4.1

Placing a heading at the top of every XHTML page helps viewers understand the purpose of each page.



Look-and-Feel Observation 4.2

Use larger headings to emphasize more important sections of a web page.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.2: heading.html -->
6 <!-- Heading elements h1 through h6. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Headings</title>
10    </head>
11
12  <body>
13    <h1>Level 1 Heading</h1>
14    <h2>Level 2 heading</h2>
15    <h3>Level 3 heading</h3>
16    <h4>Level 4 heading</h4>
17    <h5>Level 5 heading</h5>
18    <h6>Level 6 heading</h6>
19  </body>
20 </html>

```

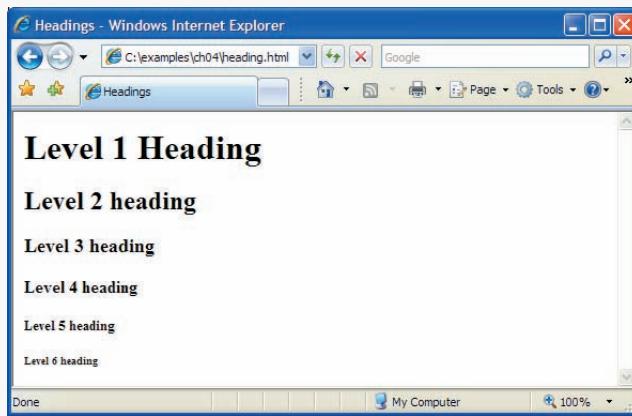


Fig. 4.2 | Heading elements h1 through h6.

4.6 Linking

One of the most important XHTML features is the **hyperlink**, which references (or **links** to) other resources, such as XHTML documents and images. When a user clicks a hyperlink, the browser tries to execute an action associated with it (e.g., navigate to a URL, open an e-mail client, etc.). In XHTML, both text and images can act as hyperlinks. Web browsers typically underline text hyperlinks and color their text blue by default, so that users can distinguish hyperlinks from plain text. In Fig. 4.3, we create text hyperlinks to four different websites.

Line 14 introduces the **strong element**, which indicates that its contents has high importance. Browsers typically display such text in a bold font.

Links are created using the **a (anchor)** element. Line 17 defines a hyperlink to the URL assigned to attribute **href**, which specifies the location of a linked resource, such as

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.3: links.html -->
6 <!-- Linking to other web pages. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Links</title>
10    </head>
11
12  <body>
13    <h1>Here are my favorite sites</h1>
14    <p><strong>Click a name to go to that page.</strong></p>
15
16    <!-- Create four text hyperlinks -->
17    <p><a href = "http://www.deitel.com">Deitel</a></p>
18    <p><a href = "http://www.prenhall.com">Prentice Hall</a></p>
19    <p><a href = "http://www.yahoo.com">Yahoo!</a></p>
20    <p><a href = "http://www.usatoday.com">USA Today</a></p>
21
22 </body>
</html>
```



Fig. 4.3 | Linking to other web pages.

a web page, a file or an e-mail address. This particular anchor element links the text Deitel to a web page located at `http://www.deitel.com`. When a URL does not indicate a specific document on the website, the web server returns a default web page. This page is often called `index.html`; however, most web servers can be configured to use any file as the default web page for the site. If the web server cannot locate a requested document, it returns an error indication to the web browser, and the browser displays a web page containing an error message to the user.

Hyperlinking to an E-Mail Address

Anchors can link to e-mail addresses using a `mailto:` URL. When someone clicks this type of anchored link, most browsers launch the default e-mail program (e.g., Microsoft Outlook or Mozilla Thunderbird) to enable the user to write an e-mail message to the linked address. Figure 4.4 demonstrates this type of anchor. Lines 15–17 contain an e-mail link. The form of an e-mail anchor is `...`. In this case, we link to the e-mail address `deitel@deitel.com`.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.4: contact.html -->
6 <!-- Linking to an e-mail address. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Contact Page</title>
10    </head>
11
12   <body>
13     <p>
14       My email address is
15       <a href = "mailto:deitel@deitel.com">
16         deitel@deitel.com
17       </a>
18       . Click the address and your default email client
19       will open an e-mail message and address it to me.
20     </p>
21   </body>
22 </html>

```

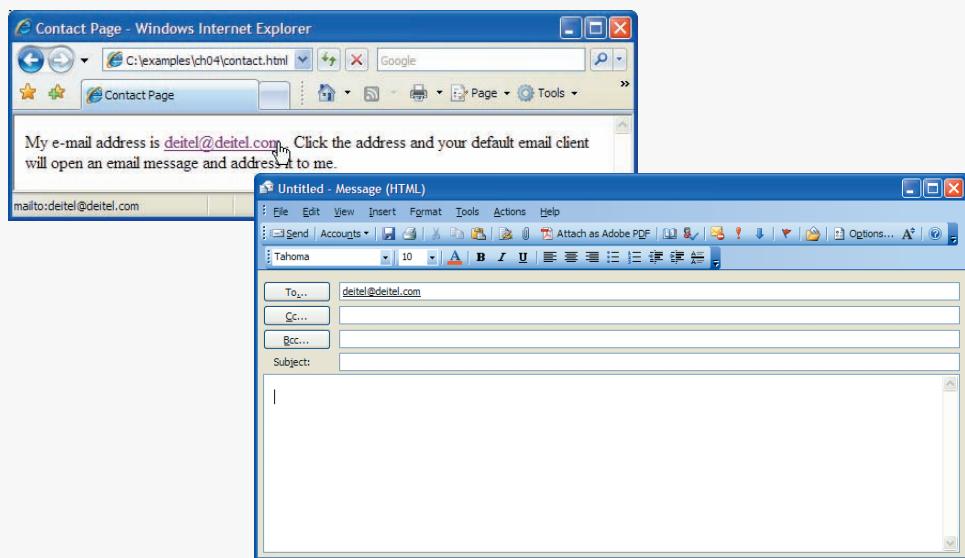


Fig. 4.4 | Linking to an e-mail address.

4.7 Images

The examples discussed so far demonstrate how to mark up documents that contain only text. However, most web pages contain both text and images. In fact, images are an equally important, if not essential, part of web-page design. The three most popular image formats used by web developers are Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG) and Portable Network Graphics (PNG) images. Users can create images using specialized software, such as Adobe Photoshop Elements (www.adobe.com), G.I.M.P. (<http://www.gimp.org>) and Inkscape (<http://www.inkscape.org>). Images may also be acquired from various websites. Figure 4.5 demonstrates how to incorporate images into web pages.

Lines 14–15 use an **img element** to insert an image in the document. The image file's location is specified with the **img element's src attribute**. This image is located in the same directory as the XHTML document, so only the image's filename is required. Optional attributes **width** and **height** specify the image's width and height, respectively. You can scale an image by increasing or decreasing the values of the image **width** and **height** attributes. If these attributes are omitted, the browser uses the image's actual width and height. Images are measured in **pixels** ("picture elements"), which represent dots of color on the screen. Any image-editing program will have a feature that displays the dimensions, in pixels, of an image. The image in Fig. 4.5 is 92 pixels wide and 120 pixels high.



Good Programming Practice 4.5

Always include the width and the height of an image inside the `` tag. When the browser loads the XHTML file, it will know immediately from these attributes how much screen space to provide for the image and will lay out the page properly, even before it downloads the image.



Performance Tip 4.1

Including the width and height attributes in an `` tag can result in the browser's loading and rendering pages faster.



Common Programming Error 4.4

Entering new dimensions for an image that change its inherent width-to-height ratio distorts the appearance of the image. For example, if your image is 200 pixels wide and 100 pixels high, you should ensure that any new dimensions have a 2:1 width-to-height ratio.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.5: picture.html -->
6 <!-- Images in XHTML files. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Images</title>
10   </head>
11 
```

Fig. 4.5 | Images in XHTML files. (Part 1 of 2.)

```

12 <body>
13   <p>
14     <img src = "cpphtp6.jpg" width = "92" height = "120"
15       alt = "C++ How to Program book cover" />
16     <img src = "jhtp.jpg" width = "92" height = "120"
17       alt = "Java How to Program book cover" />
18   </p>
19 </body>
20 </html>

```

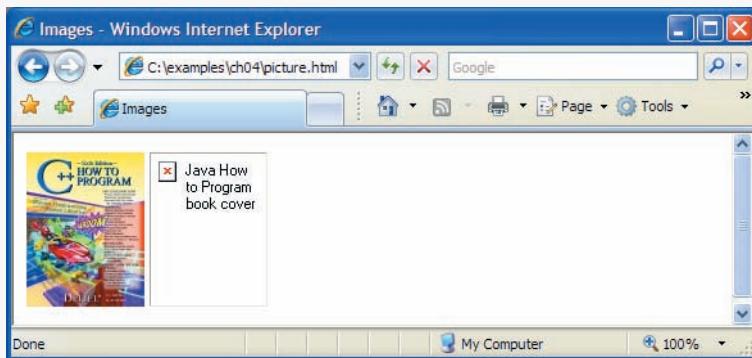


Fig. 4.5 | Images in XHTML files. (Part 2 of 2.)

Every `img` element in an XHTML document must have an `alt` attribute. If a browser cannot render an image, the browser displays the `alt` attribute's value. A browser may not be able to render an image for several reasons. It may not support images—as is the case with a **text-based browser** (i.e., a browser that can display only text)—or the client may have disabled image viewing to reduce download time. Figure 4.5 shows Internet Explorer 7 rendering a red `X` symbol and displaying the `alt` attribute's value, signifying that the image (`jhtp.jpg`) cannot be found.

The `alt` attribute helps you create **accessible** web pages for users with disabilities, especially those with vision impairments who use text-based browsers. Specialized software called a **speech synthesizer** can “speak” the `alt` attribute's value so that a user with a visual impairment knows what the browser is displaying.

Some XHTML elements (called **empty elements**) contain only attributes and do not mark up text (i.e., text is not placed between the start and end tags). Empty elements (e.g., `img`) must be terminated, either by using the **forward slash character** (/) inside the closing right angle bracket (>) of the start tag or by explicitly including the end tag. When using the forward slash character, we add a space before it to improve readability (as shown at the ends of lines 15 and 17). Rather than using the forward slash character, lines 16–17 could be written with a closing `` tag as follows:

```

<img src = "jhtp.jpg" width = "92" height = "120"
      alt = "Java How to Program book cover"></img>

```

Using Images as Hyperlinks

By using images as hyperlinks, web developers can create graphical web pages that link to other resources. In Fig. 4.6, we create six different image hyperlinks.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.6: nav.html -->
6 <!-- Images as Link anchors. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Navigation Bar</title>
10    </head>
11
12   <body>
13     <p>
14       <a href = "links.html">
15         <img src = "buttons/links.jpg" width = "65"
16           height = "50" alt = "Links Page" />
17       </a>
18
19       <a href = "list.html">
20         <img src = "buttons/list.jpg" width = "65"
21           height = "50" alt = "List Example Page" />
22       </a>
23
24       <a href = "contact.html">
25         <img src = "buttons/contact.jpg" width = "65"
26           height = "50" alt = "Contact Page" />
27       </a>
28
29       <a href = "table1.html">
30         <img src = "buttons/table.jpg" width = "65"
31           height = "50" alt = "Table Page" />
32       </a>
33
34       <a href = "form.html">
35         <img src = "buttons/form.jpg" width = "65"
36           height = "50" alt = "Feedback Form" />
37       </a>
38     </p>
39   </body>
40 </html>
```

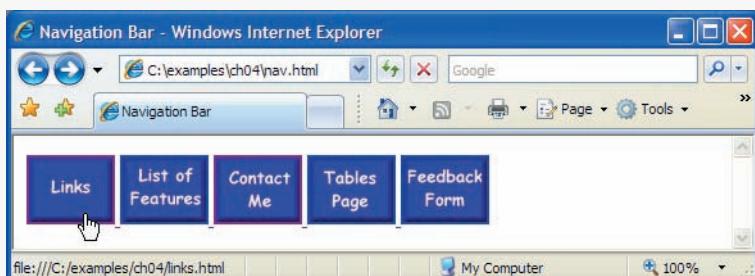


Fig. 4.6 | Images as link anchors. (Part I of 2.)

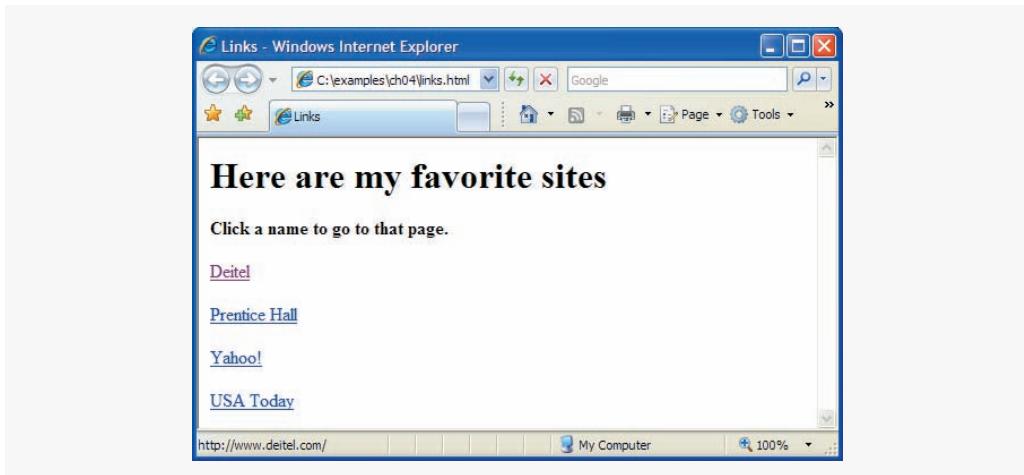


Fig. 4.6 | Images as link anchors. (Part 2 of 2.)

Lines 14–17 create an **image hyperlink** by nesting an `img` element in an anchor (`a`) element. The value of the `img` element's `src` attribute value specifies that this image (`links.jpg`) resides in a directory named `buttons`. The `buttons` directory and the XHTML document are in the same directory. Images from other web documents also can be referenced by setting the `src` attribute to the name and location of the image. Note that if you're hosting a publicly available web page that uses an image from another site, you should get permission to use the image and host a copy of image on your own website. If you refer to an image on another website, the browser has to request the image resource from the other site's server. Clicking an image hyperlink takes a user to the web page specified by the surrounding anchor element's `href` attribute. Notice that when the mouse hovers over a link of any kind, the URL that the link points to is displayed in the status bar at the bottom of the browser window.

4.8 Special Characters and Horizontal Rules

When marking up text, certain characters or symbols (e.g., `<`) may be difficult to embed directly into an XHTML document. Some keyboards do not provide these symbols, or the presence of these symbols may cause syntax errors. For example, the markup

```
<p>if x < 10 then increment x by 1</p>
```

results in a syntax error because it uses the less-than character (`<`), which is reserved for start tags and end tags such as `<p>` and `</p>`. XHTML provides **character entity references** (in the form `&code;`) for representing special characters. We could correct the previous line by writing

```
<p>if x &lt; 10 then increment x by 1</p>
```

which uses the character entity reference `<` for the less-than symbol (`<`).

Figure 4.7 demonstrates how to use special characters in an XHTML document. For a list of special characters, see Appendix A, XHTML Special Characters.

Lines 24–25 contain other special characters, which can be expressed as either character entity references (coded using word abbreviations such as & for ampersand and © for copyright) or **numeric character references**—decimal or **hexadecimal (hex)** values representing special characters. For example, the & character is represented in decimal and hexadecimal notation as #38; and #x26;, respectively. Hexadecimal numbers are base 16 numbers—digits in a hexadecimal number have values from 0 to 15 (a total of 16 different values). The letters A–F represent the hexadecimal digits corresponding to decimal values 10–15. Thus in hexadecimal notation we can have numbers like 876 consisting solely of decimal-like digits, numbers like DA19F consisting of digits and letters, and numbers like DCB consisting solely of letters. We discuss hexadecimal numbers in detail in Appendix E, Number Systems.

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 4.7: contact2.html -->
6  <!-- Inserting special characters. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8    <head>
9      <title>Contact Page</title>
10     </head>
11
12   <body>
13     <p>
14       Click
15       <a href = "mailto:deitel@deitel.com">here</a>
16       to open an email message addressed to
17       deitel@deitel.com.
18     </p>
19
20     <hr /> <!-- inserts a horizontal rule -->
21
22     <!-- special characters are entered -->
23     <!-- using the form &code; -->
24     <p>All information on this site is <strong>&copy;
25       Deitel &amp; Associates, Inc. 2007.</strong></p>
26
27     <!-- to strike through text use <del> tags -->
28     <!-- to subscript text use <sub> tags -->
29     <!-- to superscript text use <sup> tags -->
30     <!-- these tags are nested inside other tags -->
31     <p><del>You may download 3.14 x 10<sup>2</sup>
32       characters worth of information from this site.</del>
33       Only <sub>one</sub> download per hour is permitted.</p>
34     <p><em>Note: &lt; &frac14; of the information
35       presented here is updated daily.</em></p>
36   </body>
37 </html>
```

Fig. 4.7 | Inserting special characters. (Part I of 2.)

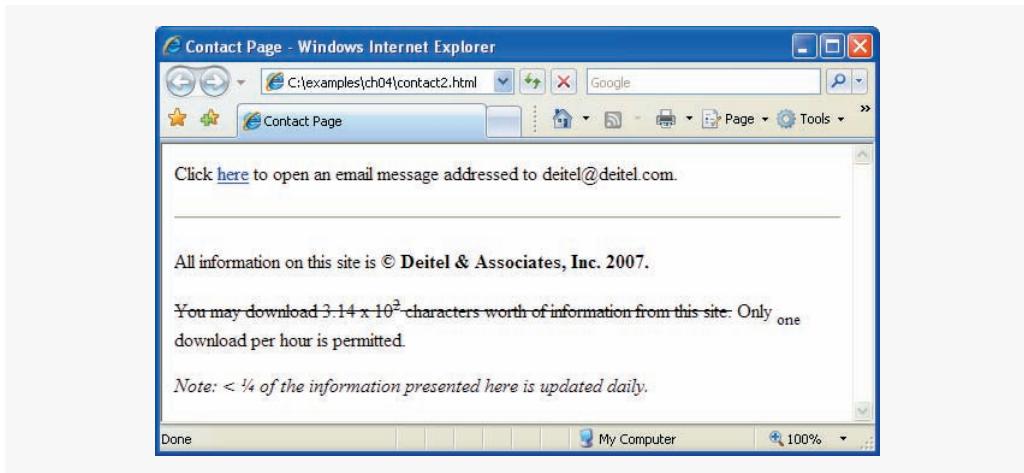


Fig. 4.7 | Inserting special characters. (Part 2 of 2.)

In lines 31–33, we introduce four new elements. Most browsers render the **del** element as strike-through text. With this format users can easily indicate document revisions. To **superscript** text (i.e., raise text above the baseline and decreased font size) or **subscript** text (i.e., lower text below the baseline and decreased font size), use the **sup** or **sub** element, respectively. The paragraph in lines 34–35 contains an **em** element, which indicates that its contents should be emphasized. Browsers usually render **em** elements in an italic font. We also use character entity reference **<** for a less-than sign and **¼** for the fraction 1/4 (line 34).

In addition to special characters, this document introduces a **horizontal rule**, indicated by the **<hr />** tag in line 22. Most browsers render a horizontal rule as a horizontal line with a blank line above and below it.

4.9 Lists

Up to this point, we have presented basic XHTML elements and attributes for linking to resources, creating headings, using special characters and incorporating images. In this section, we discuss how to organize information on a web page using lists. In the next section, we introduce another feature for organizing information, called a table. Figure 4.8 displays text in an **unordered list** (i.e., a list that does not order its items by letter or number). The unordered list element **ul** creates a list in which each item begins with a bullet symbol (called a disc). Each entry in an unordered list (element **ul** in line 17) is an **li** (**list item**) element (lines 19–22). Most web browsers render each **li** element on a new line with a bullet symbol indented from the beginning of the line.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4

```

Fig. 4.8 | Unordered list containing hyperlinks. (Part 1 of 2.)

```

5  <!-- Fig. 4.8: links2.html -->
6  <!-- Unordered list containing hyperlinks. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Links</title>
10     </head>
11
12     <body>
13         <h1>Here are my favorite sites</h1>
14         <p><strong>Click on a name to go to that page.</strong></p>
15
16         <!-- create an unordered list -->
17         <ul>
18             <!-- add four list items -->
19             <li><a href = "http://www.deitel.com">Deitel</a></li>
20             <li><a href = "http://www.w3.org">W3C</a></li>
21             <li><a href = "http://www.yahoo.com">Yahoo!</a></li>
22             <li><a href = "http://www.cnn.com">CNN</a></li>
23         </ul>
24     </body>
25 </html>

```



Fig. 4.8 | Unordered list containing hyperlinks. (Part 2 of 2.)

Nested Lists

Lists may be nested to represent hierarchical relationships, as in an outline format. Figure 4.9 demonstrates nested lists and **ordered lists**. The ordered list element **ol** creates a list in which each item begins with a number.

A web browser indents each nested list to indicate a hierarchical relationship. The first ordered list begins at line 30. Items in an ordered list are enumerated one, two, three and so on. Nested ordered lists are enumerated in the same manner. The items in the outermost unordered list (line 16) are preceded by discs. List items nested inside the unordered list of line 16 are preceded by circular bullets. Although not demonstrated in this example, subsequent nested list items are preceded by square bullets.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.9: list.html -->
6 <!-- Nested and ordered lists. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Lists</title>
10    </head>
11
12  <body>
13    <h1>The Best Features of the Internet</h1>
14
15    <!-- create an unordered list -->
16    <ul>
17      <li>You can meet new people from countries around
18        the world.</li>
19      <li>
20        You have access to new media as it becomes public:
21
22        <!-- this starts a nested list, which uses a -->
23        <!-- modified bullet. The list ends when you -->
24        <!-- close the <ul> tag. -->
25        <ul>
26          <li>New games</li>
27          <li>New applications
28
29            <!-- nested ordered list -->
30            <ol>
31              <li>For business</li>
32              <li>For pleasure</li>
33            </ol>
34          </li> <!-- ends line 27 new applications li -->
35
36          <li>Around the clock news</li>
37          <li>Search engines</li>
38          <li>Shopping</li>
39          <li>Programming
40
41            <!-- another nested ordered list -->
42            <ol>
43              <li>XML</li>
44              <li>Java</li>
45              <li>XHTML</li>
46              <li>Scripts</li>
47              <li>New languages</li>
48            </ol>
49          </li> <!-- ends programming li of line 39 -->
50        </ul> <!-- ends the nested list of line 25 -->
51      </li>
52    </body>
```

Fig. 4.9 | Nested and ordered lists. (Part I of 2.)

```
53      <li>Links</li>
54      <li>Keeping in touch with old friends</li>
55      <li>It is the technology of the future!</li>
56  </ul> <!-- ends the unordered list of line 16 -->
57 </body>
58 </html>
```

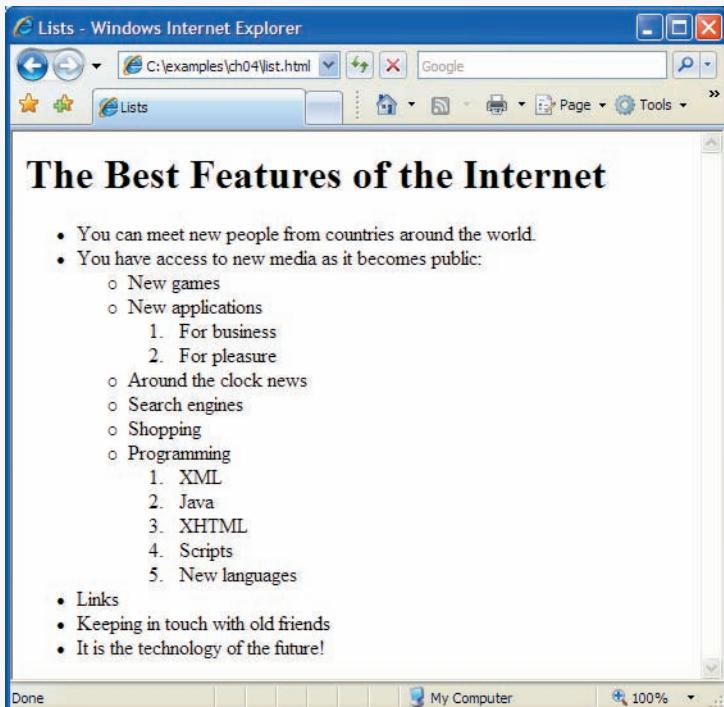


Fig. 4.9 | Nested and ordered lists. (Part 2 of 2.)

4.10 Tables

Tables are frequently used to organize data into rows and columns. Our first example (Fig. 4.10) creates a table with six rows and two columns to display price information for fruit.

Tables are defined with the **table** element (lines 15–62). Lines 15–17 specify the start tag for a **table** element that has several attributes. The **border** attribute specifies the table's border width in pixels. To create a table without a border, set **border** to "0". This example assigns attribute **width** the value "40%" to set the table's width to 40 percent of the browser's width. A developer can also set attribute **width** to a specified number of pixels. Try resizing the browser window to see how the width of the window affects the width of the table.

As its name implies, attribute **summary** (lines 16–17) describes the table's contents. Speech devices use this attribute to make the table more accessible to users with visual impairments. The **caption** element (line 21) describes the table's content and helps text-

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.10: table1.html -->
6 <!-- Creating a basic table. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>A simple XHTML table</title>
10   </head>
11
12   <body>
13
14     <!-- the <table> tag opens a table -->
15     <table border = "1" width = "40%" 
16       summary = "This table provides information about
17         the price of fruit">
18
19       <!-- the <caption> tag summarizes the table's -->
20       <!-- contents (this helps the visually impaired) -->
21       <caption><strong>Price of Fruit</strong></caption>
22
23       <!-- the <thead> section appears first in the table -->
24       <!-- it formats the table header area -->
25       <thead>
26         <tr> <!-- <tr> inserts a table row -->
27           <th>Fruit</th> <!-- insert a heading cell -->
28           <th>Price</th>
29         </tr>
30       </thead>
31
32       <!-- the <tfoot> section appears last in the table -->
33       <!-- it formats the table footer -->
34       <tfoot>
35         <tr>
36           <th>Total</th>
37           <th>$3.75</th>
38         </tr>
39       </tfoot>
40
41       <!-- all table content is enclosed -->
42       <!-- within the <tbody> -->
43       <tbody>
44         <tr>
45           <td>Apple</td> <!-- insert a data cell -->
46           <td>$0.25</td>
47         </tr>
48         <tr>
49           <td>Orange</td>
50           <td>$0.50</td>
51         </tr>
52         <tr>
53           <td>Banana</td>
```

Fig. 4.10 | Creating a basic table. (Part I of 2.)

```

54          <td>$1.00</td>
55      </tr>
56      <tr>
57          <td>Pineapple</td>
58          <td>$2.00</td>
59      </tr>
60  </tbody>
61 </table>
62
63  </body>
64 </html>

```

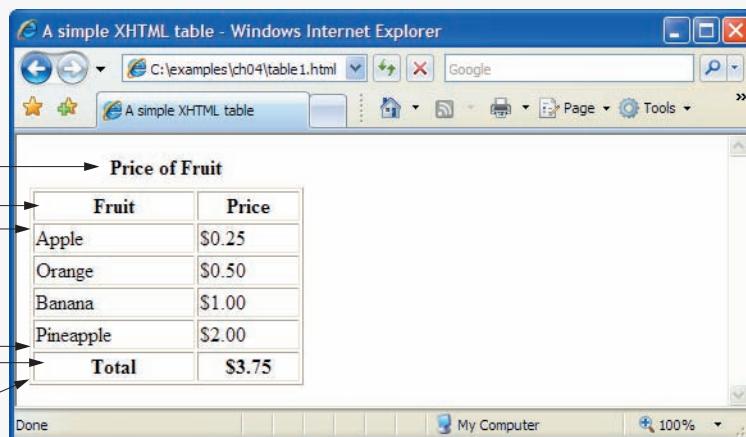


Fig. 4.10 | Creating a basic table. (Part 2 of 2.)

based browsers interpret the table data. Text inside the `<caption>` tag is rendered above the table by most browsers. Attribute `summary` and element `caption` are two of the many XHTML features that make web pages more accessible to users with disabilities.

A table has three distinct sections—**head**, **body** and **foot**. The head section (or header cell) is defined with a `thead` element (lines 25–30), which contains header information such as column names. Each `tr` element (lines 26–29) defines an individual **table row**. The columns in the head section are defined with `th` elements. Most browsers center text formatted by `th` (table header column) elements and display them in bold. Table header elements are nested inside table row elements (lines 27–28).

The body section, or **table body**, contains the table's primary data. The table body (lines 43–60) is defined in a `tbody` element. In the body, each `tr` element specifies one row. **Data cells** contain individual pieces of data and are defined with `td` (**table data**) elements in each row.

The foot section (lines 34–39) is defined with a `tfoot` (table foot) element. The text placed in the footer commonly includes calculation results and footnotes. Like other sections, the foot section can contain table rows, and each row can contain cells. As in the head section, cells in the foot section are created using `th` elements, instead of the `td` elements used in the table body. Note that the table foot section must be above the body section in the code, but the table foot displays at the bottom of the table.

Using rowspan and colspan

Figure 4.10 explored a basic table's structure. Figure 4.11 presents another table example and introduces two new attributes that allow you to build more complex tables.

The table begins in line 15. Table cells are sized to fit the data they contain. Document authors can create larger data cells using the attributes `rowspan` and `colspan`. The values assigned to these attributes specify the number of rows or columns occupied by a cell. The `th` element at lines 23–26 uses the attribute `rowspan = "2"` to allow the cell containing the picture of the camel to use two vertically adjacent cells (thus the cell *spans* two rows). The `th` element in lines 29–32 uses the attribute `colspan = "4"` to widen the header cell (containing Camelid comparison and Approximate as of 6/2007) to span four cells.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 4.11: table2.html -->
6  <!-- Complex XHTML table. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Tables</title>
10     </head>
11
12     <body>
13         <h1>Table Example Page</h1>
14
15         <table border = "1">
16             <caption>Here is a more complex sample table.</caption>
17
18             <thead>
19                 <!-- rowspans and colspans merge the specified -->
20                 <!-- number of cells vertically or horizontally -->
21                 <tr>
22                     <!-- merge two rows -->
23                     <th rowspan = "2">
24                         <img src = "camel.gif" width = "205"
25                             height = "167" alt = "Picture of a camel" />
26                     </th>
27
28                     <!-- merge four columns -->
29                     <th colspan = "4">
30                         <h1>Camelid comparison</h1>
31                         <p>Approximate as of 6/2007</p>
32                     </th>
33                 </tr>
34                 <tr>
35                     <th># of Humps</th>
36                     <th>Indigenous region</th>
37                     <th>Spits?</th>
38                     <th>Produces Wool?</th>
39                 </tr>
40             </thead>
```

Fig. 4.11 | Complex XHTML table. (Part I of 2.)

```

41      <tbody>
42          <tr>
43              <th>Camels (bactrian)</th>
44              <td>2</td>
45              <td>Africa/Asia</td>
46              <td>Yes</td>
47              <td>Yes</td>
48          </tr>
49          <tr>
50              <th>Llamas</th>
51              <td>1</td>
52              <td>Andes Mountains</td>
53              <td>Yes</td>
54              <td>Yes</td>
55          </tr>
56      </tbody>
57  </table>
58 </body>
59 </html>

```

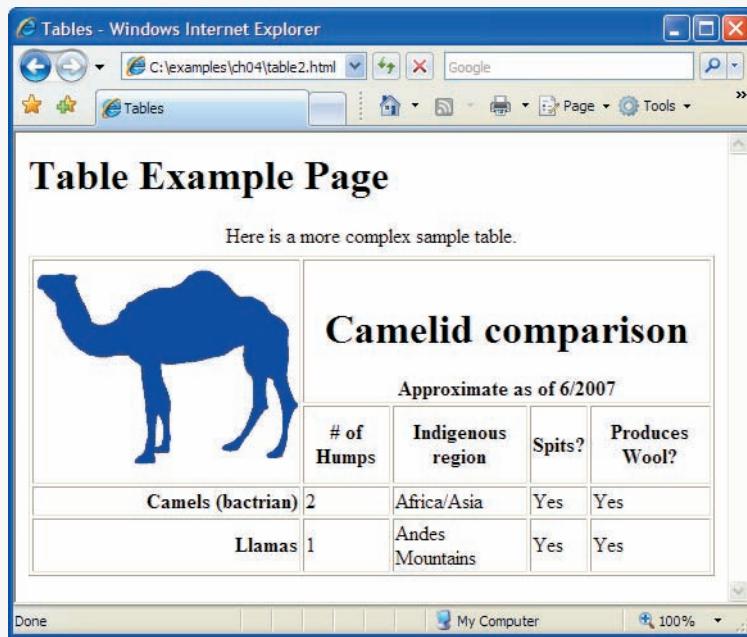


Fig. 4.11 | Complex XHTML table. (Part 2 of 2.)

4.11 Forms

When browsing websites, users often need to provide such information as search keywords, e-mail addresses and zip codes. XHTML provides a mechanism, called a form, for collecting data from a user.

Data that users enter on a web page is normally sent to a web server that provides access to a site's resources (e.g., XHTML documents, images). These resources are located

either on the same machine as the web server or on a machine that the web server can access through the network. When a browser requests a web page or file that is located on a server, the server processes the request and returns the requested resource. A request contains the name and path of the desired resource and the method of communication (called a **protocol**). XHTML documents use the Hypertext Transfer Protocol (HTTP).

Figure 4.12 is a simple form that sends data to the web server, which passes the form data to a program. The program processes the data received from the web server and typically returns information to the web server. The web server then sends the information as an XHTML document to the web browser. We discuss web servers in Chapter 21. [Note: This example demonstrates client-side functionality. If the form is submitted (by clicking **Submit**), nothing will happen, because we don't yet know how to process the form data. In later chapters, we present the server-side programming (e.g., in ASP.NET, JavaServer Faces, and PHP) necessary to process information entered into a form.]

Forms can contain visual and nonvisual components. Visual components include clickable buttons and other graphical user interface components with which users interact. Nonvisual components, called **hidden inputs**, store any data that you specify, such as e-mail addresses and XHTML document filenames that act as links. The form is defined in lines 21–46 by a **form** element.

Attribute **method** (line 21) specifies how the form's data is sent to the web server. Using **method = "post"** appends form data to the browser request, which contains the protocol (HTTP) and the requested resource's URL. This method of passing data to the server is transparent—the user doesn't see the data after the form is submitted. The other possible value, **method = "get"**, appends the form data directly to the end of the URL of the script, where it is visible in the browser's **Address** field. The *post* and *get* methods for sending form data are discussed in detail in Chapter 21, Web Servers (IIS and Apache).

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.12: form.html -->
6 <!-- Form with hidden fields and a text box. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Forms</title>
10    </head>
11
12  <body>
13    <h1>Feedback Form</h1>
14
15    <p>Please fill out this form to help
16      us improve our site.</p>
17
18    <!-- this tag starts the form, gives the -->
19    <!-- method of sending information and the -->
20    <!-- location of form script -->
21    <form method = "post" action = "">
22      <p>
```

Fig. 4.12 | Form with hidden fields and a text box. (Part 1 of 2.)

```

23      <!-- hidden inputs contain non-visual -->
24      <!-- information -->
25      <input type = "hidden" name = "recipient"
26          value = "deitel@deitel.com" />
27      <input type = "hidden" name = "subject"
28          value = "Feedback Form" />
29      <input type = "hidden" name = "redirect"
30          value = "main.html" />
31  </p>
32
33      <!-- <input type = "text"> inserts a text box -->
34  <p><label>Name:
35      <input name = "name" type = "text" size = "25"
36          maxLength = "30" />
37  </label></p>
38
39  <p>
40      <!-- input types "submit" and "reset" insert -->
41      <!-- buttons for submitting and clearing the -->
42      <!-- form's contents -->
43      <input type = "submit" value = "Submit" />
44      <input type = "reset" value = "Clear" />
45  </p>
46  </form>
47  </body>
48 </html>

```

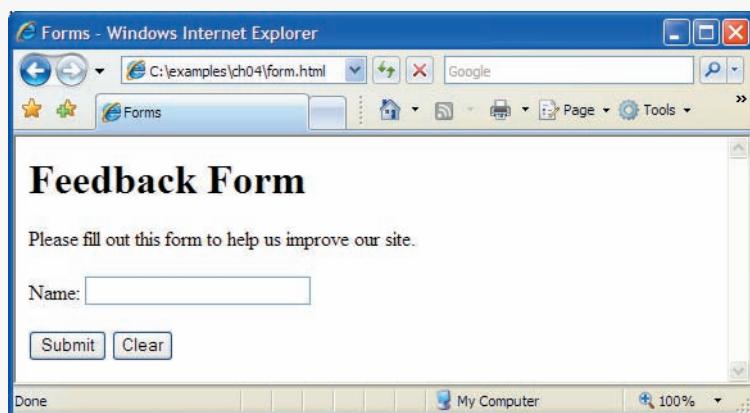


Fig. 4.12 | Form with hidden fields and a text box. (Part 2 of 2.)

The **action** attribute in the `<form>` tag in line 21 specifies the URL of a script on the web server that will be invoked to process the form's data. Since we haven't introduced server-side programming yet, we leave this attribute empty for now.

Lines 25–44 define six `input` elements that specify data to provide to the script that processes the form (also called the **form handler**). There are several types of input elements. An input's type is determined by its **type attribute**. This form uses a `text` input, a `submit` input, a `reset` input and three `hidden` inputs.

The `text` input in lines 35–36 inserts a **text box** in the form. Users can type data in text boxes. The `label` element (lines 34–37) provides users with information about the `input` element's purpose. The `input` element's `size` attribute specifies the number of characters visible in the text box. Optional attribute `maxlength` limits the number of characters input into the text box—in this case, the user is not permitted to type more than 30 characters.



Look-and-Feel Observation 4.3

Include a `label` element for each form element to help users determine the purpose of each form element.

Two `input` elements in lines 43–44 create two buttons. The `submit` `input` element is a button. When the `submit` button is pressed, the user is sent to the location specified in the `form`'s `action` attribute. The `value` attribute sets the text displayed on the button. The `reset` `input` element allows a user to reset all `form` elements to their default values. The `value` attribute of the `reset` `input` element sets the text displayed on the button (the default value is **Reset** if you omit the `value` attribute).

The three `input` elements in lines 25–30 have the `type` attribute `hidden`, which allows you to send form data that is not input by a user. The three hidden inputs are an e-mail address to which the data will be sent, the e-mail's subject line and a URL for the browser to open after submission of the form. Two other `input` attributes are `name`, which identifies the `input` element, and `value`, which provides the value that will be sent (or posted) to the web server.



Good Programming Practice 4.6

Place hidden input elements at the beginning of a form, immediately after the opening `<form>` tag. This placement allows document authors to locate hidden input elements quickly.

Additional Form Elements

In the previous example, you saw basic elements of XHTML forms. Now that you know the general structure of a form, we introduce elements and attributes for creating more complex forms. Figure 4.13 contains a form that solicits user feedback about a website.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.13: form2.html -->
6 <!-- Form using a variety of components. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>More Forms</title>
10    </head>
11
12   <body>
13     <h1>Feedback Form</h1>

```

Fig. 4.13 | Form using a variety of components. (Part 1 of 4.)

```
14 <p>Please fill out this form to help  
15 us improve our site.</p>  
16  
17 <form method = "post" action = "">  
18     <p>  
19         <input type = "hidden" name = "recipient"  
20             value = "deitel@deitel.com" />  
21         <input type = "hidden" name = "subject"  
22             value = "Feedback Form" />  
23         <input type = "hidden" name = "redirect"  
24             value = "main.html" />  
25     </p>  
26  
27     <p><label>Name:  
28         <input name = "name" type = "text" size = "25" />  
29     </label></p>  
30  
31     <!-- <textarea> creates a multiline textbox -->  
32     <p><label>Comments:<br />  
33         <textarea name = "comments"  
34             rows = "4" cols = "36">Enter comments here.</textarea>  
35     </label></p>  
36  
37     <!-- <input type = "password"> inserts a -->  
38     <!-- textbox whose display is masked with -->  
39     <!-- asterisk characters -->  
40     <p><label>E-mail Address:  
41         <input name = "email" type = "password" size = "25" />  
42     </label></p>  
43  
44     <p>  
45         <strong>Things you liked:</strong><br />  
46  
47         <label>Site design  
48             <input name = "thingsliked" type = "checkbox"  
49                 value = "Design" /></label>  
50         <label>Links  
51             <input name = "thingsliked" type = "checkbox"  
52                 value = "Links" /></label>  
53         <label>Ease of use  
54             <input name = "thingsliked" type = "checkbox"  
55                 value = "Ease" /></label>  
56         <label>Images  
57             <input name = "thingsliked" type = "checkbox"  
58                 value = "Images" /></label>  
59         <label>Source code  
60             <input name = "thingsliked" type = "checkbox"  
61                 value = "Code" /></label>  
62     </p>  
63  
64     <!-- <input type = "radio" /> creates a radio -->  
65     <!-- button. The difference between radio buttons -->
```

Fig. 4.13 | Form using a variety of components. (Part 2 of 4.)

```
66      <!-- and checkboxes is that only one radio button -->
67      <!-- in a group can be selected. -->
68      <p>
69          <strong>How did you get to our site?</strong><br />
70
71          <label>Search engine
72              <input name = "howtosite" type = "radio"
73                  value = "search engine" checked = "checked" /></label>
74          <label>Links from another site
75              <input name = "howtosite" type = "radio"
76                  value = "link" /></label>
77          <label>Deitel.com Website
78              <input name = "howtosite" type = "radio"
79                  value = "deitel.com" /></label>
80          <label>Reference in a book
81              <input name = "howtosite" type = "radio"
82                  value = "book" /></label>
83          <label>Other
84              <input name = "howtosite" type = "radio"
85                  value = "other" /></label>
86      </p>
87
88      <p>
89          <label>Rate our site:
90
91              <!-- the <select> tag presents a drop-down -->
92              <!-- List with choices indicated by the -->
93              <!-- <option> tags -->
94              <select name = "rating">
95                  <option selected = "selected">Amazing</option>
96                  <option>10</option>
97                  <option>9</option>
98                  <option>8</option>
99                  <option>7</option>
100                 <option>6</option>
101                 <option>5</option>
102                 <option>4</option>
103                 <option>3</option>
104                 <option>2</option>
105                 <option>1</option>
106                 <option>Awful</option>
107             </select>
108         </label>
109     </p>
110
111     <p>
112         <input type = "submit" value = "Submit" />
113         <input type = "reset" value = "Clear" />
114     </p>
115     </form>
116 </body>
117 </html>
```

Fig. 4.13 | Form using a variety of components. (Part 3 of 4.)

Feedback Form

Please fill out this form to help us improve our site.

Name:

Comments:
Enter comments here.

E-mail Address:

Things you liked:

Site design Links Ease of use Images Source code

How did you get to our site?:

Search engine Links from another site Deitel.com Website Reference in a book Other

Rate our site: Amazing Amazing

Submit 10
9
8
7
6
5
4
3
2
1

Done Awful My Computer 100%

Fig. 4.13 | Form using a variety of components. (Part 4 of 4.)

In line 32, we introduce the **br** element, which most browsers render as a =line break. Any markup or text following a **br** element is rendered on the next line. Like the **img** element, **br** is an example of an empty element terminated with a forward slash. We add a space before the forward slash to enhance readability.

The **textarea** element (lines 33–34) inserts a multiline text box, called a **text area**, into the form. The number of rows is specified with the **rows** attribute, and the number of columns (i.e., characters per line) is specified with the **cols** attribute. In this example, the **textarea** is four rows high and 36 characters wide. To display default text in the text area, place the text between the **<textarea>** and **</textarea>** tags. Default text can be specified in other **input** types, such as text boxes, by using the **value** attribute.

The **password** input in line 41 inserts a password box with the specified **size** (maximum number of characters allowed). A password box allows users to enter sensitive information, such as credit card numbers and passwords, by “masking” the information input with asterisks (*). The actual value input is sent to the web server, not the characters that mask the input.

Lines 47–61 introduce the **checkbox** form element. Checkboxes enable users to select from a set of options. When a user selects a checkbox, a check mark appears in the checkbox. Otherwise, the checkbox remains empty. Each “**checkbox**” **input** creates a new

checkbox. Checkboxes can be used individually or in groups. Checkboxes that belong to a group are assigned the same name (in this case, "thingsliked").



Common Programming Error 4.5

When your form has several checkboxes with the same name, you must make sure that they have different values, or the scripts running on the web server will not be able to distinguish them.

After the checkboxes, we present two more ways to allow the user to make choices. In this example, we introduce two new input types. The first type is the **radio button** (lines 71–85) specified with type "**radio**". Radio buttons are similar to checkboxes, except that only one radio button in a group of radio buttons may be selected at any time. The radio buttons in a group all have the same name attributes and are distinguished by their different value attributes. The attribute–value pair checked = "checked" (line 73) indicates which radio button, if any, is selected initially. The checked attribute also applies to checkboxes.



Common Programming Error 4.6

Not setting the name attributes of the radio buttons in a form to the same name is a logic error because it lets the user select all of them at the same time.

The **select** element (lines 94–107) provides a drop-down list from which the user can select an item. The name attribute identifies the drop-down list. The **option** elements (lines 95–106) add items to the drop-down list. The option element's **selected** attribute specifies which item initially is displayed as the selected item in the select element. If no option element is marked as **selected**, the browser selects the first option by default.

4.12 Internal Linking

Earlier in the chapter, we discussed how to hyperlink one web page to another. Figure 4.14 introduces **internal linking**—a mechanism that enables the user to jump between locations in the same document. Internal linking is useful for long documents that contain many sections. Clicking an internal link enables users to find a section without scrolling through the entire document.

Line 14 contains a tag with the **id** attribute (set to "features") for an internal hyperlink. To link to a tag with this attribute inside the same web page, the **href** attribute of an anchor element includes the **id** attribute value preceded by a pound sign (as in #features). Line 56 contains a hyperlink with the **id** features as its target. Selecting this hyperlink in a web browser scrolls the browser window to the **h1** tag in line 14. Note that you may have to resize your browser to a small window and scroll down before clicking the link to see the browser scroll to the **h1** element.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.14: internal.html -->
6 <!-- Internal hyperlinks to make pages more navigable. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
```

Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 1 of 3.)

```
8 <head>
9   <title>Internal Links</title>
10 </head>
11
12 <body>
13   <!-- id attribute creates an internal hyperlink destination -->
14   <h1 id = "features">The Best Features of the Internet</h1>
15
16   <!-- an internal link's address is "#id" -->
17   <p><a href = "#bugs">Go to <em>Favorite Bugs</em></a></p>
18
19   <ul>
20     <li>You can meet people from countries
21       around the world.</li>
22     <li>You have access to new media as it becomes public:
23       <ul>
24         <li>New games</li>
25         <li>New applications
26           <ul>
27             <li>For Business</li>
28             <li>For Pleasure</li>
29           </ul>
30         </li>
31
32         <li>Around the clock news</li>
33         <li>Search Engines</li>
34         <li>Shopping</li>
35         <li>Programming
36           <ul>
37             <li>XHTML</li>
38             <li>Java</li>
39             <li>Dynamic HTML</li>
40             <li>Scripts</li>
41             <li>New languages</li>
42           </ul>
43         </li>
44       </ul>
45     </li>
46
47     <li>Links</li>
48     <li>Keeping in touch with old friends</li>
49     <li>It is the technology of the future!</li>
50   </ul>
51
52   <!-- id attribute creates an internal hyperlink destination -->
53   <h1 id = "bugs">My 3 Favorite Bugs</h1>
54   <p>
55     <!-- internal hyperlink to features -->
56     <a href = "#features">Go to <em>Favorite Features</em></a>
57   </p>
58   <ol>
59     <li>Fire Fly</li>
60     <li>Gal Ant</li>
```

Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 2 of 3.)

```

61      <li>Roman Tic</li>
62    </ol>
63  </body>
64 </html>

```

The figure consists of two screenshots of Microsoft Internet Explorer. Both screenshots show a page titled "The Best Features of the Internet".

Screenshot 1: This screenshot shows a bulleted list of features. An internal hyperlink "Go to Favorite Bugs" is located above the list. The list includes:

- You can meet people from countries around the world.
- You have access to new media as it becomes public:
 - New games
 - New applications
 - For Business
 - For Pleasure
 - Around the clock news
 - Search Engines
 - Shopping

Screenshot 2: This screenshot shows a different bulleted list. The internal hyperlink "Go to Favorite Features" is located above the list. The list includes:

- Scripts
- New languages
- Links
- Keeping in touch with old friends
- It is the technology of the future!

My 3 Favorite Bugs

[Go to Favorite Features](#)

1. Fire Fly
2. Gal Ant
3. Roman Tic

Fig. 4.14 | Internal hyperlinks to make pages more navigable. (Part 3 of 3.)



Look-and-Feel Observation 4.4

Internal hyperlinks are useful in XHTML documents that contain large amounts of information. Internal links to different parts of the page make it easier for users to navigate the page—they do not have to scroll to find the section they want.

Although not demonstrated in this example, a hyperlink can specify an internal link in another document by specifying the document name followed by a pound sign and the id value, as in:

```
href = "filename.html#id"
```

For example, to link to a tag with the id attribute called booklist in books.html, href is assigned "books.html#booklist". You can also send the browser to an internal link on another website by appending the pound sign and id value of an element to any URL, as in:

```
href = "URL/filename.html#id"
```

4.13 meta Elements

Search engines help people find websites. They usually catalog sites by following links from page to page (often known as spidering or crawling) and saving identification and classification information for each page. One way that search engines catalog pages is by reading the content in each page's **meta** elements, which specify information about a document.

Two important attributes of the **meta** element are **name**, which identifies the type of **meta** element, and **content**, which provides the information search engines use to catalog pages. Figure 4.15 introduces the **meta** element.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 4.15: meta.html -->
6 <!-- meta elements provide keywords and a description of a page. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Welcome</title>
10
11    <!-- <meta> tags provide search engines with -->
12    <!-- information used to catalog a site -->
13    <meta name = "keywords" content = "web page, design,
14      XHTML, tutorial, personal, help, index, form,
15      contact, feedback, list, links, deitel" />
16    <meta name = "description" content = "This website will
17      help you learn the basics of XHTML and web page design
18      through the use of interactive examples and
19      instruction." />
20  </head>
21  <body>
22    <h1>Welcome to Our Website!</h1>
23
24    <p>We have designed this site to teach about the wonders
25    of <strong><em>XHTML</em></strong>. <em>XHTML</em> is
26    better equipped than <em>HTML</em> to represent complex
27    data on the Internet. <em>XHTML</em> takes advantage of
28    XML's strict syntax to ensure well-formedness. Soon you

```

Fig. 4.15 | meta elements provide keywords and a description of a page. (Part I of 2.)

```

29      will know about many of the great features of
30      <em>XHTML.</em></p>
31
32      <p>Have Fun With the Site!</p>
33      </body>
34  </html>

```



Fig. 4.15 | meta elements provide keywords and a description of a page. (Part 2 of 2.)

Lines 13–15 demonstrate a "keywords" meta element. The content attribute of such a meta element provides search engines with a list of words that describe a page. These words are compared with words in search requests. Thus, including meta elements and their content information can draw more viewers to your site.

Lines 16–19 demonstrate a "description" meta element. The content attribute of such a meta element provides a three- to four-line description of a site, written in sentence form. Search engines also use this description to catalog your site and sometimes display this information as part of the search results. Note that this use of the meta element is one of many methods of search engine optimization (SEO). For more information on SEO, visit Deitel's SEO Resource Center at www.deitel.com/searchengineoptimization.



Software Engineering Observation 4.1

meta elements are not visible to users and must be placed inside the head section of your XHTML document. If meta elements are not placed in this section, they will not be read by search engines.

4.14 Wrap-Up

This chapter introduced XHTML and explored many of its features. We discussed the basics of XHTML as well as marking up information in tables, creating forms for gathering user input, linking to sections in the same document and using `<meta>` tags. In Chapter 5, we build on the XHTML introduced in this chapter by discussing how to make web pages more visually appealing with Cascading Style Sheets.

4.15 Web Resources

www.deitel.com/xhtml

Visit our online XHTML Resource Center for links to some of the best XHTML information on the web. There you'll find categorized links to introductions, tutorials, books, blogs, forums, sample chapters, and more. Also check out links about XHTML 2, the upcoming version of the XHTML standard.

Summary

Section 4.1 Introduction

- XHTML (Extensible HyperText Markup Language) is a markup language for creating web pages.
- XHTML is based on HTML (HyperText Markup Language)—a legacy technology of the World Wide Web Consortium (W3C).
- XHTML 1.0 allows only a document's content and structure to appear in a valid XHTML document, and not its formatting.
- Formatting is specified with Cascading Style Sheets

Section 4.2 Editing XHTML

- A machine that runs a specialized piece of software called a web server stores XHTML documents.

Section 4.3 First XHTML Example

- In XHTML, text is marked up with elements delimited by tags that are names contained in pairs of angle brackets. Some elements may contain attributes that provide additional information about the element.
- Every XHTML document contains a start `<html>` tag and an end `</html>` tag.
- Comments in XHTML always begin with `<!--` and end with `-->`. The browser ignores all text inside a comment.
- Every XHTML document contains a `head` element, which generally contains information, such as a title, and a `body` element, which contains the page content. Information in the `head` element generally is not rendered in the display window but may be made available to the user through other means.
- The `title` element names a web page. The title usually appears in the colored bar (called the title bar) at the top of the browser window and also appears as the text identifying a page when users add your page to their list of **Favorites** or **Bookmarks**.
- The `body` of an XHTML document is the area in which the document's content is placed. The content may include text and tags.
- All text placed between the `<p>` and `</p>` tags forms one paragraph.

Section 4.4 W3C XHTML Validation Service

- XHTML documents that are syntactically correct are guaranteed to render properly. XHTML documents that contain syntax errors may not display properly.
- Validation services (e.g., validator.w3.org) ensure that an XHTML document is syntactically correct.

Section 4.5 Headings

- XHTML provides six headings (`h1` through `h6`) for specifying the relative importance of information. Heading element `h1` is considered the most significant heading and is rendered in a larger font than the other five headings. Each successive heading element (i.e., `h2`, `h3`, etc.) is rendered in a progressively smaller font.

Section 4.6 Linking

- Web browsers typically underline text hyperlinks and color them blue by default.
- The `strong` element typically causes the browser to render text in a bold font.
- Users can insert links with the `a` (anchor) element. The most important attribute for the `a` element is `href`, which specifies the resource (e.g., page, file, e-mail address) being linked.
- Anchors can link to an e-mail address using a `mailto:` URL. When someone clicks this type of anchored link, most browsers launch the default e-mail program (e.g., Outlook Express) to initiate an e-mail message addressed to the linked address.

Section 4.7 Images

- The `img` element's `src` attribute specifies an image's location. In a valid XHTML document every `img` element must have an `alt` attribute, which contains text that is displayed if the client cannot render the image.
- The `alt` attribute makes web pages more accessible to users with disabilities, especially those with vision impairments.
- Some XHTML elements are empty elements that contain only attributes and do not mark up text. Empty elements (e.g., `img`) must be terminated, either by using the forward slash character (/) or by explicitly writing an end tag.

Section 4.8 Special Characters and Horizontal Rules

- XHTML provides special characters or entity references (in the form &code;) for representing characters that cannot be rendered otherwise.
- Special character codes can be either word abbreviations or numbers, decimal or hexadecimal.
- Most browsers render a horizontal rule, indicated by the `<hr />` tag, as a horizontal line. The `hr` element also inserts a line break above and below the horizontal line.

Section 4.9 Lists

- The unordered list element `ul` creates a list in which each item in the list begins with a bullet symbol (called a disc). Each entry in an unordered list is an `li` (list item) element. Most web browsers render these elements with a line break and a bullet symbol at the beginning of the line.
- The ordered list element `ol` creates a list in which each item begins with a number.
- Lists may be nested to represent hierarchical data relationships.

Section 4.10 Tables

- XHTML tables are used to mark up tabular data. The `table` element defines an XHTML table.
- Element `summary` summarizes the table's contents and is used by speech devices to make the table more accessible to users with visual impairments.
- Element `caption` describes the table's content. The text inside the `<caption>` tag is rendered above the table in most browsers.
- A table can be split into three distinct sections: head (`thead`), body (`tbody`) and foot (`tfoot`). The head section contains such information as table titles and column headers. The table body contains the primary table data. The table foot contains such information as footnotes.

- Element `tr`, or table row, defines individual table rows. Element `th` defines a header cell. Other data in a row is defined with `td`, or table data, elements.
- You can merge data cells with the `rowspan` and `colspan` attributes. The values assigned to these attributes specify the number of rows or columns occupied by the cell. These attributes can be placed inside any data cell or table header cell.

Section 4.11 Forms

- XHTML provides forms for collecting information from users. Forms contain visual components, such as buttons, that users interact with. Forms may also contain nonvisual components, called hidden inputs, which are used to store any data that needs to be sent to the server, but is not entered by the user.
- A form begins with the `form` element. Attribute `method` specifies how the form's data is sent to the web server.
- The `action` attribute of the `form` element specifies the script to which the `form` data will be sent.
- The "text" input inserts a text box into the form. Text boxes allow the user to input data.
- The `input` element's `size` attribute specifies the number of characters visible in the `input` element. Optional attribute `maxlength` limits the number of characters input into a text box.
- The "submit" input submits the data entered in the form to the web server for processing. Most web browsers create a button that submits the form data when clicked. The "reset" input allows a user to reset all `form` elements to their default values.
- The `textarea` element inserts a multiline text box, called a text area, into a form. The number of rows in the text area is specified with the `rows` attribute, and the number of columns (i.e., characters per line) is specified with the `cols` attribute.
- The "password" input inserts a password box into a form. A password box allows users to enter sensitive information, such as credit card numbers and passwords, by "masking" the information input with another character. Asterisks are usually the masking character used for password boxes. The actual value input is sent to the web server, not the asterisks that mask the input.
- The checkbox input allows the user to make a selection. When the checkbox is selected, a check mark appears in the checkbox. Otherwise, the checkbox is empty. Checkboxes can be used individually and in groups. Checkboxes that are part of the same group have the same `name`.
- The `br` element causes most browsers to render a line break. Any markup or text following a `br` element is rendered on the next line.
- A radio button is similar in function and use to a checkbox, except that only one radio button in a group can be selected at any time. All radio buttons in a group have the same `name` attribute but different `value` attributes.
- The `select` input provides a drop-down list of items. The `name` attribute identifies the drop-down list. The `option` element adds items to the drop-down list.

Section 4.12 Internal Linking

- The `a` tag can be used to link to another section of the same document by specifying the element's `id` as the link's `href`.
- To link internally to an element with its `id` attribute set, use the syntax `#id`.

Section 4.13 meta Elements

- One way that search engines catalog pages is by reading the `meta` element's contents. Two important attributes of the `meta` element are `name`, which identifies the type of `meta` element, and `content`, which provides information a search engine uses to catalog a page.

- The `content` attribute of a `keywords` `meta` element provides search engines with a list of words that describe a page. These words are compared with words in search requests.
- The `content` attribute of a `description` `meta` element provides a three- to four-line description of a site, written in sentence form. Search engines also use this description to catalog your site and sometimes display this information as part of the search results.

Terminology

<!--...--> (XHTML comment)	hidden input element
a element (<a>...)	hr element (horizontal rule)
accessible web pages	href attribute (a)
action attribute (form)	.htm (XHTML filename extension)
alt attribute (img)	<html> tag
&#amp; (& special character)	.html (XHTML filename extension)
anchor	hyperlink
angle brackets (< >)	img element
attribute	input element
body element	internal linking
border attribute (table)	level of nesting
br element (line break)	li element (list item)
browser request	line break
caption element	link
Cascading Style Sheets (CSS)	linked document
character entity reference	list item
checkbox	< (< special character)
checked attribute (input)	mailto: URL
cols attribute (textarea)	markup language
colspan attribute (th, td)	maxlength attribute (input)
comment in XHTML	meta element
© (© special character)	method attribute (form)
data cells	name attribute
database	nested list
debugging	nested tag
del element	numeric character reference
element	ol element (ordered list)
em element	option element
e-mail anchor	p element (paragraph)
empty element	password box
end tag	pixel
Extensible HyperText Markup Language (XHTML)	post request type
form	presentation of a document
form element	protocol
form handler	radio input
get request type	reset input
head element	resources
header cell	rows attribute (textarea)
heading	rowspan attribute (th, tr)
h1 through h6 (heading elements)	script
height attribute (img)	select element
hexadecimal code	selected attribute (option)
	size attribute (input)

source code	tr element (table row)
special character	type attribute (<code>input</code>)
speech synthesizer	ul element (unordered list)
<code>src</code> attribute (<code>img</code>)	valid document
start tag	validation service
strong element	<code>value</code> attribute (<code>input</code>)
sub element	value of an attribute
<code>submit</code> input	web page
subscript	web server
superscript	<code>width</code> attribute (<code>img</code>)
table element	World Wide Web (WWW)
tag	World Wide Web Consortium (W3C)
<code>tbody</code> element	XHTML (Extensible HyperText
<code>td</code> element	Markup Language)
text editor	XHTML comment
text-based browser	XHTML document
<code>textarea</code>	XHTML form
<code>textarea</code> element	XHTML markup
<code>tfoot</code> element (table foot)	XHTML tag
<code>thead</code> element (table head)	XML declaration
<code>title</code> element	<code>xmlns</code> attribute

Self-Review Exercises

4.1 State whether each of the following is *true* or *false*. If *false*, explain why.

- a) An ordered list cannot be nested inside an unordered list.
- b) XHTML is an acronym for XML HTML.
- c) Element `br` represents a line break.
- d) Hyperlinks are denoted by `link` elements.
- e) The width of all data cells in a table must be the same.
- f) You are limited to a maximum of 100 internal links per page.

4.2 Fill in the blanks in each of the following:

- a) The _____ element inserts a horizontal rule.
- b) A superscript is marked up using element _____ and a subscript is marked up using element _____.
- c) The least important heading element is _____ and the most important heading element is _____.
- d) Element _____ marks up an unordered list.
- e) Element _____ marks up a paragraph.
- f) The _____ attribute in an `input` element inserts a button that, when clicked, clears the contents of the form.
- g) The _____ element marks up a table row.
- h) _____ are usually used as masking characters in a password box.

Answers to Self-Review Exercises

4.1 a) False. An ordered list can be nested inside an unordered list and vice versa. b) False. XHTML is an acronym for Extensible HyperText Markup Language. c) True. d) False. Hyperlinks are denoted by `a` elements. e) False. You can specify the width of any column, either in pixels or as a percentage of the table width. f) False. You can have an unlimited number of internal links.

4.2 a) `hr`. b) `sup`, `sub`. c) `h6`, `h1`. d) `ul`. e) `p`. f) `type = "reset"`. g) `tr`. h) Asterisks.

Exercises

- 4.3** Use XHTML to create a document that contains the following text:

```
Internet and World Wide Web How to Program: Fourth Edition
Welcome to the world of Internet programming. We have provided
topical coverage for many Internet-related topics.
```

Use `h1` for the title (the first line of text), `p` for text (the second and third lines of text) and `sub` for each word that begins with a capital letter (except the title). Insert a horizontal rule between the `h1` element and the `p` element. Open your new document in a web browser to view the marked-up document.

- 4.4** Why is the following markup invalid?

```
<p>Here is some text...
<hr />
<p>And some more text...</p>
```

- 4.5** Why is the following markup invalid?

```
<p>Here is some text...<br>
And some more text...</p>
```

- 4.6** An image named `deitel.gif` is 200 pixels wide and 150 pixels high. Write an XHTML statement using the `width` and `height` attributes of the `img` element to perform each of the following transformations:

- Increase the size of the image by 100 percent.
- Increase the size of the image by 50 percent.
- Change the width-to-height ratio to 2:1, keeping the `width` attained in part (a).

- 4.7** Create a link to each of the following:

- The file `index.html`, located in the `files` directory.
- The file `index.html`, located in the `text` subdirectory of the `files` directory.
- The file `index.html`, located in the other directory in your parent directory.
[Hint: ... signifies parent directory.]
- The President's e-mail address (`president@whitehouse.gov`).
- The file named `README` in the `pub` directory of `ftp.cdrom.com`. [Hint: Use `ftp://`.]

- 4.8** Create an XHTML document containing three ordered lists: ice cream, soft serve and frozen yogurt. Each ordered list should contain a nested, unordered list of your favorite flavors. Provide a minimum of three flavors in each unordered list.

- 4.9** Create an XHTML document that uses an image as an e-mail link. Use attribute `alt` to provide a description of the image and link.

- 4.10** Create an XHTML document that contains links to your favorite websites. Your page should contain the heading "My Favorite Web Sites."

- 4.11** Create an XHTML document that contains an unordered list with links to all the examples presented in this chapter. [Hint: Place all the chapter examples in one directory.]

- 4.12** Identify each of the following as either an element or an attribute:

- `html`
- `width`
- `href`
- `br`
- `h3`
- `a`
- `src`

4.13 State which of the following statements are *true* and which are *false*. If *false*, explain why.

- A valid XHTML document can contain uppercase letters in element names.
- Tags need not be closed in a valid XHTML document.
- XHTML documents can have the file extension .htm.
- Valid XHTML documents can contain tags that overlap.
- <ess; is the character entity reference for the less-than (<) character.
- In a valid XHTML document, can be nested inside either or tags.

4.14 Fill in the blanks in each of the following:

- XHTML comments begin with <!-- and end with _____.
- In XHTML, attribute values must be enclosed in _____.
- _____ is the character entity reference for an ampersand.
- Element _____ can be used to bold text.

4.15 Categorize each of the following as an element or an attribute:

- width
- td
- th
- name
- select
- type

4.16 Create the XHTML markup that produces the table shown in Fig. 4.16. Use and tags as necessary. The image (camel.gif) is included in the Chapter 4 examples directory that can be downloaded from <http://www.deitel.com/books/iw3htp4/>.

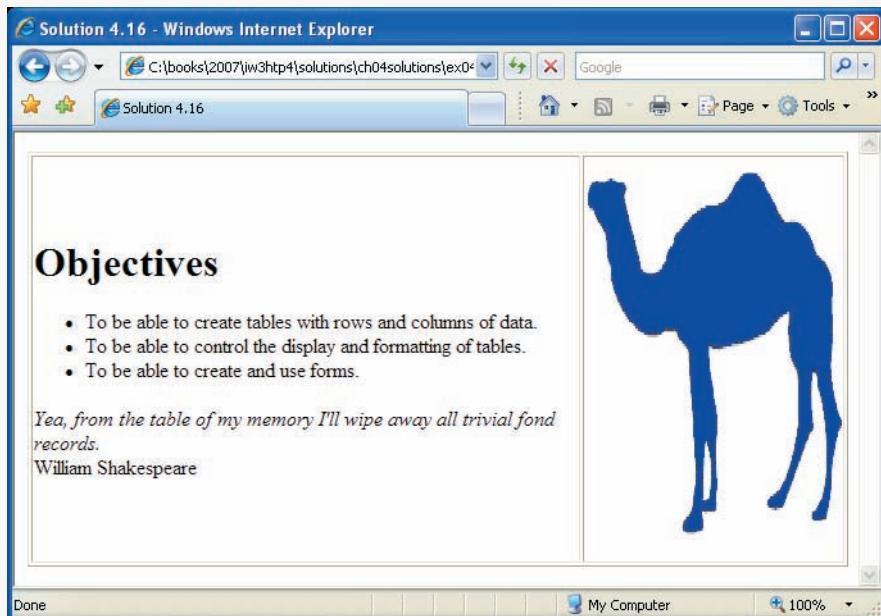


Fig. 4.16 | XHTML table for Exercise 4.16.

- 4.17** Write an XHTML document that produces the table shown in Fig. 4.17.

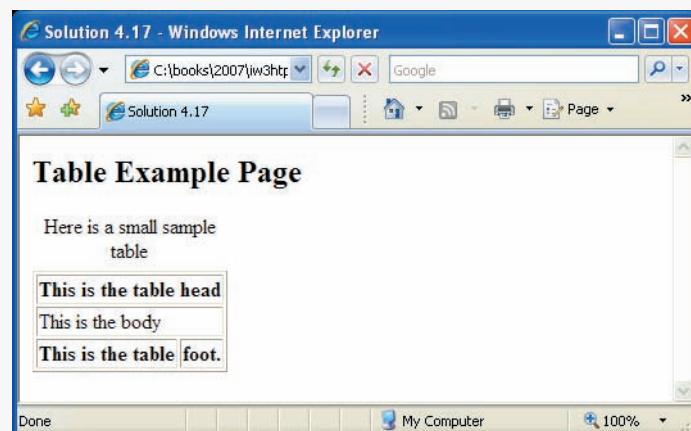
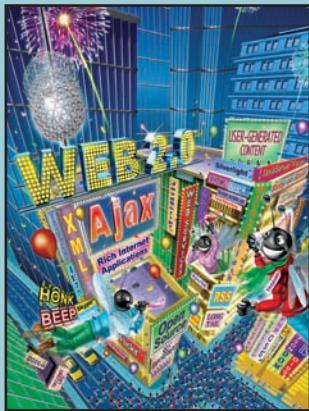


Fig. 4.17 | XHTML table for Exercise 4.17.

4.18 A local university has asked you to create an XHTML document that allows prospective students to provide feedback about their campus visit. Your XHTML document should contain a form with text boxes for a name, address and e-mail. Provide checkboxes that allow prospective students to indicate what they liked most about the campus. The checkboxes should include: students, location, campus, atmosphere, dorm rooms and sports. Also, provide radio buttons that ask the prospective students how they became interested in the university. Options should include: friends, television, Internet and other. In addition, provide a text area for additional comments, a submit button and a reset button.

4.19 Create an XHTML document titled “How to Get Good Grades.” Use `<meta>` tags to include a series of keywords that describe your document.

5



*Fashions fade, style is
eternal.*

—Yves Saint Laurent

*A style does not go out of style
as long as it adapts itself to
its period. When there is an
incompatibility between the
style and a certain state of
mind, it is never the style
that triumphs.*

—Coco Chanel

*How liberating to work in
the margins, outside a
central perception.*

—Don DeLillo

*I've gradually risen from
lower-class background to
lower-class foreground.*

—Marvin Cohen

Cascading Style Sheets™ (CSS)

OBJECTIVES

In this chapter you will learn:

- To control the appearance of a website by creating style sheets.
- To use a style sheet to give all the pages of a website the same look and feel.
- To use the `class` attribute to apply styles.
- To specify the precise font, size, color and other properties of displayed text.
- To specify element backgrounds and colors.
- To understand the box model and how to control margins, borders and padding.
- To use style sheets to separate presentation from content.

Outline

- 5.1 Introduction
- 5.2 Inline Styles
- 5.3 Embedded Style Sheets
- 5.4 Conflicting Styles
- 5.5 Linking External Style Sheets
- 5.6 Positioning Elements
- 5.7 Backgrounds
- 5.8 Element Dimensions
- 5.9 Box Model and Text Flow
- 5.10 Media Types
- 5.11 Building a CSS Drop-Down Menu
- 5.12 User Style Sheets
- 5.13 CSS 3
- 5.14 Wrap-Up
- 5.15 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercise](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

5.1 Introduction

In Chapter 4, we introduced the Extensible HyperText Markup Language (XHTML) for marking up information to be rendered in a browser. In this chapter, we shift our focus to formatting and presenting information. To do this, we use a W3C technology called [Cascading Style Sheets™ \(CSS\)](#) that allows document authors to specify the presentation of elements on a web page (e.g., fonts, spacing, colors) separately from the structure of the document (section headers, body text, links, etc.). This [separation of structure from presentation](#) simplifies maintaining and modifying a web page.

XHTML was designed to specify the content and structure of a document. Though it has some attributes that control presentation, it is better not to mix presentation with content. If a website's presentation is determined entirely by a style sheet, a web designer can simply swap in a new style sheet to completely change the appearance of the site. CSS provides a way to apply style outside of XHTML, allowing the XHTML to dictate the content while the CSS dictates how it's presented.

As with XHTML, the W3C provides a CSS code validator located at jigsaw.w3.org/css-validator/. It is a good idea to validate all CSS code with this tool to make sure that your code is correct and works on as many browsers as possible.

CSS is a large topic. As such, we can introduce only the basic knowledge of CSS that you'll need to understand the examples and exercises in the rest of the book. For more CSS references and resources, check out our CSS Resource Center at www.deitel.com/css21.

The W3C's CSS specification is currently in its second major version, with a third in development. The current versions of most major browsers support much of the functionality in CSS 2. This allows programmers to make full use of its features. In this chapter, we introduce CSS, demonstrate some of the features introduced in CSS 2 and discuss some of the upcoming CSS 3 features. As you read this book, open each XHTML document in your web browser so you can view and interact with it in a web browser, as it was originally intended.

Remember that the examples in this book have been tested in Internet Explorer 7 and Firefox 2. The latest versions of many other browsers (e.g., Safari, Opera, Konqueror) should render this chapter's examples properly, but we have not tested them. Some examples in this chapter *will not work* in older browsers, such as Internet Explorer 6 and earlier. Make sure you have either Internet Explorer 7 (Windows only) or Firefox 2 (available for all major platforms) installed before running the examples in this chapter.

5.2 Inline Styles

You can declare document styles in several ways. This section presents **inline styles** that declare an individual element's format using the XHTML attribute **style**. Inline styles override any other styles applied using the techniques we discuss later in the chapter. Figure 5.1 applies inline styles to p elements to alter their font size and color.



Good Programming Practice 5.1

Inline styles do not truly separate presentation from content. To apply similar styles to multiple elements, use embedded style sheets or external style sheets, introduced later in this chapter.

The first inline style declaration appears in line 17. Attribute **style** specifies an element's style. Each **CSS property** (**font-size** in this case) is followed by a colon and a value. In line 17, we declare this particular p element to use 20-point font size.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.1: inline.html -->
6 <!-- Using inline styles -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Inline Styles</title>
10   </head>
11   <body>
12     <p>This text does not have any style applied to it.</p>
13
14     <!-- The style attribute allows you to declare -->
15     <!-- inline styles. Separate multiple style properties -->
16     <!-- with a semicolon. -->
17     <p style = "font-size: 20pt">This text has the
18       <em>font-size</em> style applied to it, making it 20pt.
19     </p>
20
21     <p style = "font-size: 20pt; color: #6666ff">
22       This text has the <em>font-size</em> and
23       <em>color</em> styles applied to it, making it
24       20pt. and light blue.</p>
25   </body>
26 </html>
```

Fig. 5.1 | Using inline styles. (Part 1 of 2.)

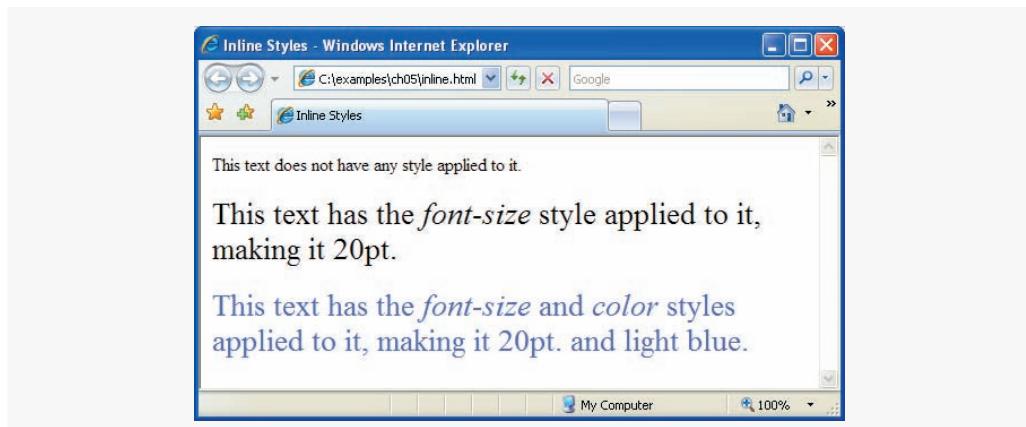


Fig. 5.1 | Using inline styles. (Part 2 of 2.)

Line 21 specifies the two properties, *font-size* and *color*, separated by a semicolon. In this line, we set the given paragraph's *color* to light blue, using the hexadecimal code #6666ff. Color names may be used in place of hexadecimal codes. We provide a list of hexadecimal color codes and color names in Appendix B, XHTML Colors.

5.3 Embedded Style Sheets

A second technique for using style sheets is **embedded style sheets**. Embedded style sheets enable you to embed an entire CSS document in an XHTML document's head section. To achieve this separation between the CSS code and the XHTML that it styles, we will use **CSS selectors**. Figure 5.2 creates an embedded style sheet containing four styles.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.2: embedded.html -->
6 <!-- Embedded style sheets. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Style Sheets</title>
10
11    <!-- this begins the style sheet section -->
12    <style type = "text/css">
13      em      { font-weight: bold;
14                    color: black }
15      h1      { font-family: tahoma, helvetica, sans-serif }
16      p       { font-size: 12pt;
17                    font-family: arial, sans-serif }
18      .special { color: #6666ff }
19    </style>
20  </head>
```

Fig. 5.2 | Embedded style sheets. (Part 1 of 2.)

```
21 <body>
22   <!-- this class attribute applies the .special style -->
23   <h1 class = "special">Deitel &amp; Associates, Inc.</h1>
24
25   <p>Deitel &amp; Associates, Inc. is an internationally
26   recognized corporate training and publishing organization
27   specializing in programming languages, Internet/World
28   Wide Web technology and object technology education.
29   The company provides courses on Java, C++, Visual Basic,
30   C#, C, Internet and World Wide Web programming, Object
31   Technology, and more.</p>
32
33   <h2>Clients</h2>
34   <p class = "special"> The company's clients include many
35   <em>Fortune 1000 companies</em>, government agencies,
36   branches of the military and business organizations.
37   Through its publishing partnership with Prentice Hall,
38   Deitel &amp; Associates, Inc. publishes leading-edge
39   programming textbooks, professional books, interactive
40   web-based multimedia Cyber Classrooms, satellite
41   courses and World Wide Web courses.</p>
42 </body>
43 </html>
```

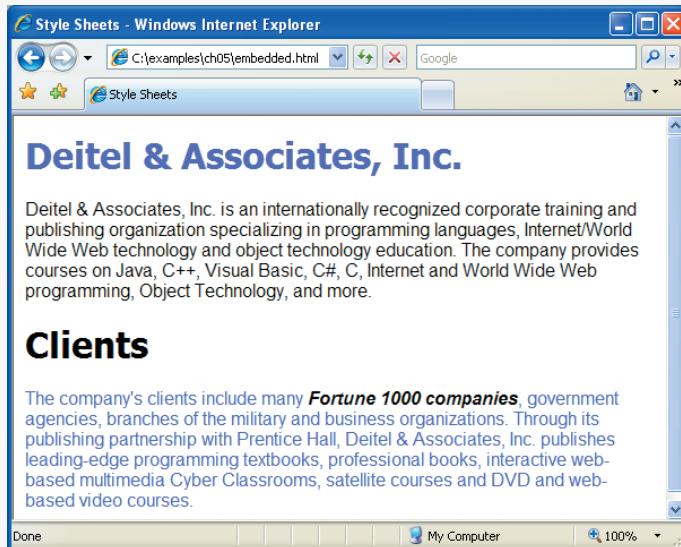


Fig. 5.2 | Embedded style sheets. (Part 2 of 2.)

The `style` element (lines 12–19) defines the embedded style sheet. Styles placed in the head apply to matching elements wherever they appear in the entire document. The `style` element's `type` attribute specifies the **Multipurpose Internet Mail Extensions (MIME) type** that describes a file's content. CSS documents use the MIME type `text/css`. Other MIME types include `image/gif` (for GIF images), `text/javascript` (for the JavaScript scripting language, which we discuss in Chapters 6–11), and more.

The body of the style sheet (lines 13–18) declares the **CSS rules** for the style sheet. A CSS selector determines which elements will be styled according to a rule. Our first rule begins with the selector `em` (line 13) to select all `em` elements in the document. The **font-weight** property in line 13 specifies the “boldness” of text. Possible values are `bold`, `normal` (the default), `bolder` (bolder than `bold` text) and `lighter` (lighter than `normal` text). Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., `100`, `200`, ..., `900`). Text specified as `normal` is equivalent to 400, and `bold` text is equivalent to 700. However, many systems do not have fonts that can scale with this level of precision, so using the values from 100 to 900 might not display the desired effect.

In this example, all `em` elements will be displayed in a bold font. We also apply styles to all `h1` (line 15) and `p` (lines 16–17) elements. The body of each rule is enclosed in curly braces (`{` and `}`).

Line 18 uses a new kind of selector to declare a **style class** named `special`. Style classes define styles that can be applied to any element. In this example, we declare class `special`, which sets `color` to `blue`. We can apply this style to any element type, whereas the other rules in this style sheet apply only to specific element types defined in the style sheet (i.e., `em`, `h1` or `p`). Style-class declarations are preceded by a period. We will discuss how to apply a style class momentarily.

CSS rules in embedded style sheets use the same syntax as inline styles; the property name is followed by a colon (`:`) and the value of the property. Multiple properties are separated by semicolons (`;`). In the rule for `em` elements, the `color` property specifies the color of the text, and property `font-weight` makes the font bold.

The **font-family** property (line 15) specifies the name of the font to use. Not all users have the same fonts installed on their computers, so CSS allows you to specify a comma-separated list of fonts to use for a particular style. The browser attempts to use the fonts in the order they appear in the list. It’s advisable to end a font list with a **generic font family name** in case the other fonts are not installed on the user’s computer. In this example, if the `tahoma` font is not found on the system, the browser will look for the `helvetica` font. If neither is found, the browser will display its default `sans-serif` font. Other generic font families include `serif` (e.g., `times new roman`, `Georgia`), `cursive` (e.g., `script`), `fantasy` (e.g., `critter`) and `monospace` (e.g., `courier`, `fixedsys`).

The **font-size** property (line 16) specifies a 12-point font. Other possible measurements in addition to `pt` (point) are introduced later in the chapter. Relative values—`xx-small`, `x-small`, `small`, `smaller`, `medium`, `large`, `larger`, `x-large` and `xx-large`—also can be used. Generally, relative values for `font-size` are preferred over point sizes because an author does not know the specific measurements of the display for each client. Relative `font-size` values permit more flexible viewing of web pages.

For example, a user may wish to view a web page on a handheld device with a small screen. Specifying an 18-point font size in a style sheet will prevent such a user from seeing more than one or two characters at a time. However, if a relative font size is specified, such as `large` or `larger`, the actual size is determined by the browser that displays the font. Using relative sizes also makes pages more accessible to users with disabilities. Users with impaired vision, for example, may configure their browser to use a larger default font, upon which all relative sizes are based. Text that the author specifies to be `smaller` than the main text still displays in a smaller size font, yet it is clearly visible to each user. Accessibility is an important consideration—in 1998, congress passed the Section 508 Amend-

ment to the Rehabilitation Act of 1973, mandating that websites of government agencies are required to be accessible to disabled users.

Line 23 uses the XHTML attribute `class` in an `h1` element to apply a style class—in this case class `special` (declared with the `.special` selector in the style sheet on line 18). When the browser renders the `h1` element, note that the text appears on screen with the properties of both an `h1` element (`arial` or `sans-serif` font defined in line 17) and the `.special` style class applied (the color `#6666ff` defined in line 18). Also notice that the browser still applies its own default style to the `h1` element—the header is still displayed in a large font size. Similarly, all `em` elements will still be italicized by the browser, but they will also be bold as a result of our style rule.

The formatting for the `p` element and the `.special` class is applied to the text in lines 34–41. In many cases, the styles applied to an element (the `parent` or `ancestor element`) also apply to the element’s nested elements (`child` or `descendant elements`). The `em` element nested in the `p` element in line 35 `inherits` the style from the `p` element (namely, the 12-point font size in line 16) but retains its italic style. In other words, styles defined for the paragraph and not defined for the `em` element is applied to the `em` element. Because multiple values of one property can be set or inherited on the same element, they must be reduced to one style per element before being rendered. We discuss the rules for resolving these conflicts in the next section.

5.4 Conflicting Styles

Styles may be defined by a `user`, an `author` or a `user agent` (e.g., a web browser). A user is a person viewing your web page, you are the author—the person who writes the document—and the user agent is the program used to render and display the document. Styles “cascade,” or flow together, such that the ultimate appearance of elements on a page results from combining styles defined in several ways. Styles defined by the user take precedence over styles defined by the user agent, and styles defined by authors take precedence over styles defined by the user.

Most styles defined for parent elements are also inherited by child (nested) elements. While it makes sense to inherit most styles, such as font properties, there are certain properties that we don’t want to be inherited. Consider for example the `background-image` property, which allows the programmer to set an image as the background of an element. If the `body` element is assigned a background image, we don’t want the same image to be in the background of every element in the body of our page. Instead, the `background-image` property of all child elements retains its default value of `none`. In this section, we discuss the rules for resolving conflicts between styles defined for elements and styles inherited from parent and ancestor elements.

Figure 5.2 presented an example of `inheritance` in which a child `em` element inherited the `font-size` property from its parent `p` element. However, in Fig. 5.2, the child `em` element had a `color` property that conflicted with (i.e., had a different value than) the `color` property of its parent `p` element. Properties defined for child and descendant elements have a greater `specificity` than properties defined for parent and ancestor elements. Conflicts are resolved in favor of properties with a higher specificity. In other words, the styles explicitly defined for a child element are more specific than the styles defined for the child’s parent element; therefore, the child’s styles take precedence. Figure 5.3 illustrates examples of inheritance and specificity.

Line 12 applies property `text-decoration` to all `a` elements whose `class` attribute is set to `nodec`. The `text-decoration` property applies `decorations` to text in an element. By default, browsers underline the text of an `a` (anchor) element. Here, we set the `text-decoration` property to `none` to indicate that the browser should not underline hyperlinks. Other possible values for `text-decoration` include `overline`, `line-through`, `underline` and `blink`. [Note: `blink` is not supported by Internet Explorer.] The `.nodec` appended to `a` is a more specific class selector; this style will apply only to `a` (anchor) elements that specify `nodec` in their `class` attribute.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.3: advanced.html -->
6 <!-- Inheritance in style sheets. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>More Styles</title>
10    <style type = "text/css">
11      body { font-family: arial, helvetica, sans-serif }
12      a.nodec { text-decoration: none }
13      a:hover { text-decoration: underline }
14      li em { font-weight: bold }
15      h1, em { text-decoration: underline }
16      ul { margin-left: 20px }
17      ul ul { font-size: .8em }
18    </style>
19  </head>
20  <body>
21    <h1>Shopping list for Monday:</h1>
22
23    <ul>
24      <li>Milk</li>
25      <li>Bread
26        <ul>
27          <li>White bread</li>
28          <li>Rye bread</li>
29          <li>Whole wheat bread</li>
30        </ul>
31      </li>
32      <li>Rice</li>
33      <li>Potatoes</li>
34      <li>Pizza <em>with mushrooms</em></li>
35    </ul>
36
37    <p><em>Go to the</em>
38      <a class = "nodec" href = "http://www.deitel.com">
39        Grocery store</a>
40    </p>
41  </body>
42 </html>
```

Fig. 5.3 | Inheritance in style sheets. (Part 1 of 2.)

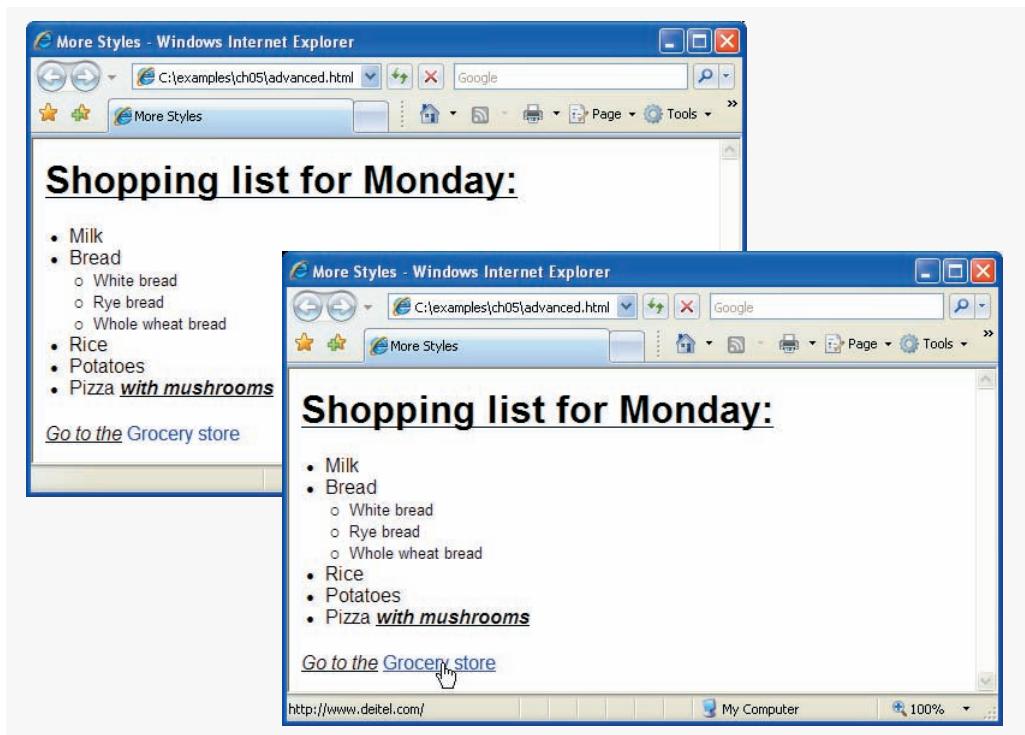


Fig. 5.3 | Inheritance in style sheets. (Part 2 of 2.)



Portability Tip 5.1

To ensure that your style sheets work in various web browsers, test them on all the client web browsers that will render documents using your styles, as well as using the W3C CSS Validator.

Line 13 specifies a style for `:hover`, which is a **pseudoclass**. Pseudoclasses give the author access to content not specifically declared in the document. The `:hover pseudoclass` is activated dynamically when the user moves the mouse cursor over an element. Note that pseudoclasses are separated by a colon (with no surrounding spaces) from the name of the element to which they are applied.



Common Programming Error 5.1

Including a space before or after the colon separating a pseudoclass from the name of the element to which it is applied is an error that prevents the pseudoclass from being applied properly.

Line 14 causes all `em` elements that are children of `li` elements to be bold. In the screen output of Fig. 5.3, note that `Go to the` (contained in an `em` element in line 37) does not appear bold, because the `em` element is not in an `li` element. However, the `em` element containing `with mushrooms` (line 34) is nested in an `li` element; therefore, it is formatted in bold. The syntax for applying rules to multiple elements is similar. In line 15, we separate the selectors with a comma to apply an underline style rule to all `h1` and all `em` elements.

Line 16 assigns a left margin of 20 pixels to all `ul` elements. We will discuss the `margin` properties in detail in Section 5.9. A pixel is a **relative-length measurement**—it varies in

size, based on screen resolution. Other relative lengths include `em` (the *M*-height of the font, which is usually set to the height of an uppercase *M*), `ex` (the *x*-height of the font, which is usually set to the height of a lowercase *x*) and percentages (e.g., `font-size: 50%`). To set an element to display text at 150 percent of its default text size, the author could use the syntax

```
font-size: 1.5em
```

Alternatively, you could use

```
font-size: 150%
```

Other units of measurement available in CSS are **absolute-length measurements**—i.e., units that do not vary in size based on the system. These units are `in` (inches), `cm` (centimeters), `mm` (millimeters), `pt` (points; 1 `pt` = 1/72 `in`) and `pc` (picas; 1 `pc` = 12 `pt`). Line 17 specifies that all nested unordered lists (`ul` elements that are descendants of `ul` elements) are to have font size `.8em`. [Note: When setting a style property that takes a measurement (e.g. `font-size`, `margin-left`), no units are necessary if the value is zero.]



Good Programming Practice 5.2

Whenever possible, use relative-length measurements. If you use absolute-length measurements, your document may not be readable on some client browsers (e.g., wireless phones).

5.5 Linking External Style Sheets

Style sheets are a convenient way to create a document with a uniform theme. With **external style sheets** (i.e., separate documents that contain only CSS rules), you can provide a uniform look and feel to an entire website. Different pages on a site can all use the same style sheet. When changes to the styles are required, the author needs to modify only a single CSS file to make style changes across the entire website. Note that while embedded style sheets separate content from presentation, both are still contained in a single file, preventing a web designer and a content author from working in parallel. External style sheets solve this problem by separating the content and style into separate files.



Software Engineering Observation 5.1

Always use an external style sheet when developing a website with multiple pages. External style sheets separate content from presentation, allowing for more consistent look-and-feel, more efficient development, and better performance.

Figure 5.4 presents an external style sheet. Lines 1–2 are **CSS comments**. Like XHTML comments, CSS comments describe the content of a CSS document. Comments may be placed in any type of CSS code (i.e., inline styles, embedded style sheets and external style sheets) and always start with `/*` and end with `*/`. Text between these delimiters is ignored by the browser.

```
1 /* Fig. 5.4: styles.css */
2 /* External stylesheet */
3
```

Fig. 5.4 | External style sheet. (Part 1 of 2.)

```

4 body      { font-family: arial, helvetica, sans-serif }
5 a.nodec   { text-decoration: none }
6 a:hover   { text-decoration: underline }
7
8 li em    { font-weight: bold }
9
10 h1, em  { text-decoration: underline }
11
12 ul       { margin-left: 20px }
13
14 ul ul   { font-size: .8em; }

```

Fig. 5.4 | External style sheet. (Part 2 of 2.)

Figure 5.5 contains an XHTML document that references the external style sheet in Fig. 5.4. Lines 10–11 (Fig. 5.5) show a `link` element that uses the `rel` attribute to specify a **relationship** between the current document and another document. In this case, we declare the linked document to be a **stylesheet** for this document. The `type` attribute specifies the MIME type of the related document as `text/css`. The `href` attribute provides the URL for the document containing the style sheet. In this case, `styles.css` is in the same directory as `external.html`.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.6: external.html -->
6 <!-- Linking an external style sheet. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Linking External Style Sheets</title>
10    <link rel = "stylesheet" type = "text/css"
11      href = "styles.css" />
12  </head>
13  <body>
14    <h1>Shopping list for <em>Monday</em>:</h1>
15
16    <ul>
17      <li>Milk</li>
18      <li>Bread
19        <ul>
20          <li>White bread</li>
21          <li>Rye bread</li>
22          <li>Whole wheat bread</li>
23        </ul>
24      </li>
25      <li>Rice</li>
26      <li>Potatoes</li>

```

Fig. 5.5 | Linking an external style sheet. (Part 1 of 2.)

```

27      <li>Pizza with mushrooms</em></li>
28  </ul>
29
30  <p><em>Go to the</em>
31      <a class = "nodec" href = "http://www.deitel.com">
32          Grocery store</a>
33      </p>
34  </body>
35 </html>

```

The figure shows two side-by-side windows of Microsoft Internet Explorer. Both windows have the title 'Linking External Style Sheets - Windows Internet Explorer' and the URL 'C:\examples\ch05\external.html'.

Left Window: This window displays the raw HTML code. It includes a heading 'Shopping list for Monday:' and an ordered list of items. A link 'Go to the Grocery store' is present at the bottom.

Right Window: This window shows the same content but with styles applied via an external CSS file. The heading is bolded, the list items have a black bullet point, and the link is underlined. The link 'Go to the Grocery store' is also styled.

Fig. 5.5 | Linking an external style sheet. (Part 2 of 2.)



Software Engineering Observation 5.2

External style sheets are reusable. Creating them once and reusing them reduces programming effort.



Performance Tip 5.1

Reusing external style sheets reduces load time and bandwidth usage on a server, since the style sheet can be downloaded once, stored by the web browser, and applied to all pages on a website.

5.6 Positioning Elements

Before CSS, controlling the positioning of elements in an XHTML document was difficult—the browser determined positioning. CSS introduced the **position** property and a capability called **absolute positioning**, which gives authors greater control over how document elements are displayed. Figure 5.6 demonstrates absolute positioning.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.6: positioning.html -->
6 <!-- Absolute positioning of elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Absolute Positioning</title>
10    <style type = "text/css">
11      .bgimg { position: absolute;
12        top: 0px;
13        left: 0px;
14        z-index: 1 }
15      .fgimg { position: absolute;
16        top: 25px;
17        left: 100px;
18        z-index: 2 }
19      .text { position: absolute;
20        top: 25px;
21        left: 100px;
22        z-index: 3;
23        font-size: 20pt;
24        font-family: tahoma, geneva, sans-serif }
25    </style>
26  </head>
27  <body>
28    <p><img src = "bgimg.gif" class = "bgimg"
29      alt = "First positioned image" /></p>
30
31    <p><img src = "fgimg.gif" class = "fgimg"
32      alt = "Second positioned image" /></p>
33
34    <p class = "text">Positioned Text</p>
35  </body>
36 </html>
```

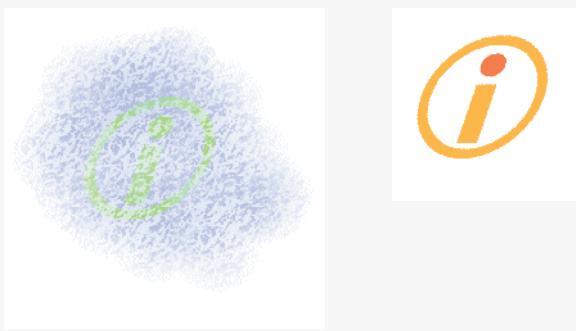


Fig. 5.6 | Absolute positioning of elements. (Part 1 of 2.)



Fig. 5.6 | Absolute positioning of elements. (Part 2 of 2.)

Normally, elements are positioned on the page in the order that they appear in the XHTML document. Lines 11–14 define a style called `bgiimg` for the first `img` element (`i.gif`) on the page. Specifying an element's position as `absolute` removes the element from the normal flow of elements on the page, instead positioning it according to the distance from the top, left, right or bottom margins of its **containing block-level element** (i.e., an element such as `body` or `p`). Here, we position the element to be 0 pixels away from both the top and left margins of its containing element. In line 28, this style is applied to the image, which is contained in a `p` element.

The `z-index` property allows you to layer overlapping elements properly. Elements that have higher `z-index` values are displayed in front of elements with lower `z-index` values. In this example, `i.gif` has the lowest `z-index` (1), so it displays in the background. The `.fgimg` CSS rule in lines 15–18 gives the circle image (`circle.gif`, in lines 31–32) a `z-index` of 2, so it displays in front of `i.gif`. The `p` element in line 34 (Positioned Text) is given a `z-index` of 3 in line 22, so it displays in front of the other two. If you do not specify a `z-index` or if elements have the same `z-index` value, the elements are placed from background to foreground in the order they are encountered in the document.

Absolute positioning is not the only way to specify page layout. Figure 5.7 demonstrates **relative positioning**, in which elements are positioned relative to other elements.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 5.7: positioning2.html -->
6  <!-- Relative positioning of elements. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Relative Positioning</title>
```

Fig. 5.7 | Relative positioning of elements. (Part 1 of 2.)

```
10 <style type = "text/css">
11   p          { font-size: 1.3em;
12             font-family: verdana, arial, sans-serif }
13   span        { color: red;
14             font-size: .6em;
15             height: 1em }
16   .super      { position: relative;
17             top: -1ex }
18   .sub        { position: relative;
19             bottom: -1ex }
20   .shiftleft  { position: relative;
21             left: -1ex }
22   .shiftright { position: relative;
23             right: -1ex }
24 </style>
25 </head>
26 <body>
27   <p>The text at the end of this sentence
28   <span class = "super">is in superscript</span>.</p>
29
30   <p>The text at the end of this sentence
31   <span class = "sub">is in subscript</span>.</p>
32
33   <p>The text at the end of this sentence
34   <span class = "shiftleft">is shifted left</span>.</p>
35
36   <p>The text at the end of this sentence
37   <span class = "shiftright">is shifted right</span>.</p>
38 </body>
39 </html>
```

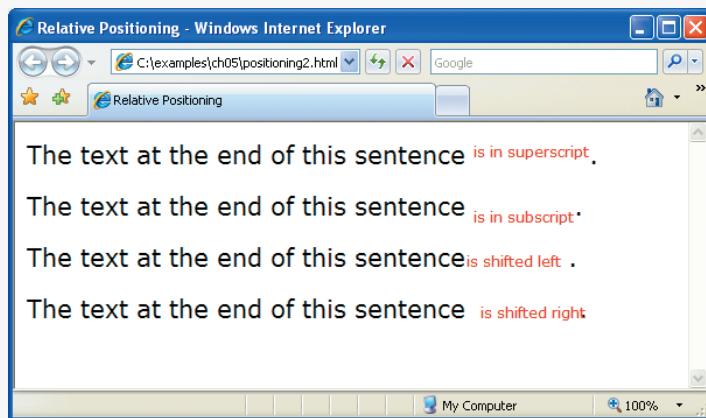


Fig. 5.7 | Relative positioning of elements. (Part 2 of 2.)

Setting the `position` property to `relative`, as in class `super` (lines 16–17), lays out the element on the page and offsets it by the specified `top`, `bottom`, `left` or `right` value. Unlike absolute positioning, relative positioning keeps elements in the general flow of elements on the page, so positioning is relative to other elements in the flow. Recall that ex

(line 17) is the *x*-height of a font, a relative-length measurement typically equal to the height of a lowercase *x*.



Common Programming Error 5.2

Because relative positioning keeps elements in the flow of text in your documents, be careful to avoid unintentionally overlapping text.

Inline and Block-Level Elements

We introduce the `span` element in line 28. Lines 13–15 define the CSS rule for all `span` elements. The height of the `span` determines how much vertical space the `span` will occupy. The `font-size` determines the size of the text inside the `span`.

Element `span` is a **grouping element**—it does not apply any inherent formatting to its contents. Its primary purpose is to apply CSS rules or `id` attributes to a section of text. Element `span` is an **inline-level element**—it applies formatting to text without changing the flow of the document. Examples of inline elements include `span`, `img`, `a`, `em` and `strong`. The `div` element is also a grouping element, but it is a **block-level element**. This means it is displayed on its own line and has a virtual box around it. Examples of block-level elements include `div`, `p` and heading elements (`h1` through `h6`). We'll discuss inline and block-level elements in more detail in Section 5.9.

5.7 Backgrounds

CSS provides control over the background of block-level elements. CSS can set a background color or add background images to XHTML elements. Figure 5.8 adds a corporate logo to the bottom-right corner of the document. This logo stays fixed in the corner even when the user scrolls up or down the screen.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.8: background.html -->
6 <!-- Adding background images and indentation. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Background Images</title>
10    <style type = "text/css">
11      body { background-image: url(logo.gif);
12             background-position: bottom right;
13             background-repeat: no-repeat;
14             background-attachment: fixed;
15             background-color: #eeeeee }
16      p { font-size: 18pt;
17           color: #1144AA;
18           text-indent: 1em;
19           font-family: arial, sans-serif; }
20      .dark { font-weight: bold }
21    </style>
22  </head>
```

Fig. 5.8 | Adding background images and indentation. (Part 1 of 2.)

```

23   <body>
24     <p>
25       This example uses the background-image,
26       background-position and background-attachment
27       styles to place the <span class = "dark">Deitel
28       & Associates, Inc.</span> logo in the bottom,
29       right corner of the page. Notice how the logo
30       stays in the proper position when you resize the
31       browser window. The background-color fills in where
32       there is no image.
33     </p>
34   </body>
35 </html>

```

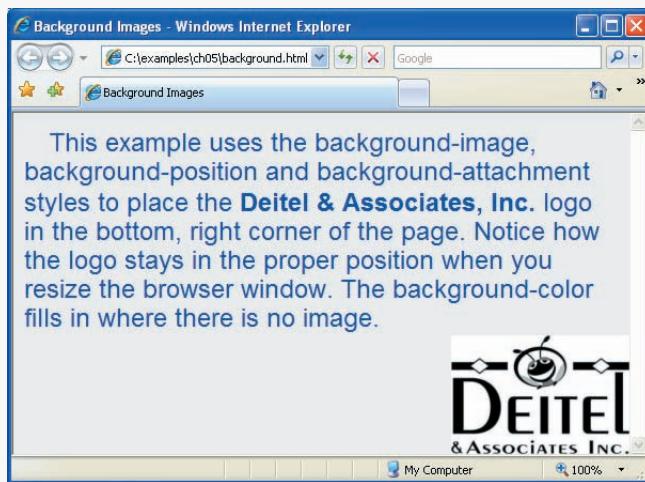


Fig. 5.8 | Adding background images and indentation. (Part 2 of 2.)

The **background-image** property (line 11) specifies the image URL for the image `logo.gif` in the format `url(fileLocation)`. You can also set the **background-color** property in case the image is not found (and to fill in where the image does not cover).

The **background-position** property (line 12) places the image on the page. The keywords `top`, `bottom`, `center`, `left` and `right` are used individually or in combination for vertical and horizontal positioning. An image can be positioned using lengths by specifying the horizontal length followed by the vertical length. For example, to position the image as horizontally centered (positioned at 50 percent of the distance across the screen) and 30 pixels from the top, use

```
background-position: 50% 30px;
```

The **background-repeat** property (line 13) controls background image **tiling**, which places multiple copies of the image next to each other to fill the background. Here, we set the tiling to `no-repeat` to display only one copy of the background image. Other values include `repeat` (the default) to tile the image vertically and horizontally, `repeat-x` to tile the image only horizontally or `repeat-y` to tile the image only vertically.

The final property setting, `background-attachment: fixed` (line 14), fixes the image in the position specified by `background-position`. Scrolling the browser window will not move the image from its position. The default value, `scroll`, moves the image as the user scrolls through the document.

Line 18 uses the `text-indent` property to indent the first line of text in the element by a specified amount, in this case `1em`. An author might use this property to create a web page that reads more like a novel, in which the first line of every paragraph is indented.

Another CSS property that formats text is the `font-style` property, which allows the developer to set text to `none`, `italic` or `oblique` (`oblique` is simply more slanted than `italic`—the browser will default to `italic` if the system or font does not support `oblique` text).

5.8 Element Dimensions

In addition to positioning elements, CSS rules can specify the actual dimensions of each page element. Figure 5.9 demonstrates how to set the dimensions of elements.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 5.9: width.html -->
6  <!-- Element dimensions and text alignment. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8    <head>
9      <title>Box Dimensions</title>
10     <style type = "text/css">
11       div { background-color: #aaccff;
12             margin-bottom: .5em;
13             font-family: arial, helvetica, sans-serif }
14     </style>
15   </head>
16   <body>
17     <div style = "width: 20%">Here is some
18       text that goes in a box which is
19       set to stretch across twenty percent
20       of the width of the screen.</div>
21
22     <div style = "width: 80%; text-align: center">
23       Here is some CENTERED text that goes in a box
24       which is set to stretch across eighty percent of
25       the width of the screen.</div>
26
27     <div style = "width: 20%; height: 150px; overflow: scroll">
28       This box is only twenty percent of
29       the width and has a fixed height.
30       What do we do if it overflows? Set the
31       overflow property to scroll!</div>
32   </body>
33 </html>
```

Fig. 5.9 | Element dimensions and text alignment. (Part I of 2.)

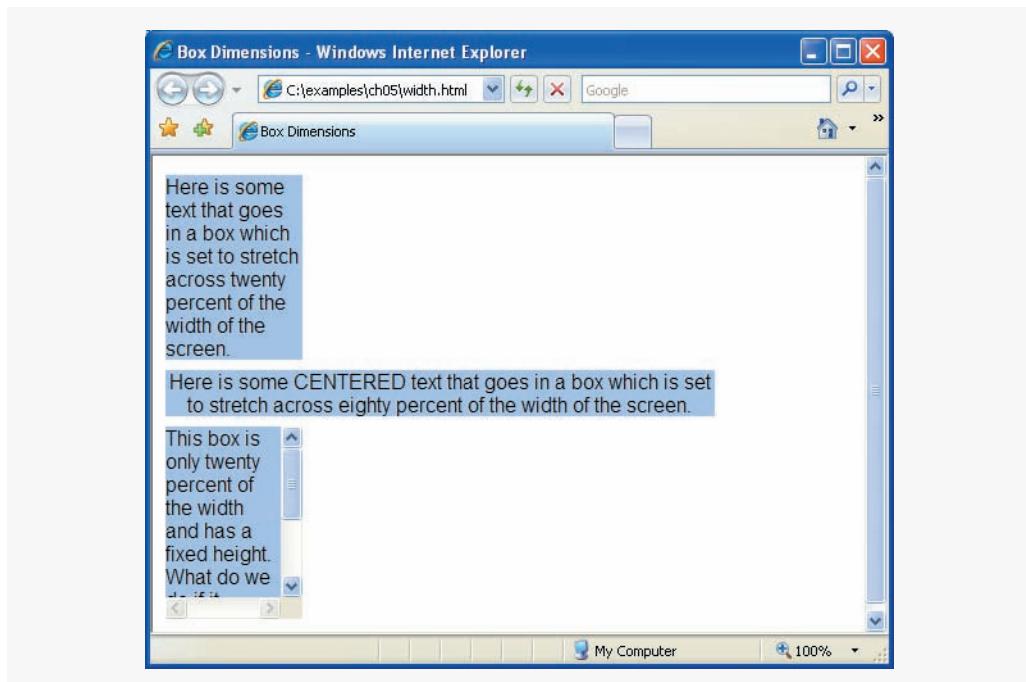


Fig. 5.9 | Element dimensions and text alignment. (Part 2 of 2.)

The inline style in line 17 illustrates how to set the `width` of an element on screen; here, we indicate that the `div` element should occupy 20 percent of the screen width. The height of an element can be set similarly, using the `height` property. The `height` and `width` values also can be specified as relative or absolute lengths. For example,

```
width: 10em
```

sets the element's width to 10 times the font size. Most elements are left aligned by default; however, this alignment can be altered to position the element elsewhere. Line 22 sets text in the element to be center aligned; other values for the `text-align` property include `left` and `right`.

In the third `div`, we specify a percentage height and a pixel width. One problem with setting both dimensions of an element is that the content inside the element can exceed the set boundaries, in which case the element is simply made large enough for all the content to fit. However, in line 27, we set the `overflow` property to `scroll`, a setting that adds scroll bars if the text overflows the boundaries.

5.9 Box Model and Text Flow

All block-level XHTML elements have a virtual box drawn around them based on what is known as the **box model**. When the browser renders elements using the box model, the content of each element is surrounded by **padding**, a **border** and a **margin** (Fig. 5.10).

CSS controls the border using three properties: `border-width`, `border-color` and `border-style`. We illustrate these three properties in Fig. 5.11.

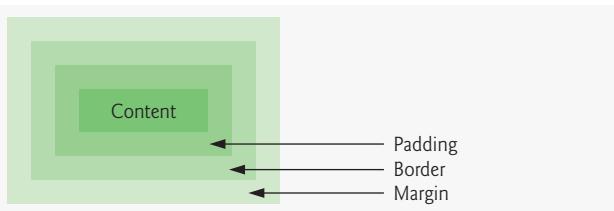


Fig. 5.10 | Box model for block-level elements.

Property `border-width` may be set to any valid CSS length (e.g., `em`, `ex`, `px`, etc.) or to the predefined value of `thin`, `medium` or `thick`. The `border-color` property sets the color. [Note: This property has different meanings for different style borders.] The `border-style` options are `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset` and `outset`. Borders `groove` and `ridge` have opposite effects, as do `inset` and `outset`. When `border-style` is set to `none`, no border is rendered.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 5.11: borders.html -->
6  <!-- Borders of block-level elements. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Borders</title>
10         <style type = "text/css">
11             div      { text-align: center;
12                         width: 50%;
13                         position: relative;
14                         left: 25%;
15                         border-width: 4px }
16             .medium { border-width: medium }
17             .thin   { border-width: thin }
18             .solid  { border-style: solid }
19             .double { border-style: double }
20             .groove { border-style: groove }
21             .inset  { border-style: inset }
22             .outset { border-style: outset }
23             .dashed { border-style: dashed }
24             .red    { border-color: red }
25             .blue   { border-color: blue }
26         </style>
27     </head>
28     <body>
29         <div class = "solid">Solid border</div><hr />
30         <div class = "double">Double border</div><hr />
31         <div class = "groove">Groove border</div><hr />
32         <div class = "inset">Inset border</div><hr />
33         <div class = "dashed">Dashed border</div><hr />

```

Fig. 5.11 | Borders of block-level elements. (Part 1 of 2.)

```

34      <div class = "thin red solid">Thin Red Solid border</div><hr />
35      <div class = "medium blue outset">Medium Blue Outset border</div>
36    </body>
37  </html>

```

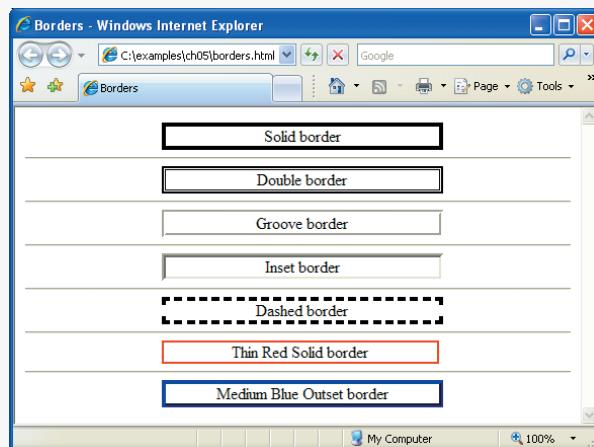


Fig. 5.11 | Borders of block-level elements. (Part 2 of 2.)

Each border property may be set for an individual side of the box (e.g., `border-top-style` or `border-left-color`). Note that we assign more than one class to an XHTML element by separating multiple class names with spaces, as shown in lines 36–37.

As we have seen with absolute positioning, it is possible to remove elements from the normal flow of text. **Floating** allows you to move an element to one side of the screen; other content in the document then flows around the floated element. Figure 5.12 demonstrates how floats and the box model can be used to control the layout of an entire page.

Looking at the XHTML code, we can see that the general structure of this document consists of a header and two main sections. Each section contains a subheading and a paragraph of text.

Block-level elements (such as `divs`) render with a line break before and after their content, so the header and two sections will render vertically one on top of another. In the absence of our styles, the subheading `divs` would also stack vertically on top of the text in the `p` tags. However, in line 24 we set the `float` property to `right` in the class `floated`, which is applied to the subheadings. This causes each subheading `div` to float to the right edge of its containing element, while the paragraph of text will flow around it.

Line 17 assigns a margin of `.5em` to all paragraph tags. The **margin property** sets the space between the outside of the border and all other content on the page. In line 21, we assign `.2em` of padding to the floated `divs`. The **padding property** determines the distance between the content inside an element and the inside of the element's border. Margins for individual sides of an element can be specified (lines 22–23) by using the properties **margin-top**, **margin-right**, **margin-left** and **margin-bottom**. Padding can be specified in the same way, using **padding-top**, **padding-right**, **padding-left** and **padding-bottom**. To see the effects of margins and padding, try putting the margin and padding properties inside comments and observing the difference.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.12: floating.html -->
6 <!-- Floating elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Flowing Text Around Floating Elements</title>
10    <style type = "text/css">
11      div.heading { background-color: #bbddff;
12                     text-align: center;
13                     font-family: arial, helvetica, sans-serif;
14                     padding: .2em }
15      p           { text-align: justify;
16                     font-family: verdana, geneva, sans-serif;
17                     margin: .5em }
18      div.floated { background-color: #eeeeee;
19                     font-size: 1.5em;
20                     font-family: arial, helvetica, sans-serif;
21                     padding: .2em;
22                     margin-left: .5em;
23                     margin-bottom: .5em;
24                     float: right;
25                     text-align: right;
26                     width: 50% }
27      div.section { border: 1px solid #bbddff }
28    </style>
29  </head>
30  <body>
31    <div class = "heading"><img src = "deitel.png" alt = "Deitel" />
32    </div>
33    <div class = "section">
34      <div class = "floated">Corporate Training and Publishing</div>
35      <p>Deitel &amp; Associates, Inc. is an internationally
36      recognized corporate training and publishing organization
37      specializing in programming languages, Internet/World
38      Wide Web technology and object technology education.
39      The company provides courses on Java, C++, Visual Basic, C#,
40      C, Internet and web programming, Object
41      Technology, and more.</p>
42    </div>
43    <div class = "section">
44      <div class = "floated">Leading-Edge Programming Textbooks</div>
45      <p>Through its publishing
46      partnership with Prentice Hall, Deitel &amp; Associates,
47      Inc. publishes leading-edge programming textbooks,
48      professional books, interactive CD-ROM-based multimedia
49      Cyber Classrooms, satellite courses and DVD and web-based
50      video courses.</p>
51    </div>
52  </body>
53 </html>
```

Fig. 5.12 | Floating elements. (Part I of 2.)



Fig. 5.12 | Floating elements. (Part 2 of 2.)

In line 27, we assign a border to the `section` boxes using a shorthand declaration of the border properties. CSS allows shorthand assignments of borders to allow you to define all three border properties in one line. The syntax for this shorthand is `border: <width> <style> <color>`. Our border is one pixel thick, solid, and the same color as the `background-color` property of the heading `div` (line 11). This allows the border to blend with the header and makes the page appear as one box with a line dividing its sections.

5.10 Media Types

CSS `media types` allow a programmer to decide what a page should look like depending on the kind of media being used to display the page. The most common media type for a web page is the `screen media type`, which is a standard computer screen. Other media types in CSS 2 include `handheld`, `braille`, `aural` and `print`. The `handheld` medium is designed for mobile Internet devices, while `braille` is for machines that can read or print web pages in braille. `aural` styles allow the programmer to give a speech-synthesizing web browser more information about the content of the web page. This allows the browser to present a web page in a sensible manner to a visually impaired person. The `print` media type affects a web page's appearance when it is printed. For a complete list of CSS media types, see <http://www.w3.org/TR/REC-CSS2/media.html#media-types>.

Media types allow a programmer to decide how a page should be presented on any one of these media without affecting the others. Figure 5.13 gives a simple example that applies one set of styles when the document is viewed on the screen, and another when the document is printed. To see the difference, look at the screen captures below the paragraph or use the `Print Preview` feature in Internet Explorer or Firefox.

In line 11, we begin a block of styles that applies to all media types, declared by @media all and enclosed in curly braces ({ and }). In lines 13–18, we define some styles for all media types. Lines 20–27 set styles to be applied only when the page is printed, beginning with the declaration @media print and enclosed in curly braces.

The styles we applied for all media types look nice on a screen but would not look good on a printed page. A colored background would use a lot of ink, and a black-and-white printer may print a page that's hard to read because there isn't enough contrast

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.13: mediatypes.html -->
6 <!-- CSS media types. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Media Types</title>
10    <style type = "text/css">
11      @media all
12      {
13        body { background-color: #4488aa }
14        h1 { font-family: verdana, helvetica, sans-serif;
15              color: #aaffcc }
16        p { font-size: 12pt;
17            color: white;
18            font-family: arial, sans-serif }
19      } /* end @media all declaration. */
20      @media print
21      {
22        body { background-color: white }
23        h1 { color: #008844}
24        p { font-size: 14pt;
25            color: #4488aa;
26            font-family: "times new roman", times, serif }
27      } /* end @media print declaration. */
28    </style>
29  </head>
30  <body>
31    <h1>CSS Media Types Example</h1>
32
33    <p>
34    This example uses CSS media types to vary how the page
35    appears in print and how it appears on any other media.
36    This text will appear one font on the screen and a
37    different font on paper or in a print preview. To see
38    the difference in Internet Explorer, go to the Print
39    menu and select Print Preview. In Firefox, select Print
40    Preview from the File menu.
41    </p>
42  </body>
43 </html>
```

Fig. 5.13 | CSS media types. (Part I of 2.)

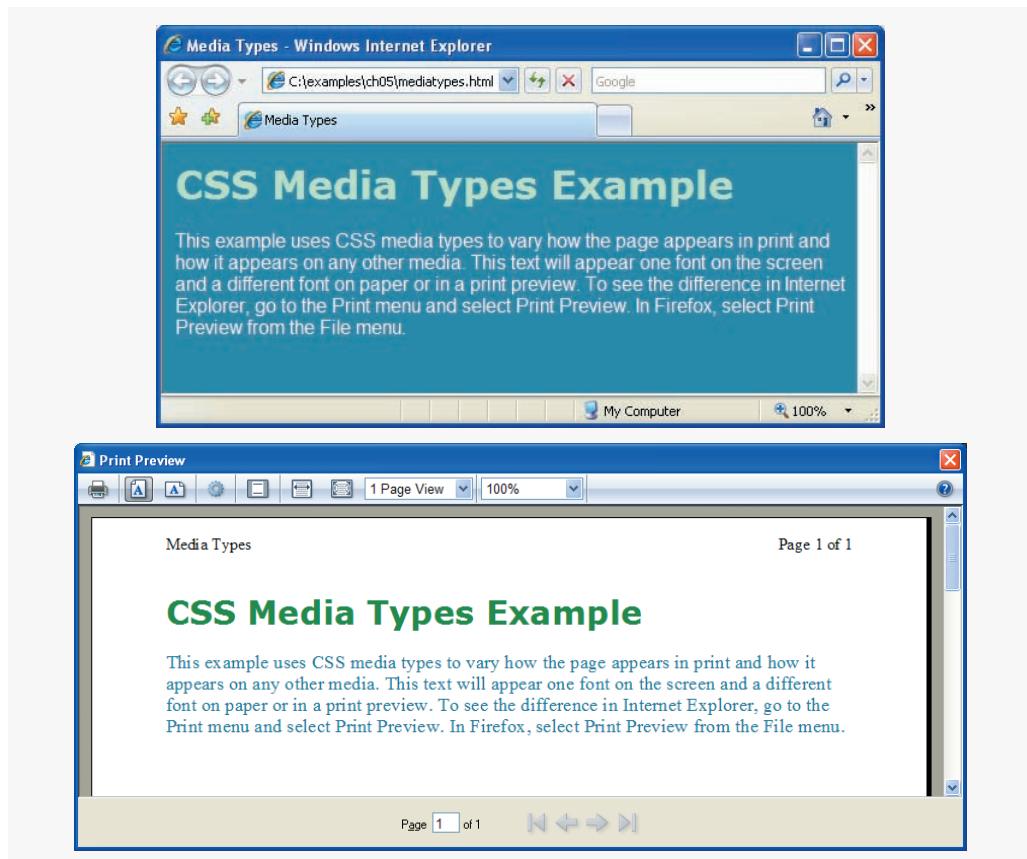


Fig. 5.13 | CSS media types. (Part 2 of 2.)

between the colors. Also, sans-serif fonts like `arial`, `helvetica`, and `geneva` are easier to read on a screen, while serif fonts like `times new roman` are easier to read on paper.



Look-and-Feel Observation 5.1

Pages with dark background colors and light text use a lot of ink and may be difficult to read when printed, especially on a black-and white-printer. Use the print media type to avoid this.



Look-and-Feel Observation 5.2

In general, sans-serif fonts look better on a screen, while serif fonts look better on paper. The print media type allows your web page to display sans-serif font on a screen and change to a serif font when it is printed.

To solve these problems, we apply specific styles for the `print` media type. We change the `background-color` of the `body`, the `color` of the `h1` tag, and the `font-size`, `color`, and `font-family` of the `p` tag to be more suited for printing and viewing on paper. Notice that most of these styles conflict with the declarations in the section for all media types. Since the `print` media type has higher specificity than `all` media types, the `print` styles override the styles for all media types when the page is printed. Since the `font-family`

property of the `h1` tag is not overridden in the `print` section, it retains its old value even when the page is printed.

5.11 Building a CSS Drop-Down Menu

Drop-down menus are a good way to provide navigation links on a website without using a lot of screen space. In this section, we take a second look at the `:hover` pseudoclass and introduce the `display` property to create a drop-down menu using CSS and XHTML.

We've already seen the `:hover` pseudoclass used to change a link's style when the mouse hovers over it. We will use this feature in a more advanced way to cause a menu to appear when the mouse hovers over a menu button. The other important property we need is the `display` property. This allows a programmer to decide whether an element is rendered on the page or not. Possible values include `block`, `inline` and `none`. The `block` and `inline` values display the element as a block element or an inline element, while `none` stops the element from being rendered. The code for the drop-down menu is shown in Fig. 5.14.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 5.14: dropdown.html -->
6  <!-- CSS drop-down menu. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>
10             Drop-Down Menu
11         </title>
12         <style type = "text/css">
13             body                 { font-family: arial, sans-serif }
14             div.menu            { font-weight: bold;
15                             color: white;
16                             border: 2px solid #225599;
17                             text-align: center;
18                             width: 10em;
19                             background-color: #225599 }
20             div.menu:hover a   { display: block }
21             div.menu a        { display: none;
22                             border-top: 2px solid #225599;
23                             background-color: white;
24                             width: 10em;
25                             text-decoration: none;
26                             color: black }
27             div.menu a:hover { background-color: #dfeeff }
28         </style>
29     </head>
30     <body>
31         <div class = "menu">Menu
32             <a href = "#">Home</a>
33             <a href = "#">News</a>
```

Fig. 5.14 | CSS drop-down menu. (Part I of 2.)

```
34      <a href = "#">Articles</a>
35      <a href = "#">Blog</a>
36      <a href = "#">Contact</a>
37  </div>
38 </body>
39 </html>
```

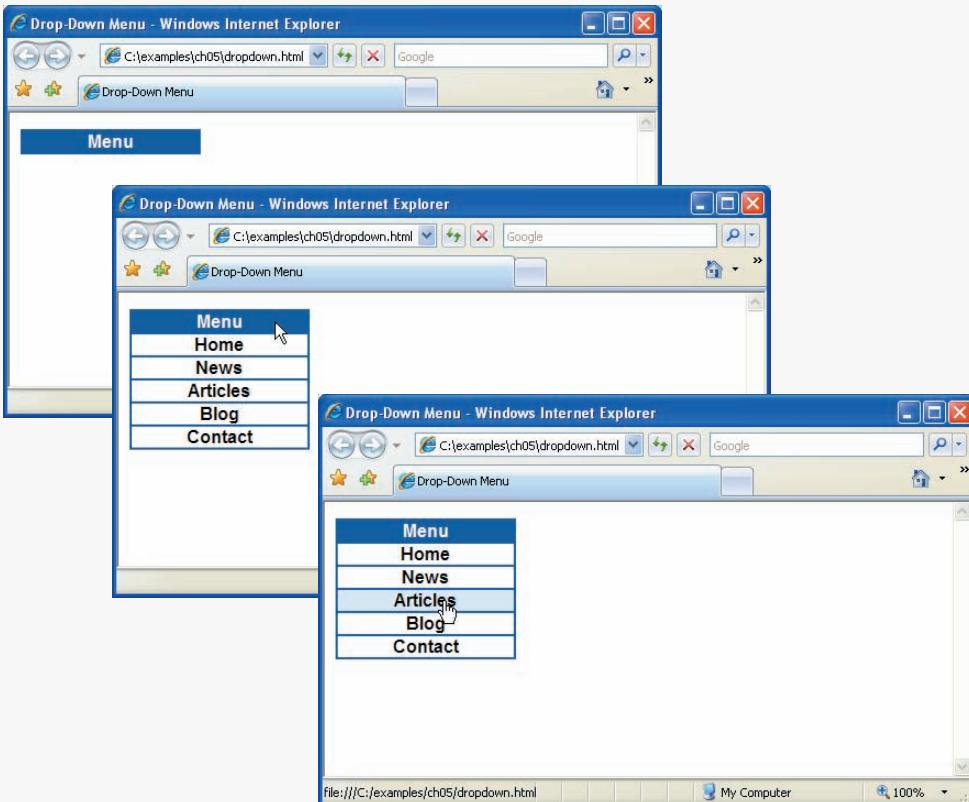


Fig. 5.14 | CSS drop-down menu. (Part 2 of 2.)

First let's look at the XHTML code. In lines 31–37, a `div` of class `menu` has the text "Menu" and five links inside it. This is our drop-down menu. The behavior we want is as follows: the text that says "Menu" should be the only thing visible on the page, unless the mouse is over the menu `div`. When the mouse cursor hovers over the menu `div`, we want the links to appear below the menu for the user to choose from.

To see how we get this functionality, let's look at the CSS code. There are two lines that give us the drop-down functionality. Line 21 selects all the links inside the menu `div` and sets their `display` value to `none`. This instructs the browser not to render the links. The other important style is in line 20. The selectors in this line are similar to those in line 21, except that this line selects only the `a` (anchor) elements that are children of a menu `div` that has the mouse over it. The `display: block` in this line means that when the mouse is over the menu `div`, the links inside it will be displayed as block-level elements.

The selectors in line 27 are also similar to lines 20 and 21. This time, however, the style is applied only to any `a` element that is a child of the `menu` `div` when that child has the mouse cursor over it. This style changes the `background-color` of the currently highlighted menu option. The rest of the CSS simply adds aesthetic style to the components of our menu. Look at the screen captures or run the code example to see the menu in action.

This drop-down menu is just one example of more advanced CSS formatting. Many additional resources are available online for CSS navigation menus and lists. Specifically, check out List-o-Matic, an automatic CSS list generator located at www.accessify.com/tools-and-wizards/developer-tools/list-o-matic/ and Dynamic Drive's library of vertical and horizontal CSS menus at www.dynamicdrive.com/style/.

5.12 User Style Sheets

Users can define their own `user style sheets` to format pages based on their preferences. For example, people with visual impairments may want to increase the page's text size. Web page authors need to be careful not to inadvertently override user preferences with defined styles. This section discusses possible conflicts between `author styles` and `user styles`.

Figure 5.15 contains an author style. The `font-size` is set to `9pt` for all `<p>` tags that have `class note` applied to them.

User style sheets are external style sheets. Figure 5.16 shows a user style sheet that sets the `body`'s `font-size` to `20pt`, `color` to `yellow` and `background-color` to `#0000080`.

User style sheets are not linked to a document; rather, they are set in the browser's options. To add a user style sheet in IE7, select `Internet Options...`, located in the `Tools` menu. In the `Internet Options` dialog (Fig. 5.17) that appears, click `Accessibility...`, check the `Format documents using my style sheet` checkbox, and type the location of the user style sheet. Internet Explorer 7 applies the user style sheet to any document it loads. To add a user style sheet in Firefox, find your Firefox profile using the instructions at

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 5.15: user_absolute.html -->
6 <!-- pt measurement for text size. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>User Styles</title>
10    <style type = "text/css">
11      .note { font-size: 9pt }
12    </style>
13  </head>
14  <body>
15    <p>Thanks for visiting my website. I hope you enjoy it.
16    </p><p class = "note">Please Note: This site will be
17      moving soon. Please check periodically for updates.</p>
18  </body>
19 </html>

```

Fig. 5.15 | pt measurement for text size. (Part 1 of 2.)

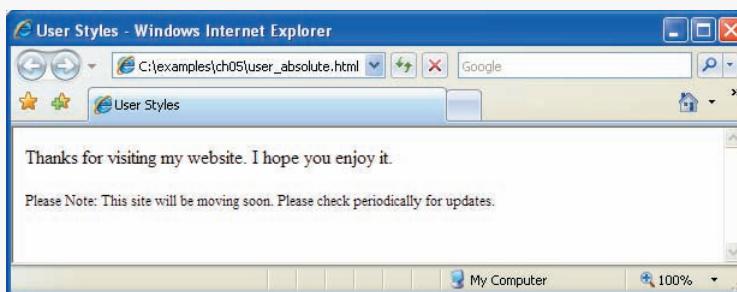


Fig. 5.15 | pt measurement for text size. (Part 2 of 2.)

```

1  /* Fig. 5.16: userstyles.css */
2  /* A user stylesheet */
3  body { font-size: 20pt;
4        color: yellow;
5        background-color: #000080 }
```

Fig. 5.16 | User style sheet.

www.mozilla.org/support/firefox/profile#locate and place a style sheet called `userContent.css` in the `chrome` subdirectory.

The web page from Fig. 5.15 is displayed in Fig. 5.18, with the user style sheet from Fig. 5.16 applied.

In this example, if users define their own `font-size` in a user style sheet, the author style has a higher precedence and overrides the user style. The 9pt font specified in the author style sheet overrides the 20pt font specified in the user style sheet. This small font



Fig. 5.17 | User style sheet in Internet Explorer 7. (Part 1 of 2.)



Fig. 5.17 | User style sheet in Internet Explorer 7. (Part 2 of 2.)

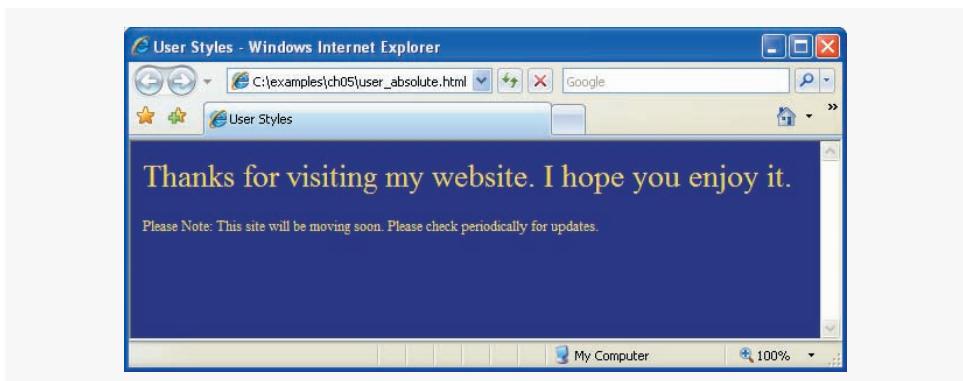


Fig. 5.18 | User style sheet applied with pt measurement.

may make pages difficult to read, especially for individuals with visual impairments. You can avoid this problem by using relative measurements (e.g., em or ex) instead of absolute measurements, such as pt. Figure 5.19 changes the font-size property to use a relative measurement (line 11) that does not override the user style set in Fig. 5.16. Instead, the font size displayed is relative to the one specified in the user style sheet. In this case, text enclosed in the <p> tag displays as 20pt, and <p> tags that have class note applied to them are displayed in 15pt (.75 times 20pt).

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 5.19: user_relative.html -->
6  <!-- em measurement for text size. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>User Styles</title>
```

Fig. 5.19 | em measurement for text size. (Part 1 of 2.)

```

10      <style type = "text/css">
11          .note { font-size: .75em }
12      </style>
13  </head>
14  <body>
15      <p>Thanks for visiting my website. I hope you enjoy it.
16      </p><p class = "note">Please Note: This site will be
17          moving soon. Please check periodically for updates.</p>
18  </body>
19 </html>

```

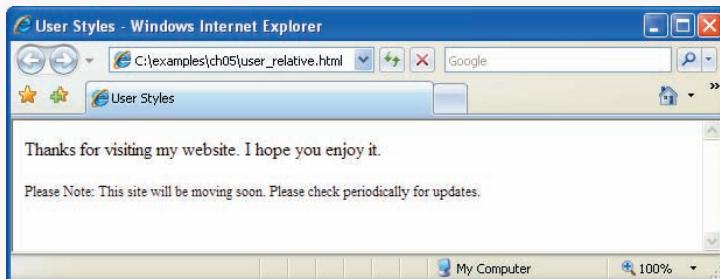


Fig. 5.19 | em measurement for text size. (Part 2 of 2.)

Figure 5.20 displays the web page from Fig. 5.19 with the user style sheet from Fig. 5.16 applied. Note that the second line of text displayed is larger than the same line of text in Fig. 5.18.

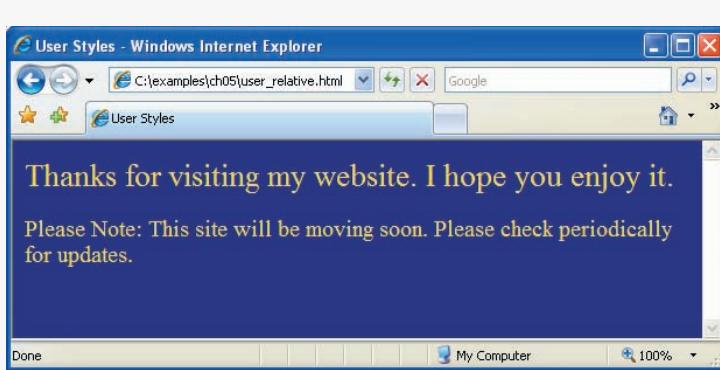


Fig. 5.20 | User style sheet applied with em measurement.

5.13 CSS 3

The W3C is currently developing CSS 3 and some browsers are beginning to implement some of the new features that will be in the [CSS 3](#) specification. We discuss a few of the upcoming features that will most likely be included in CSS 3.

CSS 3 will allow for more advanced control of borders. In addition to the `border-style`, `border-color`, and `border-width` properties, you will be able to set multiple

border colors, use images for borders, add shadows to boxes, and create borders with rounded corners.

Background images will be more versatile in CSS 3, allowing the programmer to set the size of a background image, specify an offset to determine where in the element the image should be positioned, and use multiple background images in one element. There will also be properties to set shadow effects on text and more options for text overflow when the text is too long to fit in its containing element.

Additional features will include resizable boxes, enhanced selectors, multicolumn layouts, and more developed speech (aural) styles. The Web Resources section points you to the Deitel CSS Resource Center, where you can find links to the latest information on the development and features of CSS 3.

5.14 Wrap-Up

In this chapter we introduced Cascading Style Sheets, a technology that allows us to create flexibility in formatting XHTML documents. We learned how to define styles inline, using embedded style sheets, and linking to external style sheets. We learned that external style sheets achieve true separation of content and presentation by putting each in its own file. We discussed how style sheets “cascade” to resolve conflicting style property definitions. We used absolute and relative positioning, background images and colors, and set the dimensions of XHTML elements.

We also learned about the box model and how to use borders, padding, and margins, as well as how to float elements to one side of the page. We provided examples of a web page with separate styles for the print media type, a drop-down menu using the `display` property, and learned more about accessibility and user styles.

XHTML and CSS are the basic technologies behind web pages. In the next chapter, we begin our treatment of JavaScript, which allows us to make more interactive and dynamic pages.

5.15 Web Resources

<http://www.deitel.com/css21>

The Deitel CSS Resource Center contains links to some of the best CSS information on the web. There you'll find categorized links to tutorials, references, code examples, demos, videos, and more. Check out the demos section for more advanced examples of layouts, menus, and other web page components.

Summary

Section 5.2 *Inline Styles*

- The inline style allows you to declare a style for an individual element by using the `style` attribute in the element's start tag.
- Each CSS property is followed by a colon and the value of the attribute. Multiple property declarations are separated by a semicolon.
- The `color` property sets text color. Color names and hexadecimal codes may be used as the value.

Section 5.3 Embedded Style Sheets

- Styles that are placed in a `style` element use selectors to apply style elements throughout the entire document.
- `style` element attribute `type` specifies the MIME type (the specific encoding format) of the style sheet. Style sheets use `text/css`.
- Each rule body in a style sheet begins and ends with a curly brace (`{` and `}`).
- The `font-weight` property specifies the “boldness” of text. Possible values are `bold`, `normal` (the default), `bolder` (bolder than bold text) and `lighter` (lighter than `normal` text).
- Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, ..., 900). Text specified as `normal` is equivalent to 400, and `bold` text is equivalent to 700.
- Style-class declarations are preceded by a period and are applied to elements of the specific class. The `class` attribute applies a style class to an element.
- The CSS rules in a style sheet use the same format as inline styles: The property is followed by a colon (`:`) and the value of that property. Multiple properties are separated by semicolons (`;`).
- The `background-color` attribute specifies the background color of the element.
- The `font-family` attribute names a specific font that should be displayed. Generic font families allow authors to specify a type of font instead of a specific font, in case a browser does not support a specific font. The `font-size` property specifies the size used to render the font.

Section 5.4 Conflicting Styles

- Most styles are inherited from parent elements. Styles defined for children have higher specificity and take precedence over the parent’s styles.
- Pseudoclasses give the author access to content not specifically declared in the document. The `hover` pseudoclass is activated when the user moves the mouse cursor over an element.
- The `text-decoration` property applies decorations to text in an element, such as `underline`, `overline`, `line-through` and `blink`.
- To apply rules to multiple elements, separate the elements with commas in the style sheet.
- To apply rules to only a certain type of element that is a child of another type, separate the element names with spaces.
- A pixel is a relative-length measurement: It varies in size based on screen resolution. Other relative lengths are `em`, `ex` and percentages.
- The other units of measurement available in CSS are absolute-length measurements—that is, units that do not vary in size. These units can be `in` (inches), `cm` (centimeters), `mm` (millimeters), `pt` (points; 1 `pt` = 1/72 `in`) or `pc` (picas; 1 `pc` = 12 `pt`).

Section 5.5 Linking External Style Sheets

- External linking of style sheets can create a uniform look for a website; separate pages can all use the same styles. Modifying a single style-sheet file makes changes to styles across an entire website.
- `link`’s `rel` attribute specifies a relationship between two documents. For style sheets, the `rel` attribute declares the linked document to be a `stylesheet` for the document. The `type` attribute specifies the MIME type of the related document as `text/css`. The `href` attribute provides the URL for the document containing the style sheet.

Section 5.6 Positioning Elements

- The CSS `position` property allows absolute positioning, which provides greater control over where on a page elements reside. Specifying an element’s `position` as `absolute` removes it from

the normal flow of elements on the page and positions it according to distance from the top, left, right or bottom margin of its parent element.

- The `z-index` property allows a developer to layer overlapping elements. Elements that have higher `z-index` values are displayed in front of elements with lower `z-index` values.
- Unlike absolute positioning, relative positioning keeps elements in the general flow on the page and offsets them by the specified top, left, right or bottom value.
- Element `span` is a grouping element—it does not apply any inherent formatting to its contents. Its primary purpose is to apply CSS rules or `id` attributes to a section of text.
- `span` is an inline-level element—it applies formatting to text without changing the flow of the document. Examples of inline elements include `span`, `img`, `a`, `em` and `strong`.
- The `div` element is also a grouping element, but it is a block-level element. This means it is displayed on its own line and has a virtual box around it. Examples of block-level elements include `div`, `p` and heading elements (`h1` through `h6`).

Section 5.7 Backgrounds

- Property `background-image` specifies the URL of the image, in the format `url(fileLocation)`. The property `background-position` places the image on the page using the values top, bottom, center, left and right individually or in combination for vertical and horizontal positioning. You can also position by using lengths.
- The `background-repeat` property controls the tiling of the background image. Setting the tiling to no-repeat displays one copy of the background image on screen. The `background-repeat` property can be set to repeat (the default) to tile the image vertically and horizontally, to `repeat-x` to tile the image only horizontally or to `repeat-y` to tile the image only vertically.
- The property setting `background-attachment: fixed` fixes the image in the position specified by `background-position`. Scrolling the browser window will not move the image from its set position. The default value, `scroll`, moves the image as the user scrolls the window.
- The `text-indent` property indents the first line of text in the element by the specified amount.
- The `font-style` property allows you to set text to none, italic or oblique (oblique will default to italic if the system does not have a separate font file for oblique text, which is normally the case).

Section 5.8 Element Dimensions

- The dimensions of elements on a page can be set with CSS by using properties `height` and `width`.
- Text in an element can be centered using `text-align`; other values for the `text-align` property are `left` and `right`.
- One problem with setting both vertical and horizontal dimensions of an element is that the content inside the element might sometimes exceed the set boundaries, in which case the element must be made large enough for all the content to fit. However, a developer can set the `overflow` property to scroll; this setting adds scroll bars if the text overflows the boundaries set for it.

Section 5.9 Box Model and Text Flow

- The `border-width` property may be set to any of the CSS lengths or to the predefined value of `thin`, `medium` or `thick`.
- The `border-styles` available are `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset` and `outset`.
- The `border-color` property sets the color used for the border.
- The `class` attribute allows more than one class to be assigned to an XHTML element by separating each class name from the next with a space.

- Browsers normally place text and elements on screen in the order in which they appear in the XHTML file. Elements can be removed from the normal flow of text. Floating allows you to move an element to one side of the screen; other content in the document will then flow around the floated element.
- CSS uses a box model to render elements on screen. The content of each element is surrounded by padding, a border and margins. The properties of this box are easily adjusted.
- The `margin` property determines the distance between the element's edge and any outside text.
- Margins for individual sides of an element can be specified by using `margin-top`, `margin-right`, `margin-left` and `margin-bottom`.
- The `padding` property determines the distance between the content inside an element and the edge of the element. Padding also can be set for each side of the box by using `padding-top`, `padding-right`, `padding-left` and `padding-bottom`.

Section 5.10 Media Types

- CSS media types allow a programmer to decide what a page should look like depending on the kind of media being used to display the page. The most common media type for a web page is the `screen` media type, which is a standard computer screen.
- A block of styles that applies to all media types is declared by `@media all` and enclosed in curly braces. To create a block of styles that apply to a single media type such as `print`, use `@media print` and enclose the style rules in curly braces.
- Other media types in CSS 2 include `handheld`, `braille`, `aural` and `print`. The `handheld` medium is designed for mobile Internet devices, while `braille` is for machines that can read or print web pages in braille. `aural` styles allow the programmer to give a speech-synthesizing web browser more information about the content of the web page. The `print` media type affects a web page's appearance when it is printed.

Section 5.11 Building a CSS Drop-Down Menu

- The `:hover` pseudoclass is used to apply styles to an element when the mouse cursor is over it.
- The `display` property allows a programmer to decide if an element is displayed as a `block` element, `inline` element, or is not rendered at all (`none`).

Section 5.12 User Style Sheets

- Users can define their own user style sheets to format pages based on their preferences.
- Absolute font size measurements override user style sheets, while relative font sizes will yield to a user-defined style.

Section 5.13 CSS 3

- While CSS 2 is the current W3C Recommendation, CSS 3 is in development, and some browsers are beginning to implement some of the new features that will be in the CSS 3 specification.
- CSS 3 will introduce new features related to borders, backgrounds, text effects, layout, and more.

Terminology

absolute positioning	<code>background-attachment</code> property
absolute-length measurement	<code>background-color</code> property
ancestor element	<code>background-image</code> property
<code>arial</code> font	<code>background-position</code> property
<code>aural</code> media type	<code>background-repeat</code> property
author style	<code>blink</code> text decoration

block-level element
 border
 border-color property
 border-style property
 border-width property
 box model
 braille media type
 Cascading Style Sheets (CSS)
 class attribute
 cm (centimeter)
 colon (:)
 color property
 CSS 2
 CSS 3
 CSS comment
 CSS property
 CSS rule
 CSS selector
 cursive generic font family
 dashed border style
 descendant element
 display property
 div element
 dotted border style
 double border style
 em (M -height of font)
 embedded style sheet
 ex (x -height of font)
 external style sheets
 float property
 floated element
 font-family property
 font-size property
 font-style property
 font-weight property
 generic font family
 groove border style
 grouping element
 handheld media type
 height property
 hidden border style
 in (inch)
 inheritance
 inline-level element
 inline style
 inset border style
 large relative font size
 larger relative font size
 left property value
 line-through text decoration
 link element
 linking to an external style sheet
 margin property
 margin-bottom property
 margin-left property
 margin-right property
 margin-top property
 media types
 medium relative border width
 medium relative font size
 MIME (Multipurpose Internet Mail
 Extensions) type
 mm (millimeter)
 monospace font
 none border style
 outset border style
 overflow property
 overline text decoration
 padding property
 parent element
 pc (pica)
 position property
 print media type
 pseudoclass
 pt (point)
 rel attribute (link)
 relative positioning
 relative-length measurement
 repeat property value
 ridge border style
 right property value
 sans-serif generic font family
 screen media type
 scroll property value
 selector
 separation of structure from content
 serif generic font family
 small relative font size
 smaller relative font size
 solid border style
 span element
 specificity
 style
 style attribute
 style class
 style in header section of document
 text flow
 text/css MIME type
 text-align property
 text-decoration property

text-indent property	x-large relative font size
thick border width	x-small relative font size
thin border width	xx-large relative font size
user agent	xx-small relative font size
user style sheet	z-index property
width property	

Self-Review Exercises

5.1 Assume that the size of the base font on a system is 12 points.

- a) How big is a 36-point font in ems?
- b) How big is a 9-point font in ems?
- c) How big is a 24-point font in picas?
- d) How big is a 12-point font in inches?
- e) How big is a 1-inch font in picas?

5.2 Fill in the blanks in the following statements:

- a) Using the _____ element allows authors to use external style sheets in their pages.
- b) To apply a CSS rule to more than one element at a time, separate the element names with a(n) _____.
- c) Pixels are a(n) _____-length measurement unit.
- d) The _____ pseudoclass is activated when the user moves the mouse cursor over the specified element.
- e) Setting the overflow property to _____ provides a mechanism for containing inner content without compromising specified box dimensions.
- f) _____ is a generic inline element that applies no inherent formatting and _____ is a generic block-level element that applies no inherent formatting.
- g) Setting property background-repeat to _____ tiles the specified background-image vertically.
- h) To begin a block of styles that applies to only the print media type, you use the declaration _____ print, followed by an opening curly brace {}.
- i) The _____ property allows you to indent the first line of text in an element.
- j) The three components of the box model are the _____, _____ and _____.

Answers to Self-Review Exercises

5.1 a) 3 ems. b) 0.75 ems. c) 2 picas. d) 1/6 inch. e) 6 picas.

5.2 a) link. b) comma. c) relative. d) hover. e) scroll. f) span, div. g) repeat-y. h) @media. i) text-indent. j) padding, border, margin.

Exercises

5.3 Write a CSS rule that makes all text 1.5 times larger than the base font of the system and colors the text red.

5.4 Write a CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.

5.5 Write a CSS rule that gives all h1 and h2 elements a padding of 0.5 ems, a dashed border style and a margin of 0.5 ems.

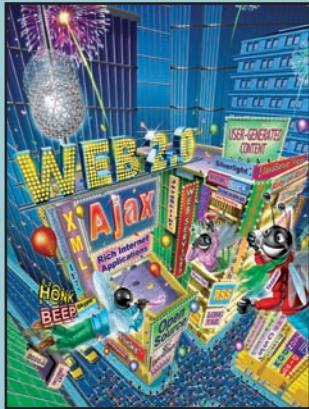
5.6 Write a CSS rule that changes the color of all elements containing attribute class = "greenMove" to green and shifts them down 25 pixels and right 15 pixels.

5.7 Make a layout template that contains a header and two columns. Use `divs` for each layout component, and use `float` to line up the columns side by side. Give each component a border and/or a background color so you can see where your `divs` are.

5.8 Add an embedded style sheet to the XHTML document in Fig. 4.5. The style sheet should contain a rule that displays `h1` elements in blue. In addition, create a rule that displays all links in blue without underlining them. When the mouse hovers over a link, change the link's background color to yellow.

5.9 Make a navigation button using a `div` with a link inside it. Give it a border, background, and text color, and make them change when the user hovers over the button. Use an external style sheet. Make sure your style sheet validates at <http://jigsaw.w3.org/css-validator/>. Note that some warnings may be unavoidable, but your CSS should have no errors.

6



*Comment is free, but facts
are sacred.*

—C. P. Scott

*The creditor hath a better
memory than the debtor.*

—James Howell

*When faced with a decision,
I always ask, "What would
be the most fun?"*

—Peggy Walker

*Equality, in a social sense,
may be divided into that of
condition and that of rights.*

—James Fenimore Cooper

JavaScript: Introduction to Scripting

OBJECTIVES

In this chapter you will learn:

- To write simple JavaScript programs.
- To use input and output statements.
- Basic memory concepts.
- To use arithmetic operators.
- The precedence of arithmetic operators.
- To write decision-making statements.
- To use relational and equality operators.

Outline

- 6.1 Introduction
- 6.2 Simple Program: Displaying a Line of Text in a Web Page
- 6.3 Modifying Our First Program
- 6.4 Obtaining User Input with `prompt` Dialogs
 - 6.4.1 Dynamic Welcome Page
 - 6.4.2 Adding Integers
- 6.5 Memory Concepts
- 6.6 Arithmetic
- 6.7 Decision Making: Equality and Relational Operators
- 6.8 Wrap-Up
- 6.9 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

6.1 Introduction

In the first five chapters, we introduced the Internet and Web, web browsers, Web 2.0, XHTML and Cascading Style Sheets (CSS). In this chapter, we begin our introduction to the [JavaScript¹](#) [scripting language](#), which facilitates a disciplined approach to designing computer programs that enhance the functionality and appearance of web pages.²

In Chapters 6–11, we present a detailed discussion of JavaScript—the *de facto* standard client-side scripting language for web-based applications due to its highly portable nature. Our treatment of JavaScript serves two purposes—it introduces client-side scripting (used in Chapters 6–13), which makes web pages more dynamic and interactive, and it provides the programming foundation for the more complex server-side scripting presented later in the book.

We now introduce JavaScript programming and present examples that illustrate several important features of JavaScript. Each example is carefully analyzed one line at a time. In Chapters 7–8, we present a detailed treatment of program development and program control in JavaScript.

Before you can run code examples with JavaScript on your computer, you may need to change your browser’s security settings. By default, Internet Explorer 7 prevents scripts on your local computer from running, displaying a yellow warning bar at the top of the window instead. To allow scripts to run in files on your computer, select **Internet Options** from the **Tools** menu. Click the **Advanced** tab and scroll down to the **Security** section of

-
1. Many people confuse the scripting language JavaScript with the programming language Java (from Sun Microsystems, Inc.). Java is a full-fledged object-oriented programming language. It can be used to develop applications that execute on a range of devices—from the smallest devices (such as cell phones and PDAs) to supercomputers. Java is popular for developing large-scale distributed enterprise applications and web applications. JavaScript is a browser-based scripting language developed by Netscape and implemented in all major browsers.
 2. JavaScript was originally created by Netscape. Both Netscape and Microsoft have been instrumental in the standardization of JavaScript by ECMA International as ECMAScript. Detailed information about the current ECMAScript standard can be found at www.ecma-international.org/publications/standards/ECMA-262.htm.

the **Settings** list. Check the box labeled **Allow active content to run in files on My Computer** (Fig. 6.1). Click **OK** and restart Internet Explorer. XHTML documents on your own computer that contain JavaScript code will now run properly. Firefox has JavaScript enabled by default.

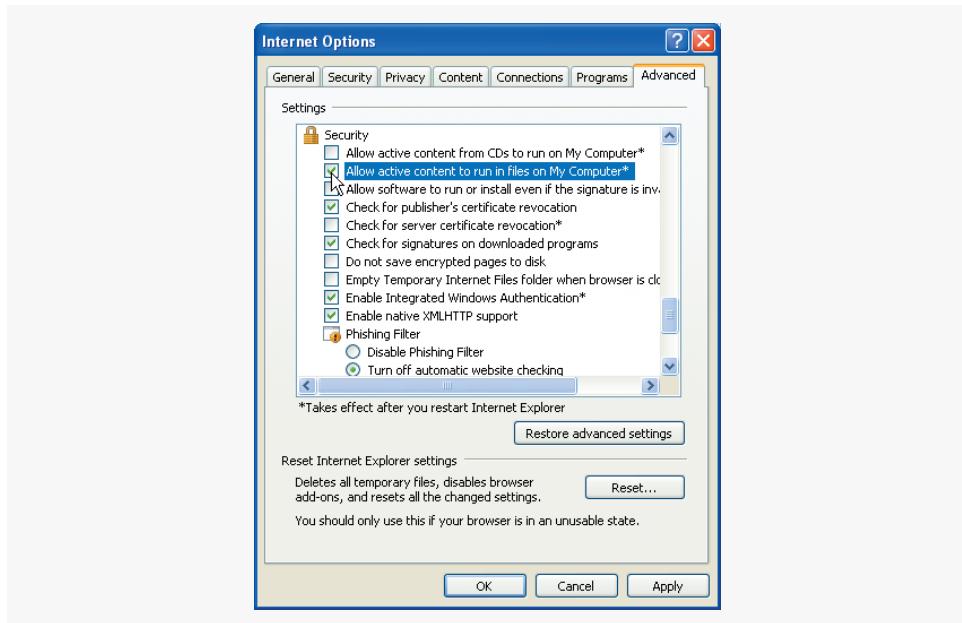


Fig. 6.1 | Enabling JavaScript in Internet Explorer 7

6.2 Simple Program: Displaying a Line of Text in a Web Page

JavaScript uses notations that may appear strange to nonprogrammers. We begin by considering a simple **script** (or **program**) that displays the text "Welcome to JavaScript Programming!" in the body of an XHTML document. All major web browsers contain **JavaScript interpreters**, which process the commands written in JavaScript. The JavaScript code and its output in Internet Explorer are shown in Fig. 6.2.

This program illustrates several important JavaScript features. We consider each line of the XHTML document and script in detail. As in the preceding chapters, we have given each XHTML document line numbers for the reader's convenience; the line numbers are not part of the XHTML document or of the JavaScript programs. Lines 12–13 do the “real work” of the script, namely, displaying the phrase Welcome to JavaScript Programming! in the web page.

Line 8 indicates the beginning of the `<head>` section of the XHTML document. For the moment, the JavaScript code we write will appear in the `<head>` section. The browser interprets the contents of the `<head>` section first, so the JavaScript programs we write there execute before the `<body>` of the XHTML document displays. In later chapters on JavaScript and in the chapters on dynamic HTML, we illustrate **inline scripting**, in which JavaScript code is written in the `<body>` of an XHTML document.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6.2: welcome.html -->
6 <!-- Displaying a line of text. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>A First Program in JavaScript</title>
10    <script type = "text/javascript">
11      <!--
12        document.writeln(
13          "<h1>Welcome to JavaScript Programming!</h1>" );
14      // --
15    </script>
16  </head><body></body>
17 </html>

```



Fig. 6.2 | Displaying a line of text.

Line 10 uses the `<script>` tag to indicate to the browser that the text which follows is part of a script. The `type` attribute specifies the type of file as well as the **scripting language** used in the script—in this case, a `text` file written in `javascript`. Both Internet Explorer and Firefox use JavaScript as the default scripting language.

Line 11 contains the XHTML opening comment tag `<!--`. Some older web browsers do not support scripting. In such browsers, the actual text of a script often will display in the web page. To prevent this from happening, many script programmers enclose the script code in an XHTML comment, so that browsers that do not support scripts will simply ignore the script. The syntax used is as follows:

```

<script type = "text/javascript">
  <!--
    script code here
  // --
</script>

```

When a browser that does not support scripts encounters the preceding code, it ignores the `<script>` and `</script>` tags and the script code in the XHTML comment. Browsers

that do support scripting will interpret the JavaScript code as expected. [Note: Some browsers require the **JavaScript single-line comment //** (see Section 6.4 for an explanation) before the ending XHTML comment delimiter (--) to interpret the script properly. The opening HTML comment tag (<!--) also serves as a single line comment delimiter in JavaScript, therefore it does not need to be commented.]



Portability Tip 6.1

Some browsers do not support the <script>...</script> tags. If your document is to be rendered with such browsers, enclose the script code between these tags in an XHTML comment, so that the script text does not get displayed as part of the web page. The closing comment tag of the XHTML comment (--) is preceded by a JavaScript comment (//) to prevent the browser from trying to interpret the XHTML comment as a JavaScript statement.

Lines 12–13 instruct the browser’s JavaScript interpreter to perform an **action**, namely, to display in the web page the **string** of characters contained between the **double quotation ("") marks**. A string is sometimes called a **character string**, a **message** or a **string literal**. We refer to characters between double quotation marks as strings. Individual white-space characters between words in a string are not ignored by the browser. However, if consecutive spaces appear in a string, browsers condense them to a single space. Also, in most cases, browsers ignore leading white-space characters (i.e., white space at the beginning of a string).



Software Engineering Observation 6.1

Strings in JavaScript can be enclosed in either double quotation marks ("") or single quotation marks ('').

Lines 12–13 use the browser’s **document object**, which represents the XHTML document the browser is currently displaying. The document object allows you to specify text to display in the XHTML document. The browser contains a complete set of objects that allow script programmers to access and manipulate every element of an XHTML document. In the next several chapters, we overview some of these objects as we discuss the Document Object Model (DOM).

An object resides in the computer’s memory and contains information used by the script. The term **object** normally implies that **attributes (data)** and **behaviors (methods)** are associated with the object. The object’s methods use the attributes to perform useful actions for the **client of the object** (i.e., the script that calls the methods). A method may require additional information (**arguments**) to perform its action; this information is enclosed in parentheses after the name of the method in the script. In lines 12–13, we call the **document object’s writeln method** to write a line of XHTML markup in the XHTML document. The parentheses following the method name `writeln` contain the one argument that method `writeln` requires (in this case, the string of XHTML that the browser is to display). Method `writeln` instructs the browser to display the argument string. If the string contains XHTML elements, the browser interprets these elements and renders them on the screen. In this example, the browser displays the phrase `Welcome to JavaScript Programming!` as an `h1`-level XHTML heading, because the phrase is enclosed in an `h1` element.

The code elements in lines 12–13, including `document.writeln`, its argument in the parentheses (the string) and the **semicolon** (;), together are called a **statement**. Every state-

ment ends with a semicolon (also known as the **statement terminator**), although this practice is not required by JavaScript. Line 15 indicates the end of the script.



Good Programming Practice 6.1

Always include a semicolon at the end of a statement to terminate the statement. This notation clarifies where one statement ends and the next statement begins.



Common Programming Error 6.1

Forgetting the ending </script> tag for a script may prevent the browser from interpreting the script properly and may prevent the XHTML document from loading properly.

The `</head>` tag in line 16 indicates the end of the `<head>` section. Also in line 16, the tags `<body>` and `</body>` specify that this XHTML document has an empty body. Line 17 indicates the end of this XHTML document.

We are now ready to view our XHTML document in a web browser—open it in Internet Explorer or Firefox. If the script contains no syntax errors, it should produce the output shown in Fig. 6.2.



Common Programming Error 6.2

*JavaScript is case sensitive. Not using the proper uppercase and lowercase letters is a syntax error. A syntax error occurs when the script interpreter cannot recognize a statement. The interpreter normally issues an error message to help you locate and fix the incorrect statement. Syntax errors are violations of the rules of the programming language. The interpreter notifies you of a syntax error when it attempts to execute the statement containing the error. The JavaScript interpreter in Internet Explorer reports all syntax errors by indicating in a separate popup window that a “runtime error” has occurred (i.e., a problem occurred while the interpreter was running the script). [Note: To enable this feature in IE7, select **Internet Options...** from the **Tools** menu. In the **Internet Options** dialog that appears, select the **Advanced** tab and click the checkbox labelled **Display a notification about every script error** under the **Browsing** category. Firefox has an error console that reports JavaScript errors and warnings. It is accessible by choosing **Error Console** from the **Tools** menu.]*



Error-Prevention Tip 6.1

When the interpreter reports a syntax error, sometimes the error is not on the line number indicated by the error message. First, check the line for which the error was reported. If that line does not contain errors, check the preceding several lines in the script.

6.3 Modifying Our First Program

This section continues our introduction to JavaScript programming with two examples that modify the example in Fig. 6.2.

Displaying a Line of Colored Text

A script can display `Welcome to JavaScript Programming!` several ways. Figure 6.3 uses two JavaScript statements to produce one line of text in the XHTML document. This example also displays the text in a different color, using the CSS `color` property.

Most of this XHTML document is identical to Fig. 6.2, so we concentrate only on lines 12–14 of Fig. 6.3, which display one line of text in the XHTML document. The first statement uses `document` method `write` to display a string. Unlike `writeln`, `write` does

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6.3: welcome2.html -->
6 <!-- Printing one line with multiple statements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Printing a Line with Multiple Statements</title>
10    <script type = "text/javascript">
11      <!--
12        document.write( "<h1 style = \"color: magenta\">" );
13        document.write( "Welcome to JavaScript " +
14          "Programming!</h1>" );
15      // --
16    </script>
17  </head><body></body>
18 </html>
```

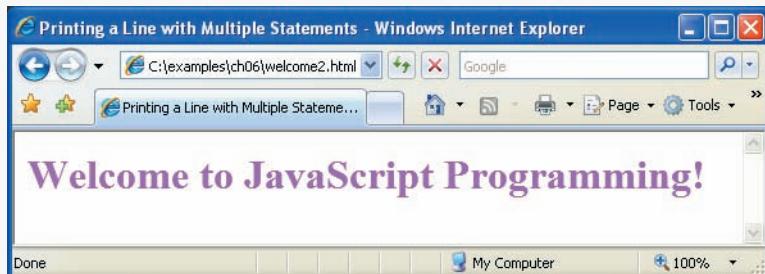


Fig. 6.3 | Printing one line with separate statements.

not position the output cursor in the XHTML document at the beginning of the next line after writing its argument. [Note: The output cursor keeps track of where the next character appears in the XHTML document, not where the next character appears in the web page as rendered by the browser.] The next character written in the XHTML document appears immediately after the last character written with `write`. Thus, when lines 13–14 execute, the first character written, “`W`,” appears immediately after the last character displayed with `write` (the `>` character inside the right double quote in line 12). Each `write` or `writeln` statement resumes writing characters where the last `write` or `writeln` statement stopped writing characters. So, after a `writeln` statement, the next output appears on the beginning of the next line. In effect, the two statements in lines 12–14 result in one line of XHTML text. Remember that statements in JavaScript are separated by semicolons (`;`). Therefore, lines 13–14 represent only one complete statement. JavaScript allows large statements to be split over many lines. However, you cannot split a statement in the middle of a string. The `+` operator (called the “concatenation operator” when used in this manner) in line 13 joins two strings together and is explained in more detail later in this chapter.



Common Programming Error 6.3

Splitting a statement in the middle of a string is a syntax error.

Note that the characters \" (in line 12) are not displayed in the browser. The **backslash** (\) in a string is an **escape character**. It indicates that a “special” character is to be used in the string. When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an **escape sequence**. The escape sequence \" is the **double-quote character**, which causes a double-quote character to be inserted into the string. We use this escape sequence to insert double quotes around the attribute value for style without terminating the string. Note that we could also have used single quotes for the attribute value, as in document.write("<h1 style = 'color: magenta'>");, because the single quotes do not terminate a double-quoted string. We discuss escape sequences in greater detail momentarily.

It is important to note that the preceding discussion has nothing to do with the actual rendering of the XHTML text. Remember that the browser does not create a new line of text unless the browser window is too narrow for the text being rendered or the browser encounters an XHTML element that explicitly starts a new line—for example,
 to start a new line or <p> to start a new paragraph.



Common Programming Error 6.4

Many people confuse the writing of XHTML text with the rendering of XHTML text. Writing XHTML text creates the XHTML that will be rendered by the browser for presentation to the user.

Displaying Multiple Lines of Text

In the next example, we demonstrate that a single statement can cause the browser to display multiple lines by using line-break XHTML tags (
) throughout the string of XHTML text in a write or writeln method call. Figure 6.4 demonstrates the use of line-break XHTML tags. Lines 12–13 produce three separate lines of text when the browser renders the XHTML document.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 6.4: welcome3.html -->
6  <!-- Printing on multiple lines with a single statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Printing Multiple Lines</title>
10         <script type = "text/javascript">
11             <!--
12                 document.writeln( "<h1>Welcome to<br />JavaScript" +
13                     "<br />Programming!</h1>" );
14             // --
15         </script>
16     </head><body></body>
17 </html>
```

Fig. 6.4 | Printing on multiple lines with a single statement. (Part 1 of 2.)



Fig. 6.4 | Printing on multiple lines with a single statement. (Part 2 of 2.)

Displaying Text in an Alert Dialog

The first several programs in this chapter display text in the XHTML document. Sometimes it is useful to display information in windows called **dialogs** (or **dialog boxes**) that “pop up” on the screen to grab the user’s attention. Dialogs typically display important messages to users browsing the web page. JavaScript allows you easily to display a dialog box containing a message. The program in Fig. 6.5 displays Welcome to JavaScript Programming! as three lines in a predefined dialog called an **alert dialog**.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6.5: welcome4.html -->
6 <!-- Alert dialog displaying multiple lines. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Printing Multiple Lines in a Dialog Box</title>
10    <script type = "text/javascript">
11      <!--
12        window.alert( "Welcome to\nJavaScript\nProgramming!" );
13      // --
14    </script>
15  </head>
16  <body>
17    <p>Click Refresh (or Reload) to run this script again.</p>
18  </body>
19 </html>
```



Fig. 6.5 | Alert dialog displaying multiple lines. (Part 1 of 2.)

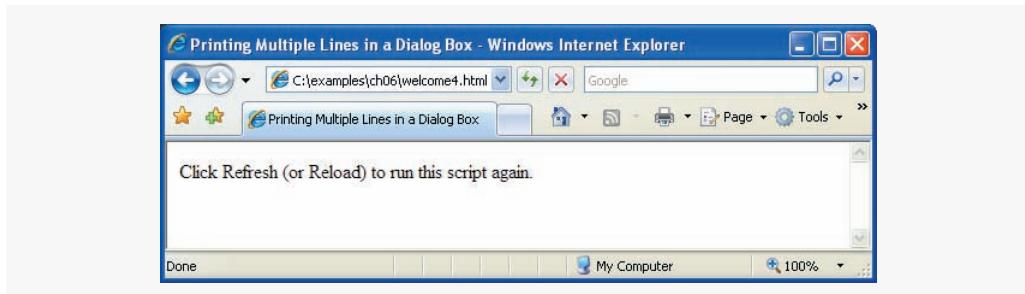


Fig. 6.5 | Alert dialog displaying multiple lines. (Part 2 of 2.)

Line 12 in the script uses the browser's `window` object to display an alert dialog. The argument to the `window` object's `alert` method is the string to display. Executing the preceding statement displays the dialog shown in the first window of Fig. 6.5. The **title bar** of the dialog contains the string **Windows Internet Explorer** to indicate that the browser is presenting a message to the user. The dialog provides an **OK** button that allows the user to **dismiss** (i.e., **close**) the dialog by clicking the button. To dismiss the dialog, position the **mouse cursor** (also called the **mouse pointer**) over the **OK** button and click the mouse. Firefox's alert dialog looks similar, but the title bar contains the text [**JavaScript Application**].



Common Programming Error 6.5

Dialogs display plain text; they do not render XHTML. Therefore, specifying XHTML elements as part of a string to be displayed in a dialog results in the actual characters of the tags being displayed.

Note that the `alert` dialog contains three lines of plain text. Normally, a dialog displays the characters in a string exactly as they appear between the double quotes. Note, however, that the dialog does not display the characters `\n`. The escape sequence `\n` is the **newline character**. In a dialog, the newline character causes the **cursor** (i.e., the current screen position indicator) to move to the beginning of the next line in the dialog. Some other common escape sequences are listed in Fig. 6.6. The `\n`, `\t` and `\r` escape sequences in the table do not affect XHTML rendering unless they are in a **pre element** (this element displays the text between its tags in a fixed-width font exactly as it is formatted between the tags, including leading white-space characters and consecutive white-space characters). The other escape sequences result in characters that will be displayed in plain text dialogs and in XHTML.

Escape sequence	Description
<code>\n</code>	New line. Position the screen cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the screen cursor to the next tab stop.

Fig. 6.6 | Some common escape sequences. (Part 1 of 2.)

Escape sequence	Description
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
\\"	Backslash. Used to represent a backslash character in a string.
\\"	Double quote. Used to represent a double-quote character in a string contained in double quotes. For example, <code>window.alert(\"in quotes\\\");</code> displays "in quotes" in an alert dialog.
\'	Single quote. Used to represent a single-quote character in a string. For example, <code>window.alert('\\\'in quotes\\\');</code> displays 'in quotes' in an alert dialog.

Fig. 6.6 | Some common escape sequences. (Part 2 of 2.)



Common Programming Error 6.6

XHTML elements in an alert dialog's message are not interpreted as XHTML. This means that using `
`, for example, to create a line break in an alert box is an error. The string `
` will simply be included in your message.

6.4 Obtaining User Input with prompt Dialogs

Scripting gives you the ability to generate part or all of a web page's content at the time it is shown to the user. A script can adapt the content based on input from the user or other variables, such as the time of day or the type of browser used by the client. Such web pages are said to be **dynamic**, as opposed to static, since their content has the ability to change. The next two subsections use scripts to demonstrate dynamic web pages.

6.4.1 Dynamic Welcome Page

Our next script builds on prior scripts to create a dynamic welcome page that obtains the user's name, then displays it on the page. The script uses another predefined dialog box from the `window` object—a **prompt** dialog—which allows the user to input a value that the script can use. The program asks the user to input a name, then displays the name in the XHTML document. Figure 6.7 presents the script and sample output. [Note: In later JavaScript chapters, we obtain input via GUI components in XHTML forms, as introduced in Chapter 4.]

Line 12 is a **declaration** that contains the JavaScript **keyword** `var`. Keywords are words that have special meaning in JavaScript. The keyword `var` at the beginning of the statement indicates that the word `name` is a **variable**. A variable is a location in the computer's memory where a value can be stored for use by a program. All variables have a name, type and value, and should be declared with a `var` statement before they are used in

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6.7: welcome5.html -->
6 <!-- Prompt box used on a welcome screen. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Using Prompt and Alert Boxes</title>
10    <script type = "text/javascript">
11      <!--
12        var name; // string entered by the user
13
14        // read the name from the prompt box as a string
15        name = window.prompt( "Please enter your name" );
16
17        document.writeln( "<h1>Hello, " + name +
18          ", welcome to JavaScript programming!</h1>" );
19        // -->
20      </script>
21    </head>
22    <body>
23      <p>Click Refresh (or Reload) to run this script again.</p>
24    </body>
25  </html>

```

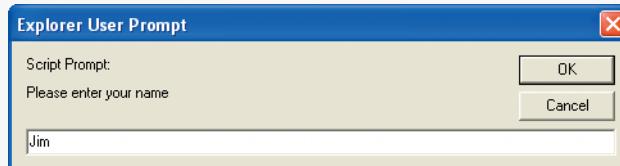


Fig. 6.7 | Prompt box used on a welcome screen.

a program. Although using `var` to declare variables is not required, we will see in Chapter 9, *JavaScript: Functions*, that `var` sometimes ensures proper behavior of a script.

The name of a variable can be any valid **identifier**. An identifier is a series of characters consisting of letters, digits, underscores (`_`) and dollar signs (`$`) that does not begin with a digit and is not a reserved JavaScript keyword. [Note: A complete list of keywords can be

found in Fig. 7.2.] Identifiers may not contain spaces. Some valid identifiers are `Welcome`, `$value`, `_value`, `m_inputField1` and `button7`. The name `7button` is not a valid identifier, because it begins with a digit, and the name `input field` is not valid, because it contains a space. Remember that JavaScript is **case sensitive**—uppercase and lowercase letters are considered to be different characters, so `name`, `Name` and `NAME` are different identifiers.



Good Programming Practice 6.2

Choosing meaningful variable names helps a script to be “self-documenting” (i.e., easy to understand by simply reading the script, rather than having to read manuals or extended comments).



Good Programming Practice 6.3

By convention, variable-name identifiers begin with a lowercase first letter. Each subsequent word should begin with a capital first letter. For example, identifier `itemPrice` has a capital P in its second word, Price.



Common Programming Error 6.7

Splitting a statement in the middle of an identifier is a syntax error.

Declarations end with a semicolon (`;`) and can be split over several lines with each variable in the declaration separated by a comma—known as a **comma-separated list** of variable names. Several variables may be declared either in one declaration or in multiple declarations.

Programmers often indicate the purpose of each variable in the program by placing a JavaScript comment at the end of each line in the declaration. In line 12, a **single-line comment** that begins with the characters `//` states the purpose of the variable in the script. This form of comment is called a single-line comment because it terminates at the end of the line in which it appears. A `//` comment can begin at any position in a line of JavaScript code and continues until the end of the line. Comments do not cause the browser to perform any action when the script is interpreted; rather, comments are ignored by the JavaScript interpreter.



Good Programming Practice 6.4

Some programmers prefer to declare each variable on a separate line. This format allows for easy insertion of a descriptive comment next to each declaration. This is a widely followed professional coding standard.

Another comment notation facilitates the writing of **multiline comments**. For example,

```
/* This is a multiline
comment. It can be
split over many lines. */
```

is a multiline comment spread over several lines. Such comments begin with the delimiter `/*` and end with the delimiter `*/`. All text between the delimiters of the comment is ignored by the interpreter.



Common Programming Error 6.8

Forgetting one of the delimiters of a multiline comment is a syntax error.



Common Programming Error 6.9

Nesting multiline comments (*i.e.*, placing a multiline comment between the delimiters of another multiline comment) is a syntax error.

JavaScript adopted comments delimited with /* and */ from the C programming language and single-line comments delimited with // from the C++ programming language. JavaScript programmers generally prefer C++-style single-line comments over C-style comments. Throughout this book, we use C++-style single-line comments.

Line 14 is a comment indicating the purpose of the statement in the next line. Line 15 calls the window object's prompt method, which displays the dialog in Fig. 6.8. The dialog allows the user to enter a string representing the user's name.

The argument to prompt specifies a message telling the user what to type in the text field. This message is called a **prompt** because it directs the user to take a specific action. An optional second argument, separated from the first by a comma, may specify the default string displayed in the text field; our code does not supply a second argument. In this case, Internet Explorer displays the default value undefined, while Firefox and most other browsers leave the text field empty. The user types characters in the text field, then clicks the **OK** button to submit the string to the program. We normally receive input from a user through a GUI component such as the prompt dialog, as in this program, or through an XHTML form GUI component, as we will see in later chapters.

The user can type anything in the text field of the prompt dialog. For this program, whatever the user enters is considered the name. If the user clicks the **Cancel** button, no string value is sent to the program. Instead, the prompt dialog submits the value null, a JavaScript keyword signifying that a variable has no value. Note that null is not a string literal, but rather a predefined term indicating the absence of value. Writing a null value to the document, however, displays the word null in the web page.

The statement in line 15 **assigns** the value returned by the window object's prompt method (a string containing the characters typed by the user—or the default value or null if the **Cancel** button is clicked) to variable name by using the **assignment operator**, =. The statement is read as, “name gets the value returned by window.prompt(“Please enter your name”).” The = operator is called a **binary operator** because it has two **operands**—name and the result of the expression window.prompt(“Please enter your name”). This entire statement is called an **assignment statement** because it assigns a value to a variable. The expression to the right of the assignment operator is always evaluated first.

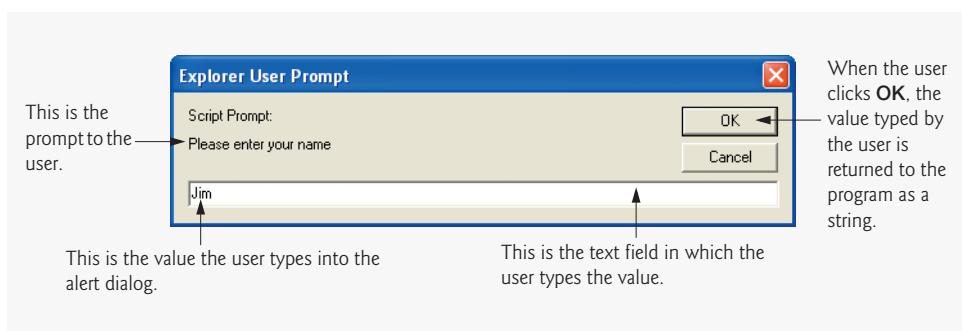


Fig. 6.8 | Prompt dialog displayed by the window object's prompt method.



Good Programming Practice 6.5

Place spaces on either side of a binary operator. This format makes the operator stand out and makes the program more readable.

Lines 17–18 use `document.write` to display the new welcome message. The expression inside the parentheses uses the operator `+` to “add” a string (the literal "`<h1>Hello,` "), the variable name (the string that the user entered in line 15) and another string (the literal "`,` `welcome to JavaScript programming!</h1>`"). JavaScript has a version of the `+` operator for **string concatenation** that enables a string and a value of another data type (including another string) to be combined. The result of this operation is a new (and normally longer) string. If we assume that `name` contains the string literal "Jim", the expression evaluates as follows: JavaScript determines that the two operands of the first `+` operator (the string "`<h1>Hello,` " and the value of variable `name`) are both strings, then concatenates the two into one string. Next, JavaScript determines that the two operands of the second `+` operator (the result of the first concatenation operation, the string "`<h1>Hello, Jim`", and the string "`,` `welcome to JavaScript programming!</h1>`") are both strings and concatenates the two. This results in the string "`<h1>Hello, Jim, welcome to JavaScript programming!</h1>`". The browser renders this string as part of the XHTML document. Note that the space between `Hello`, and `Jim` is part of the string "`<h1>Hello,` ".

As we'll illustrate later, the `+` operator used for string concatenation can convert other variable types to strings if necessary. Because string concatenation occurs between two strings, JavaScript must convert other variable types to strings before it can proceed with the operation. For example, if a variable `age` has an integer value equal to 21, then the expression "`my age is` + `age`" evaluates to the string "`my age is 21`". JavaScript converts the value of `age` to a string and concatenates it with the existing string literal "`my age is` ".

After the browser interprets the `<head>` section of the XHTML document (which contains the JavaScript), it then interprets the `<body>` of the XHTML document (lines 22–24) and renders the XHTML. Notice that the XHTML page is not rendered until the prompt is dismissed because the prompt pauses execution in the head, before the body is processed. If you click your browser's **Refresh** (Internet Explorer) or **Reload** (Firefox) button after entering a name, the browser will reload the XHTML document, so that you can execute the script again and change the name. [Note: In some cases, it may be necessary to hold down the *Shift* key while clicking the **Refresh** or **Reload** button, to ensure that the XHTML document reloads properly. Browsers often save a recent copy of a page in memory, and holding the *Shift* key forces the browser to download the most recent version of a page.]

6.4.2 Adding Integers

Our next script illustrates another use of `prompt` dialogs to obtain input from the user. Figure 6.9 inputs two **integers** (whole numbers, such as 7, -11, 0 and 31914) typed by a user at the keyboard, computes the sum of the values and displays the result.

Lines 12–16 declare the variables `firstNumber`, `secondNumber`, `number1`, `number2` and `sum`. Single-line comments state the purpose of each of these variables. Line 19 employs a `prompt` dialog to allow the user to enter a string representing the first of the two

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6.9: addition.html -->
6 <!-- Addition script. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>An Addition Program</title>
10    <script type = "text/javascript">
11      <!--
12        var firstNumber; // first string entered by user
13        var secondNumber; // second string entered by user
14        var number1; // first number to add
15        var number2; // second number to add
16        var sum; // sum of number1 and number2
17
18        // read in first number from user as a string
19        firstNumber = window.prompt( "Enter first integer" );
20
21        // read in second number from user as a string
22        secondNumber = window.prompt( "Enter second integer" );
23
24        // convert numbers from strings to integers
25        number1 = parseInt( firstNumber );
26        number2 = parseInt( secondNumber );
27
28        sum = number1 + number2; // add the numbers
29
30        // display the results
31        document.writeln( "<h1>The sum is " + sum + "</h1>" );
32        // --
33    </script>
34  </head>
35  <body>
36    <p>Click Refresh (or Reload) to run the script again</p>
37  </body>
38 </html>

```

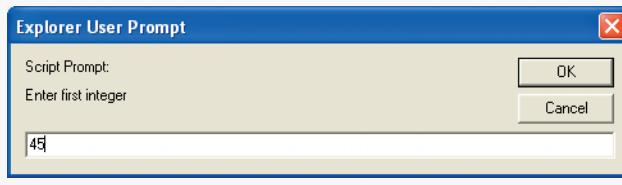


Fig. 6.9 | Addition script. (Part I of 2.)



Fig. 6.9 | Addition script. (Part 2 of 2.)

integers that will be added. The script assigns the first value entered by the user to the variable `firstNumber`. Line 22 displays a `prompt` dialog to obtain the second number to add and assign this value to the variable `secondNumber`.

As in the preceding example, the user can type anything in the `prompt` dialog. For this program, if the user either types a noninteger value or clicks the `Cancel` button, a logic error will occur, and the sum of the two values will appear in the XHTML document as `NaN` (meaning **not a number**). A logic error is caused by syntactically correct code that produces an undesired result. In Chapter 11, *JavaScript: Objects*, we discuss the `Number` object and its methods that can determine whether a value is not a number.

Recall that a `prompt` dialog returns to the program as a string the value typed by the user. Lines 25–26 convert the two strings input by the user to integer values that can be used in a calculation. Function `parseInt` converts its string argument to an integer. Line 25 assigns to the variable `number1` the integer that function `parseInt` returns. Line 26 assigns an integer value to variable `number2` in a similar manner. Any subsequent references to `number1` and `number2` in the program use these integer values. [Note: We refer to `parseInt` as a **function** rather than a method because we do not precede the function call with an object name (such as `document` or `window`) and a dot (.). The term **method** means that the function belongs to a particular object. For example, method `writeln` belongs to the `document` object and method `prompt` belongs to the `window` object.]

Line 28 calculates the sum of the variables `number1` and `number2` using the **addition operator**, `+`, and assigns the result to variable `sum` by using the assignment operator, `=`. Notice that the `+` operator can perform both addition and string concatenation. In this case, the `+` operator performs addition, because both operands contain integers. After line 28 performs this calculation, line 31 uses `document.writeln` to display the result of the addition on the web page. Lines 33 and 34 close the `script` and `head` elements, respectively. Lines 35–37 render the body of XHTML document. Use your browser's **Refresh** or **Reload** button to reload the XHTML document and run the script again.



Common Programming Error 6.10

Confusing the `+` operator used for string concatenation with the `+` operator used for addition often leads to undesired results. For example, if integer variable `y` has the value 5, the expression `"y + 2 = " + y + 2` results in `"y + 2 = 52"`, not `"y + 2 = 7"`, because first the value of `y` (i.e., 5) is concatenated with the string `"y + 2 = "`, then the value 2 is concatenated with the new, larger string `"y + 2 = 5"`. The expression `"y + 2 = " + (y + 2)` produces the string `"y + 2 = 7"` because the parentheses ensure that `y + 2` is executed mathematically before it is converted to a string.

6.5 Memory Concepts

Variable names such as `number1`, `number2` and `sum` actually correspond to **locations** in the computer's memory. Every variable has a **name**, a **type** and a **value**.

In the addition program in Fig. 6.9, when line 25 executes, the string `firstNumber` (previously entered by the user in a `prompt` dialog) is converted to an integer and placed into a memory location to which the name `number1` has been assigned by the interpreter. Suppose the user entered the string `45` as the value for `firstNumber`. The program converts `firstNumber` to an integer, and the computer places the integer value `45` into location `number1`, as shown in Fig. 6.10. Whenever a value is placed in a memory location, the value replaces the previous value in that location. The previous value is lost.

Suppose that the user enters `72` as the second integer. When line 26 executes, the program converts `secondNumber` to an integer and places that integer value, `72`, into location `number2`; then the memory appears as shown in Fig. 6.11.

Once the program has obtained values for `number1` and `number2`, it adds the values and places the sum into variable `sum`. The statement

```
sum = number1 + number2;
```

performs the addition and also replaces `sum`'s previous value. After `sum` is calculated, the memory appears as shown in Fig. 6.12. Note that the values of `number1` and `number2` appear exactly as they did before they were used in the calculation of `sum`. These values were used, but not destroyed, when the computer performed the calculation—when a value is read from a memory location, the process is nondestructive.



Fig. 6.10 | Memory location showing the name and value of variable `number1`.

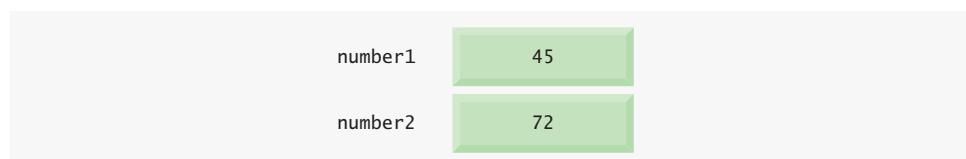


Fig. 6.11 | Memory locations after inputting values for variables `number1` and `number2`.

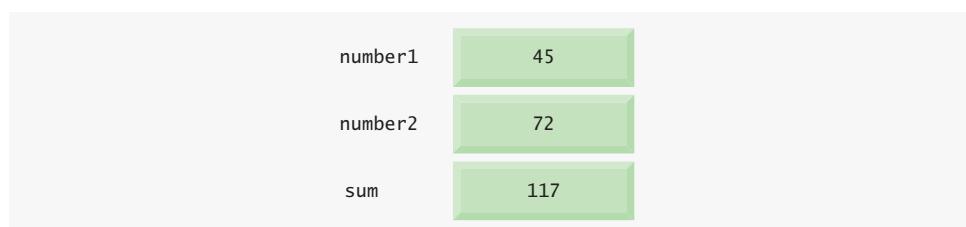


Fig. 6.12 | Memory locations after calculating the `sum` of `number1` and `number2`.

Data Types in JavaScript

Unlike its predecessor languages C, C++ and Java, JavaScript does not require variables to have a declared type before they can be used in a program. A variable in JavaScript can contain a value of any data type, and in many situations JavaScript automatically converts between values of different types for you. For this reason, JavaScript is referred to as a **loosely typed language**. When a variable is declared in JavaScript, but is not given a value, the variable has an **undefined** value. Attempting to use the value of such a variable is normally a logic error.

When variables are declared, they are not assigned values unless specified by the programmer. Assigning the value `null` to a variable indicates that it does not contain a value.

6.6 Arithmetic

Many scripts perform arithmetic calculations. Figure 6.13 summarizes the **arithmetic operators**. Note the use of various special symbols not used in algebra. The **asterisk (*)** indicates multiplication; the **percent sign (%)** is the **remainder operator**, which will be discussed shortly. The arithmetic operators in Fig. 6.13 are binary operators, because each operates on two operands. For example, the expression `sum + value` contains the binary operator `+` and the two operands `sum` and `value`.

JavaScript provides the remainder operator, `%`, which yields the remainder after division. [Note: The `%` operator is known as the modulus operator in some programming languages.] The expression `x % y` yields the remainder after `x` is divided by `y`. Thus, `17 % 5` yields `2` (i.e., `17` divided by `5` is `3`, with a remainder of `2`), and `7.4 % 3.1` yields `1.2`. In later chapters, we consider applications of the remainder operator, such as determining whether one number is a multiple of another. There is no arithmetic operator for exponentiation in JavaScript. (Chapter 8, JavaScript: Control Statements II, shows how to perform exponentiation in JavaScript using the `Math` object's `pow` method.)

Arithmetic expressions in JavaScript must be written in **straight-line form** to facilitate entering programs into the computer. Thus, expressions such as “`a` divided by `b`” must be written as `a / b`, so that all constants, variables and operators appear in a straight line. The following algebraic notation is generally not acceptable to computers:

$$\frac{a}{b}$$

JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	<code>+</code>	$f + 7$	<code>f + 7</code>
Subtraction	<code>-</code>	$p - c$	<code>p - c</code>
Multiplication	<code>*</code>	bm	<code>b * m</code>
Division	<code>/</code>	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Remainder	<code>%</code>	$r \bmod s$	<code>r % s</code>

Fig. 6.13 | Arithmetic operators.

Parentheses are used to group expressions in the same manner as in algebraic expressions. For example, to multiply a times the quantity $b + c$ we write:

```
a * ( b + c )
```

JavaScript applies the operators in arithmetic expressions in a precise sequence determined by the following **rules of operator precedence**, which are generally the same as those followed in algebra:

1. Multiplication, division and remainder operations are applied first. If an expression contains several multiplication, division and remainder operations, operators are applied from left to right. Multiplication, division and remainder operations are said to have the same level of precedence.
2. Addition and subtraction operations are applied next. If an expression contains several addition and subtraction operations, operators are applied from left to right. Addition and subtraction operations have the same level of precedence.

The rules of operator precedence enable JavaScript to apply operators in the correct order. When we say that operators are applied from left to right, we are referring to the **associativity** of the operators—the order in which operators of equal priority are evaluated. We will see that some operators associate from right to left. Figure 6.14 summarizes the rules of operator precedence. The table in Fig. 6.14 will be expanded as additional JavaScript operators are introduced. A complete precedence chart is included in Appendix C.

Now, in light of the rules of operator precedence, let us consider several algebraic expressions. Each example lists an algebraic expression and the equivalent JavaScript expression.

The following is an example of an arithmetic mean (average) of five terms:

$$\text{Algebra: } m = \frac{a + b + c + d + e}{5}$$

JavaScript: `m = (a + b + c + d + e) / 5;`

The parentheses are required to group the addition operators, because division has higher precedence than addition. The entire quantity $(a + b + c + d + e)$ is to be divided by 5. If the parentheses are erroneously omitted, we obtain $a + b + c + d + e / 5$, which evaluates as

$$a + b + c + d + \frac{e}{5}$$

and would not lead to the correct answer.

Operator(s)	Operation(s)	Order of evaluation (precedence)
<code>*</code> , <code>/</code> or <code>%</code>	Multiplication Division Remainder	Evaluated first. If there are several such operations, they are evaluated from left to right.
<code>+</code> or <code>-</code>	Addition Subtraction	Evaluated last. If there are several such operations, they are evaluated from left to right.

Fig. 6.14 | Precedence of arithmetic operators.

The following is an example of the equation of a straight line:

Algebra: $y = mx + b$

JavaScript: `y = m * x + b;`

No parentheses are required. The multiplication operator is applied first, because multiplication has a higher precedence than addition. The assignment occurs last, because it has a lower precedence than multiplication and addition.

The following example contains remainder (%), multiplication, division, addition and subtraction operations:

Algebra: $z = pr \% q + w/x - y$

Java: `z = p * r % q + w / x - y;`



The circled numbers under the statement indicate the order in which JavaScript applies the operators. The multiplication, remainder and division operations are evaluated first in left-to-right order (i.e., they associate from left to right), because they have higher precedence than addition and subtraction. The addition and subtraction operations are evaluated next. These operations are also applied from left to right.

To develop a better understanding of the rules of operator precedence, consider the evaluation of a second-degree polynomial ($y = ax^2 + bx + c$):

`y = a * x * x + b * x + c;`



The circled numbers indicate the order in which JavaScript applies the operators.

Suppose that a , b , c and x are initialized as follows: $a = 2$, $b = 3$, $c = 7$ and $x = 5$. Figure 6.15 illustrates the order in which the operators are applied in the preceding second-degree polynomial.

As in algebra, it is acceptable to use unnecessary parentheses in an expression to make the expression clearer. These are also called **redundant parentheses**. For example, the preceding second-degree polynomial might be parenthesized as follows:

`y = (a * x * x) + (b * x) + c;`



Good Programming Practice 6.6

Using parentheses for complex arithmetic expressions, even when the parentheses are not necessary, can make the arithmetic expressions easier to read.

6.7 Decision Making: Equality and Relational Operators

This section introduces a version of JavaScript's **if** statement that allows a program to make a decision based on the truth or falsity of a **condition**. If the condition is met (i.e., the condition is **true**), the statement in the body of the **if** statement is executed. If the condition is not met (i.e., the condition is **false**), the statement in the body of the **if** state-

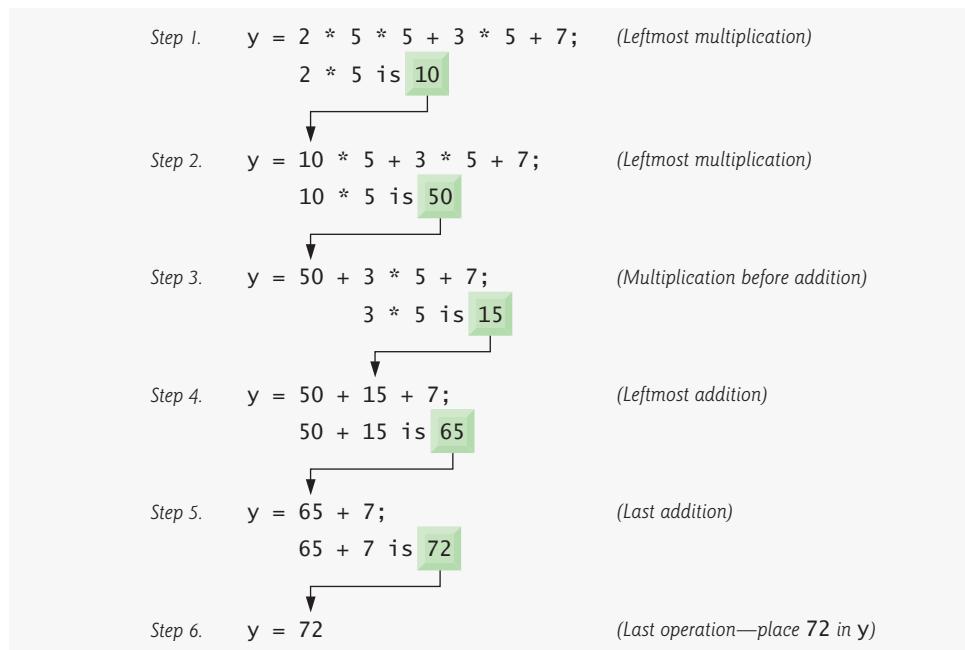


Fig. 6.15 | Order in which a second-degree polynomial is evaluated.

ment is not executed. We will see an example shortly. [Note: Other versions of the `if` statement are introduced in Chapter 7, *JavaScript: Control Statements I*.]

Conditions in `if` statements can be formed by using the **equality operators** and **relational operators** summarized in Fig. 6.16. The relational operators all have the same level

Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	≥	x ≥ y	x is greater than or equal to y
≤	≤	x ≤ y	x is less than or equal to y

Fig. 6.16 | Equality and relational operators.

of precedence and associate from left to right. The equality operators both have the same level of precedence, which is lower than the precedence of the relational operators. The equality operators also associate from left to right.



Common Programming Error 6.11

It is a syntax error if the operators ==, !=, >= and <= contain spaces between their symbols, as in = =, ! =, > = and < =, respectively.



Common Programming Error 6.12

Reversing the operators !=, >= and <=, as in !=, => and =<, respectively, is a syntax error.



Common Programming Error 6.13

Confusing the equality operator, ==, with the assignment operator, =, is a logic error. The equality operator should be read as “is equal to,” and the assignment operator should be read as “gets” or “gets the value of.” Some people prefer to read the equality operator as “double equals” or “equals equals.”

The script in Fig. 6.17 uses four `if` statements to display a time-sensitive greeting on a welcome page. The script obtains the local time from the user’s computer and converts it from 24-hour clock format (0–23) to a 12-hour clock format (0–11). Using this value, the script displays an appropriate greeting for the current time of day. The script and sample output are shown in Fig. 6.17.

Lines 12–14 declare the variables used in the script. Remember that variables may be declared in one declaration or in multiple declarations. If more than one variable is declared in a single declaration (as in this example), the names are separated by commas (,). This list of names is referred to as a comma-separated list. Once again, note the comment at the end of each line, indicating the purpose of each variable in the program. Also note that some of the variables are assigned a value in the declaration—JavaScript allows you to assign a value to a variable when the variable is declared.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 6.17: welcome6.html -->
6  <!-- Using equality and relational operators. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Using Relational Operators</title>
10         <script type = "text/javascript">
11             <!--
12                 var name; // string entered by the user
13                 var now = new Date(); // current date and time
14                 var hour = now.getHours(); // current hour (0-23)
15
16                 // read the name from the prompt box as a string
17                 name = window.prompt( "Please enter your name" );

```

Fig. 6.17 | Using equality and relational operators. (Part I of 2.)

```

18
19      // determine whether it is morning
20      if ( hour < 12 )
21          document.write( "<h1>Good Morning, " );
22
23      // determine whether the time is PM
24      if ( hour >= 12 )
25      {
26          // convert to a 12-hour clock
27          hour = hour - 12;
28
29          // determine whether it is before 6 PM
30          if ( hour < 6 )
31              document.write( "<h1>Good Afternoon, " );
32
33          // determine whether it is after 6 PM
34          if ( hour >= 6 )
35              document.write( "<h1>Good Evening, " );
36      } // end if
37
38      document.writeln( name +
39                      ", welcome to JavaScript programming!</h1>" );
40      // -->
41  </script>
42  </head>
43  <body>
44      <p>Click Refresh (or Reload) to run this script again.</p>
45  </body>
46 </html>

```

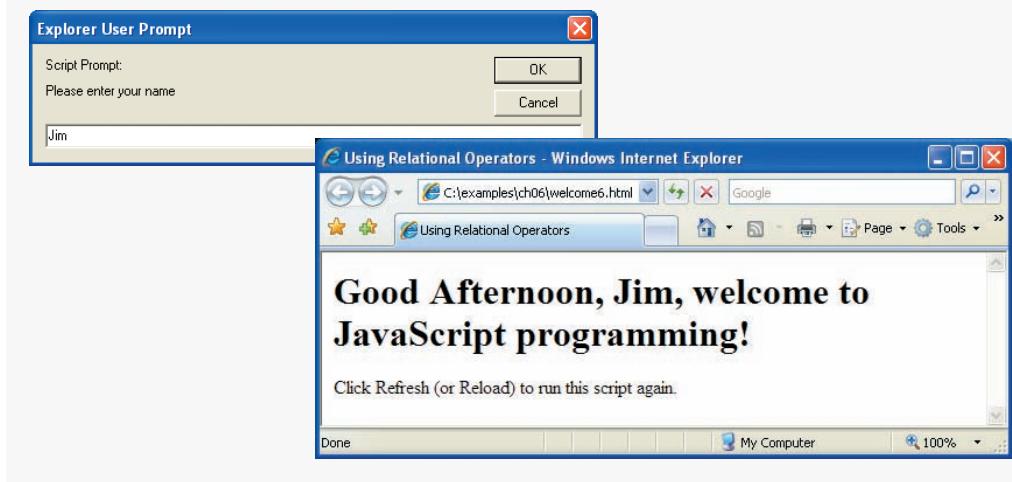


Fig. 6.17 | Using equality and relational operators. (Part 2 of 2.)

Line 13 sets the variable `now` to a new **Date object**, which contains information about the current local time. In Section 6.2, we introduced the `document` object, an object that encapsulates data pertaining to the current web page. Programmers may choose to use

other objects to perform specific tasks or obtain particular pieces of information. Here, we use JavaScript's built-in `Date` object to acquire the current local time. We create a new instance of an object by using the `new` operator followed by the type of the object, `Date`, and a pair of parentheses. Some objects require that arguments be placed in the parentheses to specify details about the object to be created. In this case, we leave the parentheses empty to create a default `Date` object containing information about the current date and time. After line 13 executes, the variable now refers to the new `Date` object. [Note: We did not need to use the `new` operator when we used the `document` and `window` objects because these objects always are created by the browser.] Line 14 sets the variable `hour` to an integer equal to the current hour (in a 24-hour clock format) returned by the `Date` object's `getHours` method. Chapter 11 presents a more detailed discussion of the `Date` object's attributes and methods, and of objects in general. As in the preceding example, the script uses `window.prompt` to allow the user to enter a name to display as part of the greeting (line 17).

To display the correct time-sensitive greeting, the script must determine whether the user is visiting the page during the morning, afternoon or evening. The first `if` statement (lines 20–21) compares the value of variable `hour` with 12. If `hour` is less than 12, then the user is visiting the page during the morning, and the statement at line 21 outputs the string "Good morning". If this condition is not met, line 21 is not executed. Line 24 determines whether `hour` is greater than or equal to 12. If `hour` is greater than or equal to 12, then the user is visiting the page in either the afternoon or the evening. Lines 25–36 execute to determine the appropriate greeting. If `hour` is less than 12, then the JavaScript interpreter does not execute these lines and continues to line 38.

The brace `{` in line 25 begins a block of statements (lines 27–35) that are all executed together if `hour` is greater than or equal to 12—to execute multiple statements inside an `if` construct, enclose them in curly braces. Line 27 subtracts 12 from `hour`, converting the current hour from a 24-hour clock format (0–23) to a 12-hour clock format (0–11). The `if` statement (line 30) determines whether `hour` is now less than 6. If it is, then the time is between noon and 6 PM, and line 31 outputs the beginning of an XHTML `h1` element ("<h1>Good Afternoon, "). If `hour` is greater than or equal to 6, the time is between 6 PM and midnight, and the script outputs the greeting "Good Evening" (lines 34–35). The brace `}` in line 36 ends the block of statements associated with the `if` statement in line 24. Note that `if` statements can be **nested**, i.e., one `if` statement can be placed inside another `if` statement. The `if` statements that determine whether the user is visiting the page in the afternoon or the evening (lines 30–31 and lines 34–35) execute only if the script has already established that `hour` is greater than or equal to 12 (line 24). If the script has already determined the current time of day to be morning, these additional comparisons are not performed. (Chapter 7, JavaScript: Control Statements I, presents a more in-depth discussion of blocks and nested `if` statements.) Finally, lines 38–39 output the rest of the XHTML `h1` element (the remaining part of the greeting), which does not depend on the time of day.



Good Programming Practice 6.7

Include comments after the closing curly brace of control statements (such as `if` statements) to indicate where the statements end, as in line 36 of Fig. 6.17.

Note the indentation of the `if` statements throughout the program. Such indentation enhances program readability.



Good Programming Practice 6.8

Indent the statement in the body of an if statement to make the body of the statement stand out and to enhance program readability.



Good Programming Practice 6.9

Place only one statement per line in a program. This enhances program readability.



Common Programming Error 6.14

Forgetting the left and/or right parentheses for the condition in an if statement is a syntax error. The parentheses are required.

Note that there is no semicolon (;) at the end of the first line of each if statement. Including such a semicolon would result in a logic error at execution time. For example,

```
if ( hour < 12 ) ;  
document.write( "<h1>Good Morning, " );
```

would actually be interpreted by JavaScript erroneously as

```
if ( hour < 12 )  
;  
document.write( "<h1>Good Morning, " );
```

where the semicolon on the line by itself—called the **empty statement**—is the statement to execute if the condition in the if statement is true. When the empty statement executes, no task is performed in the program. The program then continues with the next statement, which executes regardless of whether the condition is true or false. In this example, "<h1>Good Morning, " would be printed regardless of the time of day.



Common Programming Error 6.15

Placing a semicolon immediately after the right parenthesis of the condition in an if statement is normally a logic error. The semicolon would cause the body of the if statement to be empty, so the if statement itself would perform no action, regardless of whether its condition was true. Worse yet, the intended body statement of the if statement would now become a statement in sequence after the if statement and would always be executed.



Common Programming Error 6.16

Leaving out a condition in a series of if statements is normally a logic error. For instance, checking if hour is greater than 12 or less than 12, but not if hour is equal to 12, would mean that the script takes no action when hour is equal to 12. Always be sure to handle every possible condition.

Note the use of spacing in lines 38–39 of Fig. 6.17. Remember that white-space characters, such as tabs, newlines and spaces, are normally ignored by the browser. So, statements may be split over several lines and may be spaced according to the programmer's preferences without affecting the meaning of a program. However, it is incorrect to split identifiers and string literals. Ideally, statements should be kept small, but it is not always possible to do so.



Good Programming Practice 6.10

A lengthy statement may be spread over several lines. If a single statement must be split across lines, choose breaking points that make sense, such as after a comma in a comma-separated list or after an operator in a lengthy expression. If a statement is split across two or more lines, indent all subsequent lines.

The chart in Fig. 6.18 shows the precedence of the operators introduced in this chapter. The operators are shown from top to bottom in decreasing order of precedence. Note that all of these operators, with the exception of the assignment operator, `=`, associate from left to right. Addition is left associative, so an expression like `x + y + z` is evaluated as if it had been written as `(x + y) + z`. The assignment operator, `=`, associates from right to left, so an expression like `x = y = 0` is evaluated as if it had been written as `x = (y = 0)`, which first assigns the value 0 to variable `y`, then assigns the result of that assignment, 0, to `x`.



Good Programming Practice 6.11

Refer to the operator precedence chart when writing expressions containing many operators. Confirm that the operations are performed in the order in which you expect them to be performed. If you are uncertain about the order of evaluation in a complex expression, use parentheses to force the order, exactly as you would do in algebraic expressions. Be sure to observe that some operators, such as assignment (`=`), associate from right to left rather than from left to right.

Operators	Associativity	Type
<code>*</code> <code>/</code> <code>%</code>	left to right	multiplicative
<code>+</code> <code>-</code>	left to right	additive
<code><</code> <code><=</code> <code>></code> <code>>=</code>	left to right	relational
<code>==</code> <code>!=</code>	left to right	equality
<code>=</code>	right to left	assignment

Fig. 6.18 | Precedence and associativity of the operators discussed so far.

6.8 Wrap-Up

We've introduced many important features of JavaScript, including how to display data, how to input data from the keyboard, how to perform calculations and how to make decisions. We also discussed the basics of memory and how variables are stored in a computer. In Chapter 7, we introduce structured programming. You will become more familiar with indentation techniques. We will study how to specify and vary the order in which statements are executed; this order is called the flow of control.

6.9 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 and more. Also check out the tutorials for all skill levels, from introductory to advanced.

Summary

Section 6.1 Introduction

- The JavaScript language facilitates a disciplined approach to the design of computer programs that enhance web pages.

Section 6.2 Simple Program: Displaying a Line of Text in a Web Page

- The spacing displayed by a browser in a web page is determined by the XHTML elements used to format the page.
- Often, JavaScripts appear in the `<head>` section of the XHTML document.
- The browser interprets the contents of the `<head>` section first.
- The `<script>` tag indicates to the browser that the text that follows is part of a script. Attribute `type` specifies the scripting language used in the script—such as `text/javascript`.
- A string of characters can be contained between double ("") or single ('') quotation marks.
- A string is sometimes called a character string, a message or a string literal.
- The browser's `document` object represents the XHTML document currently being displayed in the browser. The `document` object allows a script programmer to specify XHTML text to be displayed in the XHTML document.
- The browser contains a complete set of objects that allow script programmers to access and manipulate every element of an XHTML document.
- An object resides in the computer's memory and contains information used by the script. The term object normally implies that attributes (data) and behaviors (methods) are associated with the object. The object's methods use the attributes' data to perform useful actions for the client of the object—the script that calls the methods.
- The `document` object's `writeln` method writes a line of XHTML text in the XHTML document.
- The parentheses following the name of a method contain the arguments that the method requires to perform its task (or its action).
- Using `writeln` to write a line of XHTML text into a `document` does not guarantee that a corresponding line of text will appear in the XHTML document. The text displayed is dependent on the contents of the string written, which is subsequently rendered by the browser. The browser will interpret the XHTML elements as it normally does to render the final text in the document.
- Every statement should end with a semicolon (also known as the statement terminator), although none is required by JavaScript.
- JavaScript is case sensitive. Not using the proper uppercase and lowercase letters is a syntax error.

Section 6.3 Modifying Our First Program

- Sometimes it is useful to display information in windows called dialogs that “pop up” on the screen to grab the user's attention. Dialogs are typically used to display important messages to the user browsing the web page. The browser's `window` object uses method `alert` to display an alert dialog. Method `alert` requires as its argument the string to be displayed.
- When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence. The escape sequence `\n` is the newline character. It causes the cursor in the XHTML document to move to the beginning of the next line.

Section 6.4 Obtaining User Input with `prompt` Dialogs

- Keywords are words with special meaning in JavaScript.

- The keyword `var` is used to declare the names of variables. A variable is a location in the computer's memory where a value can be stored for use by a program. All variables have a name, type and value, and should be declared with a `var` statement before they are used in a program.
- A variable name can be any valid identifier consisting of letters, digits, underscores (`_`) and dollar signs (`$`) that does not begin with a digit and is not a reserved JavaScript keyword.
- Declarations end with a semicolon (`;`) and can be split over several lines, with each variable in the declaration separated by a comma (forming a comma-separated list of variable names). Several variables may be declared in one declaration or in multiple declarations.
- Programmers often indicate the purpose of a variable in the program by placing a JavaScript comment at the end of the variable's declaration. A single-line comment begins with the characters `/` `/` and terminates at the end of the line. Comments do not cause the browser to perform any action when the script is interpreted; rather, comments are ignored by the JavaScript interpreter.
- Multiline comments begin with delimiter `/*` and end with delimiter `*/`. All text between the delimiters of the comment is ignored by the interpreter.
- The `window` object's `prompt` method displays a dialog into which the user can type a value. The first argument is a message (called a prompt) that directs the user to take a specific action. The optional second argument is the default string to display in the text field.
- A variable is assigned a value with an assignment statement, using the assignment operator, `=`. The `=` operator is called a binary operator, because it has two operands.
- The `null` keyword signifies that a variable has no value. Note that `null` is not a string literal, but rather a predefined term indicating the absence of value. Writing a `null` value to the document, however, displays the word "null".
- Function `parseInt` converts its string argument to an integer.
- JavaScript has a version of the `+` operator for string concatenation that enables a string and a value of another data type (including another string) to be concatenated.

Section 6.5 Memory Concepts

- Variable names correspond to locations in the computer's memory. Every variable has a name, a type and a value.
- When a value is placed in a memory location, the value replaces the previous value in that location. When a value is read out of a memory location, the process is nondestructive.
- JavaScript does not require variables to have a type before they can be used in a program. A variable in JavaScript can contain a value of any data type, and in many situations, JavaScript automatically converts between values of different types for you. For this reason, JavaScript is referred to as a loosely typed language.
- When a variable is declared in JavaScript, but is not given a value, it has an `undefined` value. Attempting to use the value of such a variable is normally a logic error.
- When variables are declared, they are not assigned default values, unless specified otherwise by the programmer. To indicate that a variable does not contain a value, you can assign the value `null` to it.

Section 6.6 Arithmetic

- The basic arithmetic operators (`+`, `-`, `*`, `/`, and `%`) are binary operators, because they each operate on two operands.
- Parentheses can be used to group expressions as in algebra.
- Operators in arithmetic expressions are applied in a precise sequence determined by the rules of operator precedence.

- When we say that operators are applied from left to right, we are referring to the associativity of the operators. Some operators associate from right to left.

Section 6.7 Decision Making: Equality and Relational Operators

- JavaScript's `if` statement allows a program to make a decision based on the truth or falsity of a condition. If the condition is met (i.e., the condition is true), the statement in the body of the `if` statement is executed. If the condition is not met (i.e., the condition is false), the statement in the body of the `if` statement is not executed.
- Conditions in `if` statements can be formed by using the equality operators and relational operators.

Terminology

\\" double-quote escape sequence	empty statement
\\" backslash escape sequence	equality operators
\' single quote escape sequence	error message
\n newline escape sequence	escape sequence
\r carriage return escape sequence	false
\t tab escape sequence	function
action	identifier
addition operator (+)	<code>if</code> statement
<code>alert</code> dialog	inline scripting
<code>alert</code> method of the <code>window</code> object	integer
argument to a method	interpreter
arithmetic expression in straight-line form	JavaScript
arithmetic operator	JavaScript interpreter
assignment	keyword
assignment operator (=)	level of precedence
assignment statement	literal
associativity of operators	location in the computer's memory
attribute	logic error
backslash (\) escape character	loosely typed language
behavior	meaningful variable name
binary operator	method
case sensitive	mouse cursor
character string	multiline comment /* and */
client of an object	multiplication operator (*)
comma-separated list	name of a variable
comment	<code>Nan</code> (not a number)
condition	nested <code>if</code> statements
cursor	<code>new</code> operator
data	newline character (\n)
data type	<code>null</code>
<code>Date</code> object	object
decision making	operand
declaration	operator associativity
dialog	operator precedence
division operator (/)	parentheses
<code>document</code> object	<code>parseInt</code> function
double quotation ("") marks	perform an action
ECMAScript standard	<code>pre</code> element

program	string concatenation operator (+)
prompt	string literal
prompt dialog	string of characters
prompt method of the window object	subtraction operator (-)
redundant parentheses	syntax error
relational operator	text field
remainder after division	title bar of a dialog
remainder operator (%)	true
rules of operator precedence	type attribute of the <script> tag
runtime error	type of a variable
script	undefined
script element	value of a variable
scripting language	var keyword
semicolon (;) statement terminator	variable
single quotation ('') mark	violation of the language rules
single-line comment (//)	white-space character
statement	whole number
straight-line form	window object
string	write method of the document object
string concatenation	writeln method of the document object

Self-Review Exercises

6.1 Fill in the blanks in each of the following statements:

- _____ begins a single-line comment.
- Every statement should end with a(n) _____.
- The _____ statement is used to make decisions.
- _____, ____, ____ and _____ are known as white space.
- The _____ object displays alert dialogs and prompt dialogs.
- _____ are words that are reserved for use by JavaScript.
- Methods _____ and _____ of the _____ object write XHTML text into an XHTML document.

6.2 State whether each of the following is *true* or *false*. If *false*, explain why.

- Comments cause the computer to print the text after the // on the screen when the program is executed.
- JavaScript considers the variables `number` and `NuMbEr` to be identical.
- The remainder operator (%) can be used only with numeric operands.
- The arithmetic operators *, /, %, + and - all have the same level of precedence.
- Method `parseInt` converts an integer to a string.

6.3 Write JavaScript statements to accomplish each of the following tasks:

- Declare variables `c`, `thisIsAVariable`, `q76354` and `number`.
- Display a dialog asking the user to enter an integer. Show a default value of 0 in the text field.
- Convert a string to an integer, and store the converted value in variable `age`. Assume that the string is stored in `stringValue`.
- If the variable `number` is not equal to 7, display "The variable number is not equal to 7" in a message dialog.
- Output a line of XHTML text that will display the message "This is a JavaScript program" on one line in the XHTML document.

- f) Output a line of XHTML text that will display the message "This is a JavaScript program" on two lines in the XHTML document. Use only one statement.

6.4 Identify and correct the errors in each of the following statements:

- a) `if (c < 7);
 window.alert("c is less than 7");`
- b) `if (c => 7)
 window.alert("c is equal to or greater than 7");`

6.5 Write a statement (or comment) to accomplish each of the following tasks:

- a) State that a program will calculate the product of three integers [*Hint:* Use text that helps to document a program.]
- b) Declare the variables `x`, `y` and `result`.
- c) Declare the variables `xVal`, `yVal` and `zVal`.
- d) Prompt the user to enter the first value, read the value from the user and store it in the variable `xVal`.
- e) Prompt the user to enter the second value, read the value from the user and store it in the variable `yVal`.
- f) Prompt the user to enter the third value, read the value from the user and store it in the variable `zVal`.
- g) Convert `xVal` to an integer, and store the result in the variable `x`.
- h) Convert `yVal` to an integer, and store the result in the variable `y`.
- i) Convert `zVal` to an integer, and store the result in the variable `z`.
- j) Compute the product of the three integers contained in variables `x`, `y` and `z`, and assign the result to the variable `result`.
- k) Write a line of XHTML text containing the string "The product is " followed by the value of the variable `result`.

6.6 Using the statements you wrote in Exercise 6.5, write a complete program that calculates and prints the product of three integers.

Answers to Self-Review Exercises

6.1 a) `//`. b) Semicolon (`;`). c) `if`. d) Space characters, newline characters and tab characters. e) `window`. f) Keywords. g) `write`, `writeln`, `document`.

6.2 a) False. Comments do not cause any action to be performed when the program is executed. They are used to document programs and improve their readability. b) False. JavaScript is case sensitive, so these variables are distinct. c) True. d) False. The operators `*`, `/` and `%` are on the same level of precedence, and the operators `+` and `-` are on a lower level of precedence. e) False. Function `parseInt` converts a string to an integer value.

6.3 a) `var c, thisIsAVariable, q76354, number;`
 b) `value = window.prompt("Enter an integer", "0");`
 c) `var age = parseInt(stringValue);`
 d) `if (number != 7)
 window.alert("The variable number is not equal to 7");`
 e) `document.writeln("This is a JavaScript program");`
 f) `document.writeln("This is a
JavaScript program");`

6.4 a) Error: There should not be a semicolon after the right parenthesis of the condition in the `if` statement.

Correction: Remove the semicolon after the right parenthesis. [Note: The result of this error is that the output statement is executed whether or not the condition in the `if`

statement is true. The semicolon after the right parenthesis is considered an empty statement—a statement that does nothing.]

- b) Error: The relational operator `=>` is incorrect.

Correction: Change `=>` to `>=`.

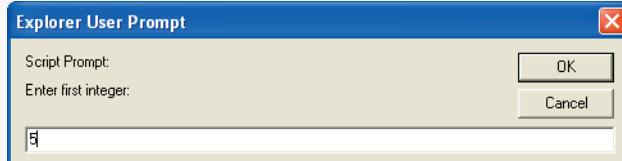
6.5

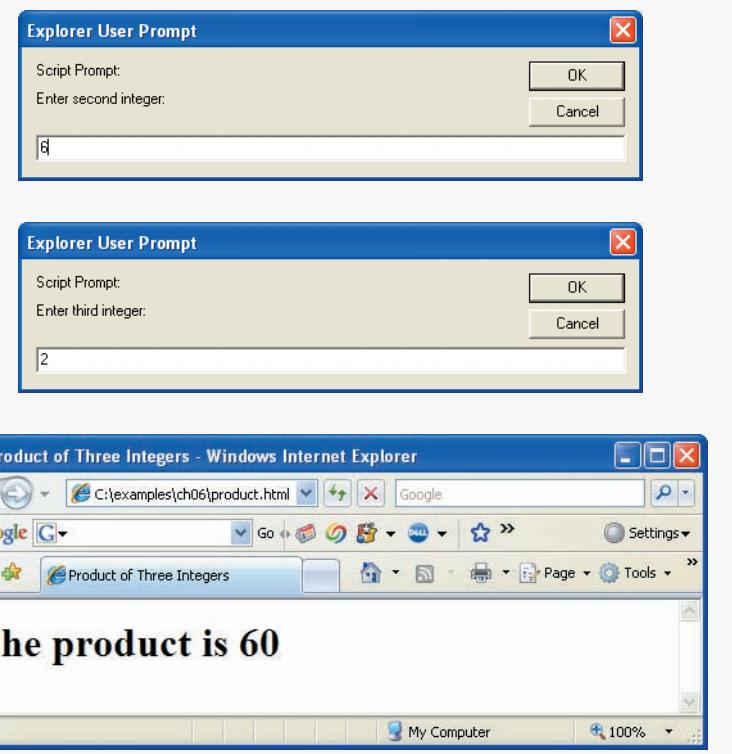
```
a) // Calculate the product of three integers
b) var x, y, z, result;
c) var xVal, yVal, zVal;
d) xVal = window.prompt( "Enter first integer:", "0" );
e) yVal = window.prompt( "Enter second integer:", "0" );
f) zVal = window.prompt( "Enter third integer:", "0" );
g) x = parseInt( xVal );
h) y = parseInt( yVal );
i) z = parseInt( zVal );
j) result = x * y * z;
k) document.writeln( "<h1>The product is " + result + "</h1>" );
```

6.6

The program is as follows:

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Exercise 6.6: product.html -->
6 <html xmlns = "http://www.w3.org/1999/xhtml">
7   <head>
8     <title>Product of Three Integers</title>
9     <script type = "text/javascript">
10       <!--
11         // Calculate the product of three integers
12         var x, y, z, result;
13         var xVal, yVal, zVal;
14
15         xVal = window.prompt( "Enter first integer:", "0" );
16         yVal = window.prompt( "Enter second integer:", "0" );
17         zVal = window.prompt( "Enter third integer:", "0" );
18
19         x = parseInt( xVal );
20         y = parseInt( yVal );
21         z = parseInt( zVal );
22
23         result = x * y * z;
24         document.writeln( "<h1>The product is " +
25           result + "</h1>" );
26       // --
27     </script>
28   </head><body></body>
29 </html>
```





Exercises

- 6.7** Fill in the blanks in each of the following statements:
- _____ are used to document a program and improve its readability.
 - A dialog capable of receiving input from the user is displayed with method _____ of object _____.
 - A JavaScript statement that makes a decision is the _____ statement.
 - Calculations are normally performed by _____ operators.
 - A dialog capable of showing a message to the user is displayed with method _____ of object _____.
- 6.8** Write JavaScript statements that accomplish each of the following tasks:
- Display the message "Enter two numbers" using the window object.
 - Assign the product of variables b and c to variable a.
 - State that a program performs a sample payroll calculation.
- 6.9** State whether each of the following is *true* or *false*. If *false*, explain why.
- JavaScript operators are evaluated from left to right.
 - The following are all valid variable names: `_under_bar_`, `m928134`, `t5`, `j7`, `her_sales$`, `his$_account_total1`, `a`, `b$`, `c`, `z`, `z2`.
 - A valid JavaScript arithmetic expression with no parentheses is evaluated from left to right.
 - The following are all invalid variable names: `3g`, `87`, `67h2`, `h22`, `2h`.

- 6.10** Fill in the blanks in each of the following statements:
- What arithmetic operations have the same precedence as multiplication? _____.
 - When parentheses are nested, which set of parentheses is evaluated first in an arithmetic expression? _____.
 - A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a _____.
- 6.11** What displays in the message dialog when each of the given JavaScript statements is performed? Assume that $x = 2$ and $y = 3$.
- `window.alert("x = " + x);`
 - `window.alert("The value of x + x is " + (x + x));`
 - `window.alert("x = ");`
 - `window.alert((x + y) + " = " + (y + x));`
- 6.12** Which of the following JavaScript statements contain variables whose values are destroyed (i.e., changed or replaced)?
- `p = i + j + k + 7;`
 - `window.alert("variables whose values are destroyed");`
 - `window.alert("a = 5");`
 - `stringVal = window.prompt("Enter string:");`
- 6.13** Given $y = ax^3 + 7$, which of the following are correct JavaScript statements for this equation?
- `y = a * x * x * x + 7;`
 - `y = a * x * x * (x + 7);`
 - `y = (a * x) * x * (x + 7);`
 - `y = (a * x) * x * x + 7;`
 - `y = a * (x * x * x) + 7;`
 - `y = a * x * (x * x + 7);`
- 6.14** State the order of evaluation of the operators in each of the following JavaScript statements, and show the value of x after each statement is performed.
- `x = 7 + 3 * 6 / 2 - 1;`
 - `x = 2 % 2 + 2 * 2 - 2 / 2;`
 - `x = (3 * 9 * (3 + (9 * 3 / (3))));`
- 6.15** Write a script that displays the numbers 1 to 4 on the same line, with each pair of adjacent numbers separated by one space. Write the program using the following methods:
- Using one `document.writeln` statement.
 - Using four `document.write` statements.
- 6.16** Write a script that asks the user to enter two numbers, obtains the two numbers from the user and outputs text that displays the sum, product, difference and quotient of the two numbers. Use the techniques shown in Fig. 6.9.
- 6.17** Write a script that asks the user to enter two integers, obtains the numbers from the user and outputs text that displays the larger number followed by the words "is larger" in an alert dialog. If the numbers are equal, output XHTML text that displays the message "These numbers are equal." Use the techniques shown in Fig. 6.17.
- 6.18** Write a script that takes three integers from the user and displays the sum, average, product, smallest and largest of the numbers in an `alert` dialog.
- 6.19** Write a script that gets from the user the radius of a circle and outputs XHTML text that displays the circle's diameter, circumference and area. Use the constant value 3.14159 for π . Use the GUI techniques shown in Fig. 6.9. [Note: You may also use the predefined constant `Math.PI` for the value of π . This constant is more precise than the value 3.14159. The `Math` object is defined by Java-

Script and provides many common mathematical capabilities.] Use the following formulas (r is the radius): $diameter = 2r$, $circumference = 2\pi r$, $area = \pi r^2$.

- 6.20** Write a script that outputs XHTML text that displays in the XHTML document a rectangle, an oval, an arrow and a diamond using asterisks (*), as follows [Note: Use the `<pre>` and `</pre>` tags to specify that the asterisks should be displayed using a fixed-width font]:

```
*****  
*   *   ***   *   *  
*   *   *   *   ***  
*   *   *   *   *  
*   *   *   *   *  
*   *   *   *   *  
*   *   *   *   *  
*   *   *   *   *  
*   *   *   *   *  
*****  
***   *   *
```

- 6.21** Modify the program you created in Exercise 6.20 by removing the `<pre>` and `</pre>` tags. Does the program display the shapes exactly as in Exercise 6.20?

- 6.22** What does the following code print?

```
document.writeln( "*\\n**\\n***\\n****\\n*****" );
```

- 6.23** What does the following code print?

```
document.writeln( "**" );  
document.writeln( "***" );  
document.writeln( "****" );  
document.writeln( "*****" );  
document.writeln( "***" );
```

- 6.24** What does the following code print?

```
document.write( "*<br />" );  
document.write( "***<br />" );  
document.write( "****<br />" );  
document.write( "*****<br />" );  
document.writeln( "***" );
```

- 6.25** What does the following code print?

```
document.write( "*<br />" );  
document.writeln( "***" );  
document.writeln( "****" );  
document.write( "***<br />" );  
document.writeln( "***" );
```

- 6.26** Write a script that reads five integers and determines and outputs XHTML text that displays the largest and smallest integers in the group. Use only the programming techniques you learned in this chapter.

- 6.27** Write a script that reads an integer and determines and outputs XHTML text that displays whether it is odd or even. [Hint: Use the remainder operator. An even number is a multiple of 2. Any multiple of 2 leaves a remainder of zero when divided by 2.]

- 6.28** Write a script that reads in two integers and determines and outputs XHTML text that displays whether the first is a multiple of the second. [Hint: Use the remainder operator.]

- 6.29** Write a script that outputs XHTML text that displays in the XHTML document a checkboard pattern, as follows:

```
* * * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * *  
* * * * * * * *
```

6.30 Write a script that inputs five numbers and determines and outputs XHTML text that displays the number of negative numbers input, the number of positive numbers input and the number of zeros input.

6.31 Write a script that calculates the squares and cubes of the numbers from 0 to 10 and outputs XHTML text that displays the resulting values in an XHTML table format, as follows:

number	square	cube
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

[Note: This program does not require any input from the user.]

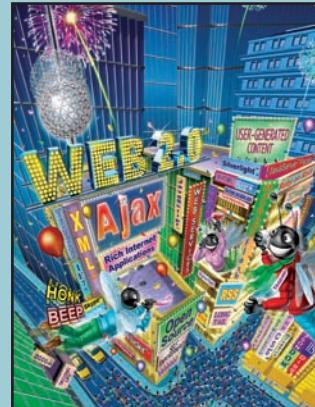
7

JavaScript: Control Statements I

OBJECTIVES

In this chapter you will learn:

- Basic problem-solving techniques.
- To develop algorithms through the process of top-down, stepwise refinement.
- To use the `if` and `if...else` selection statements to choose among alternative actions.
- To use the `while` repetition statement to execute statements in a script repeatedly.
- Counter-controlled repetition and sentinel-controlled repetition.
- To use the increment, decrement and assignment operators.



Let's all move one place on.

—Lewis Carroll

The wheel is come full circle.

—William Shakespeare

*How many apples fell on
Newton's head before he took
the hint!*

—Robert Frost

Outline

- 7.1 Introduction
- 7.2 Algorithms
- 7.3 Pseudocode
- 7.4 Control Structures
- 7.5 `if` Selection Statement
- 7.6 `if...else` Selection Statement
- 7.7 `while` Repetition Statement
- 7.8 Formulating Algorithms: Counter-Controlled Repetition
- 7.9 Formulating Algorithms: Sentinel-Controlled Repetition
- 7.10 Formulating Algorithms: Nested Control Statements
- 7.11 Assignment Operators
- 7.12 Increment and Decrement Operators
- 7.13 Wrap-Up
- 7.14 Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

7.1 Introduction

Before writing a script to solve a problem, it is essential to have a thorough understanding of the problem and a carefully planned approach to solving the problem. When writing a script, it is equally essential to understand the types of building blocks that are available and to employ proven program-construction principles. In this chapter and in Chapter 8, we discuss these issues in our presentation of the theory and principles of structured programming. The techniques you will learn here are applicable to most high-level languages, including JavaScript.

7.2 Algorithms

Any computable problem can be solved by executing a series of actions in a specific order. A **procedure** for solving a problem in terms of

1. the **actions** to be executed, and
2. the **order** in which the actions are to be executed

is called an **algorithm**. The following example demonstrates that correctly specifying the order in which the actions are to execute is important.

Consider the “rise-and-shine algorithm” followed by one junior executive for getting out of bed and going to work: (1) get out of bed, (2) take off pajamas, (3) take a shower, (4) get dressed, (5) eat breakfast, (6) carpool to work. This routine gets the executive to work well prepared to make critical decisions. Suppose, however, that the same steps are performed in a slightly different order: (1) get out of bed, (2) take off pajamas, (3) get dressed, (4) take a shower, (5) eat breakfast, (6) carpool to work. In this case, our junior executive shows up for work soaking wet. Specifying the order in which statements are to be executed in a computer program is called **program control**. In this chapter and Chapter 8, we investigate the program-control capabilities of JavaScript.

7.3 Pseudocode

Pseudocode is an artificial and informal language that helps programmers develop algorithms. The pseudocode we present here is useful for developing algorithms that will be converted to structured portions of JavaScript programs. Pseudocode is similar to everyday English; it is convenient and user friendly, although it is not an actual computer programming language.



Software Engineering Observation 7.1

Pseudocode is often used to “think out” a program during the program-design process. Then the pseudocode program is converted to a programming language such as JavaScript.

The style of pseudocode we present consists purely of characters, so that programmers may conveniently type pseudocode in an editor program. The computer can produce a fresh printed copy of a pseudocode program on demand. Carefully prepared pseudocode may easily be converted to a corresponding JavaScript program. This process is done in many cases simply by replacing pseudocode statements with their JavaScript equivalents. In this chapter, we give several examples of pseudocode.

Pseudocode normally describes only executable statements—the actions that are performed when the program is converted from pseudocode to JavaScript and is run. Declarations are not executable statements. For example, the declaration

```
var value1;
```

instructs the JavaScript interpreter to reserve space in memory for the variable `value1`. This declaration does not cause any action—such as input, output or a calculation—to occur when the script executes. Some programmers choose to list variables and mention the purpose of each variable at the beginning of a pseudocode program.

7.4 Control Structures

Normally, statements in a program execute one after the other in the order in which they are written. This process is called **sequential execution**. Various JavaScript statements we will soon discuss enable the programmer to specify that the next statement to execute may not be the next one in sequence. This is known as **transfer of control**.

During the 1960s, it became clear that the indiscriminate use of transfers of control was the root of much difficulty experienced by software development groups. The finger of blame was pointed at the **goto statement**, which allowed the programmer to specify a transfer of control to one of a wide range of possible destinations in a program. The notion of so-called **structured programming** became almost synonymous with “**goto elimination**.” JavaScript does not have a `goto` statement.

The research of Bohm and Jacopini demonstrated that programs could be written without `goto` statements.¹ The challenge of the era for programmers was to shift their styles to “`goto`-less programming.” It was not until the 1970s that programmers started taking structured programming seriously. The results were impressive, as software development groups reported reduced development times, more frequent on-time delivery of

1. Bohm, C., and G. Jacopini, “Flow Diagrams, Turing Machines, and Languages with Only Two Formation Rules,” *Communications of the ACM*, Vol. 9, No. 5, May 1966, pp. 336–371.

systems and more frequent within-budget completion of software projects. The key to these successes is that structured programs are clearer, easier to debug and modify and more likely to be bug free in the first place.

Bohm and Jacopini's work demonstrated that all programs could be written in terms of only three **control structures**, namely the **sequence structure**, the **selection structure** and the **repetition structure**. The sequence structure is built into JavaScript. Unless directed otherwise, the computer executes JavaScript statements one after the other in the order in which they are written (i.e., in sequence). The **flowchart** segment of Fig. 7.1 illustrates a typical sequence structure in which two calculations are performed in order.

A **flowchart** is a graphical representation of an algorithm or of a portion of an algorithm. Flowcharts are drawn using certain special-purpose symbols, such as rectangles, diamonds, ovals and small circles; these symbols are connected by arrows called **flowlines**, which indicate the order in which the actions of the algorithm execute.

Like pseudocode, flowcharts often are useful for developing and representing algorithms, although pseudocode is strongly preferred by many programmers. Flowcharts show clearly how control structures operate; that is all we use them for in this text. Carefully compare the pseudocode and flowchart representations of each control structure.

Consider the flowchart segment for the sequence structure on the left side of Fig. 7.1. We use the **rectangle symbol** (or **action symbol**) to indicate any type of action, including a calculation or an input/output operation. The flowlines in the figure indicate the order in which the actions are performed—the first action adds grade to total, then the second action adds 1 to counter. JavaScript allows us to have as many actions as we want in a sequence structure. Anywhere a single action may be placed, as we will soon see, we may place several actions in sequence.

In a flowchart that represents a *complete* algorithm, an **oval symbol** containing the word “Begin” is the first symbol used; an oval symbol containing the word “End” indicates where the algorithm ends. In a flowchart that shows only a portion of an algorithm, as in Fig. 7.1, the oval symbols are omitted in favor of using **small circle symbols**, also called **connector symbols**.

Perhaps the most important flowcharting symbol is the **diamond symbol**, also called the **decision symbol**, which indicates that a decision is to be made. We discuss the diamond symbol in the next section.

JavaScript provides three types of selection structures; we discuss each in this chapter and in Chapter 8. The **if** selection statement performs (selects) an action if a condition is true or skips the action if the condition is false. The **if...else** selection statement per-

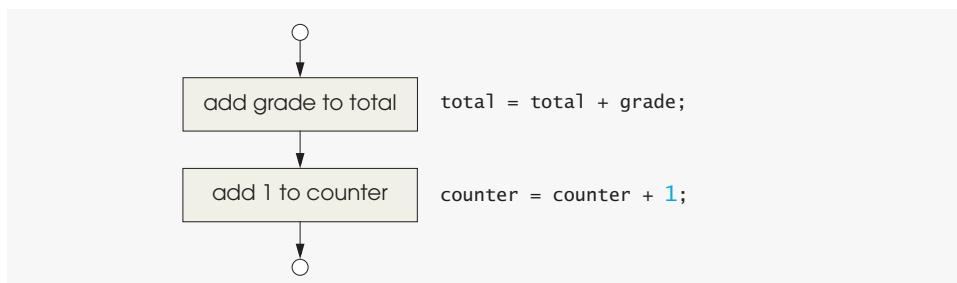


Fig. 7.1 | Flowcharting JavaScript's sequence structure.

forms an action if a condition is true and performs a different action if the condition is false. The `switch` selection statement (Chapter 8) performs one of many different actions, depending on the value of an expression.

The `if` statement is called a **single-selection structure** because it selects or ignores a single action (or, as we will soon see, a single group of actions). The `if...else` statement is a **double-selection structure** because it selects between two different actions (or groups of actions). The `switch` statement is a **multiple-selection structure** because it selects among many different actions (or groups of actions).

JavaScript provides four repetition structure types, namely `while`, `do...while`, `for` and `for...in`. (`do...while` and `for` are covered in Chapter 8; `for...in` is covered in Chapter 10.) Each of the words `if`, `else`, `switch`, `while`, `do`, `for` and `in` is a JavaScript **keyword**. These words are reserved by the language to implement various features, such as JavaScript's control structures. Keywords cannot be used as identifiers (e.g., for variable names). A complete list of JavaScript keywords is shown in Fig. 7.2.



Common Programming Error 7.1

Using a keyword as an identifier is a syntax error.

As we have shown, JavaScript has only eight control structures: sequence, three types of selection and four types of repetition. A program is formed by combining control structures as necessary to implement the program's algorithm. As with the sequence structure in Fig. 7.1, we will see that each control structure is flowcharted with two small circle symbols, one at the entry point to the control structure and one at the exit point.

JavaScript keywords				
<code>break</code>	<code>case</code>	<code>catch</code>	<code>continue</code>	<code>default</code>
<code>delete</code>	<code>do</code>	<code>else</code>	<code>false</code>	<code>finally</code>
<code>for</code>	<code>function</code>	<code>if</code>	<code>in</code>	<code>instanceof</code>
<code>new</code>	<code>null</code>	<code>return</code>	<code>switch</code>	<code>this</code>
<code>throw</code>	<code>true</code>	<code>try</code>	<code>typeof</code>	<code>var</code>
<code>void</code>	<code>while</code>	<code>with</code>		
<i>Keywords that are reserved but not used by JavaScript</i>				
<code>abstract</code>	<code>boolean</code>	<code>byte</code>	<code>char</code>	<code>class</code>
<code>const</code>	<code>debugger</code>	<code>double</code>	<code>enum</code>	<code>export</code>
<code>extends</code>	<code>final</code>	<code>float</code>	<code>goto</code>	<code>implements</code>
<code>import</code>	<code>int</code>	<code>interface</code>	<code>long</code>	<code>native</code>
<code>package</code>	<code>private</code>	<code>protected</code>	<code>public</code>	<code>short</code>
<code>static</code>	<code>super</code>	<code>synchronized</code>	<code>throws</code>	<code>transient</code>
<code>volatile</code>				

Fig. 7.2 | JavaScript keywords.

Single-entry/single-exit control structures make it easy to build programs; the control structures are attached to one another by connecting the exit point of one to the entry point of the next. This process is similar to the way in which a child stacks building blocks, so we call it **control-structure stacking**. We will learn that there is only one other way in which control structures may be connected—**control-structure nesting**. Thus, algorithms in JavaScript programs are constructed from only eight different types of control structures combined in only two ways.

7.5 if Selection Statement

A selection structure is used to choose among alternative courses of action in a program. For example, suppose that the passing grade on an examination is 60 (out of 100). Then the pseudocode statement

```
If student's grade is greater than or equal to 60
    Print "Passed"
```

determines whether the condition “student’s grade is greater than or equal to 60” is true or false. If the condition is true, then “Passed” is printed, and the next pseudocode statement in order is “performed” (remember that pseudocode is not a real programming language). If the condition is false, the print statement is ignored, and the next pseudocode statement in order is performed.

Note that the second line of this selection structure is indented. Such indentation is optional but is highly recommended, because it emphasizes the inherent structure of structured programs. The JavaScript interpreter ignores white-space characters—blanks, tabs and newlines used for indentation and vertical spacing. Programmers insert these white-space characters to enhance program clarity.



Good Programming Practice 7.1

Consistently applying reasonable indentation conventions throughout your programs improves program readability. We suggest a fixed-size tab of about 1/4 inch or three spaces per indent.

The preceding pseudocode *If* statement can be written in JavaScript as

```
if ( studentGrade >= 60 )
    document.writeln( "Passed" );
```

Note that the JavaScript code corresponds closely to the pseudocode. This similarity is the reason that pseudocode is a useful program-development tool. The statement in the body of the *if* statement outputs the character string “Passed” in the XHTML document.

The flowchart in Fig. 7.3 illustrates the single-selection *if* statement. This flowchart contains what is perhaps the most important flowcharting symbol—the diamond symbol (or decision symbol), which indicates that a decision is to be made. The decision symbol contains an expression, such as a condition, that can be either **true** or **false**. The decision symbol has two flowlines emerging from it. One indicates the path to follow in the program when the expression in the symbol is **true**; the other indicates the path to follow in the program when the expression is **false**. A decision can be made on any expression that evaluates to a value of JavaScript’s boolean type (i.e., any expression that evaluates to **true** or **false**—also known as a **boolean expression**).

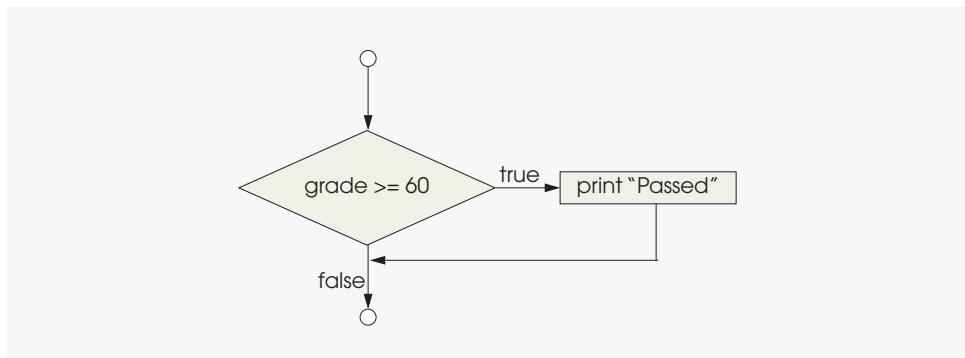


Fig. 7.3 | Flowcharting the single-selection `if` statement.



Software Engineering Observation 7.2

In JavaScript, any nonzero numeric value in a condition evaluates to `true`, and 0 evaluates to `false`. For strings, any string containing one or more characters evaluates to `true`, and the empty string (the string containing no characters, represented as "") evaluates to `false`. Also, a variable that has been declared with `var` but has not been assigned a value evaluates to `false`.

Note that the `if` statement is a single-entry/single-exit control structure. We will soon learn that the flowcharts for the remaining control structures also contain (besides small circle symbols and flowlines) only rectangle symbols, to indicate the actions to be performed, and diamond symbols, to indicate decisions to be made. This type of flowchart represents the **action/decision model of programming**.

We can envision eight bins, each containing only the control structures of one of the eight types. These control structures are empty. Nothing is written in the rectangles or in the diamonds. The programmer's task, then, is to assemble a program from as many of each type of control structure as the algorithm demands, combining them in only two possible ways (stacking or nesting), then filling in the actions and decisions in a manner appropriate for the algorithm. We will discuss the variety of ways in which actions and decisions may be written.

7.6 `if...else` Selection Statement

The `if` selection statement performs an indicated action only when the condition evaluates to `true`; otherwise, the action is skipped. The **`if...else` selection** statement allows the programmer to specify that a different action is to be performed when the condition is `true` than when the condition is `false`. For example, the pseudocode statement

```

If student's grade is greater than or equal to 60
  Print "Passed"
Else
  Print "Failed"
  
```

prints `Passed` if the student's grade is greater than or equal to 60 and prints `Failed` if the student's grade is less than 60. In either case, after printing occurs, the next pseudocode statement in sequence (i.e., the next statement after the whole `if...else` structure) is performed. Note that the body of the `Else` part of the structure is also indented.



Good Programming Practice 7.2

Indent both body statements of an if...else statement.

The indentation convention you choose should be applied carefully throughout your programs (both in pseudocode and in JavaScript). It is difficult to read programs that do not use uniform spacing conventions.

The preceding pseudocode *If...Else* statement may be written in JavaScript as

```
if ( studentGrade >= 60 )
    document.writeln( "Passed" );
else
    document.writeln( "Failed" );
```

The flowchart shown in Fig. 7.4 illustrates the *if...else* selection statement's flow of control. Once again, note that the only symbols in the flowchart (besides small circles and arrows) are rectangles (for actions) and a diamond (for a decision). We continue to emphasize this action/decision model of computing. Imagine again a deep bin containing as many empty double-selection structures as might be needed to build a JavaScript algorithm. The programmer's job is to assemble the selection structures (by stacking and nesting) with other control structures required by the algorithm and to fill in the empty rectangles and empty diamonds with actions and decisions appropriate to the algorithm's implementation.

JavaScript provides an operator, called the **conditional operator** (`? :`), that is closely related to the *if...else* statement. The operator `? :` is JavaScript's only **ternary operator**—it takes three operands. The operands together with the `? :` form a **conditional expression**. The first operand is a boolean expression, the second is the value for the conditional expression if the expression evaluates to true and the third is the value for the conditional expression if the expression evaluates to false. For example, consider the following statement

```
document.writeln( studentGrade >= 60 ? "Passed" : "Failed" );
```

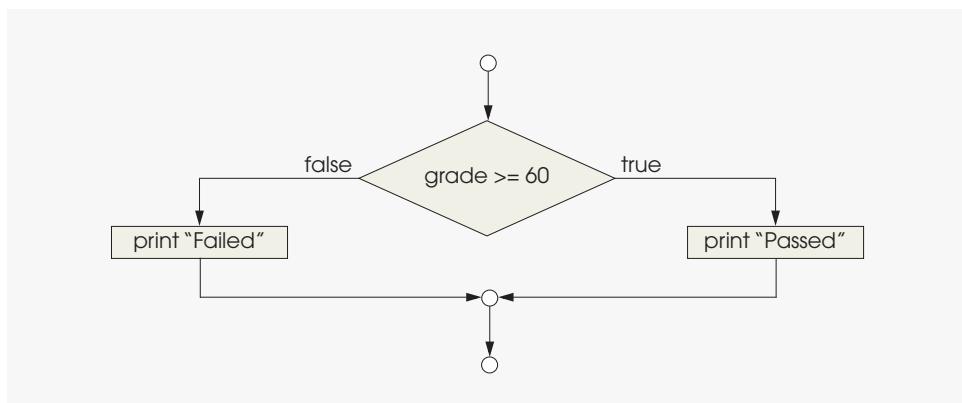


Fig. 7.4 | Flowcharting the double-selection *if...else* statement.

contains a conditional expression that evaluates to the string "Passed" if the condition `studentGrade >= 60` is true and evaluates to the string "Failed" if the condition is false. Thus, this statement with the conditional operator performs essentially the same operation as the preceding `if...else` statement. The precedence of the conditional operator is low, so the entire conditional expression is normally placed in parentheses to ensure that it evaluates correctly.

Nested `if...else` statements test for multiple cases by placing `if...else` statements inside `if...else` statements. For example, the following pseudocode statement indicates that the program should print A for exam grades greater than or equal to 90, B for grades in the range 80 to 89, C for grades in the range 70 to 79, D for grades in the range 60 to 69 and F for all other grades:

```
If student's grade is greater than or equal to 90
    Print "A"
Else
    If student's grade is greater than or equal to 80
        Print "B"
    Else
        If student's grade is greater than or equal to 70
            Print "C"
        Else
            If student's grade is greater than or equal to 60
                Print "D"
            Else
                Print "F"
```

This pseudocode may be written in JavaScript as

```
if ( studentGrade >= 90 )
    document.writeln( "A" );
else
    if ( studentGrade >= 80 )
        document.writeln( "B" );
    else
        if ( studentGrade >= 70 )
            document.writeln( "C" );
        else
            if ( studentGrade >= 60 )
                document.writeln( "D" );
            else
                document.writeln( "F" );
```

If `studentGrade` is greater than or equal to 90, all four conditions will be true, but only the `document.writeln` statement after the first test will execute. After that particular `document.writeln` executes, the `else` part of the outer `if...else` statements is skipped.



Good Programming Practice 7.3

If there are several levels of indentation, each level should be indented the same additional amount of space.

Most JavaScript programmers prefer to write the preceding `if` structure as

```
if ( grade >= 90 )
    document.writeln( "A" );
else if ( grade >= 80 )
    document.writeln( "B" );
else if ( grade >= 70 )
    document.writeln( "C" );
else if ( grade >= 60 )
    document.writeln( "D" );
else
    document.writeln( "F" );
```

The two forms are equivalent. The latter form is popular because it avoids the deep indentation of the code to the right. Such deep indentation often leaves little room on a line, forcing lines to be split and decreasing program readability.

It is important to note that the JavaScript interpreter always associates an `else` with the previous `if`, unless told to do otherwise by the placement of braces (`{}`). This situation is referred to as the **dangling-else problem**. For example,

```
if ( x > 5 )
    if ( y > 5 )
        document.writeln( "x and y are > 5" );
else
    document.writeln( "x is <= 5" );
```

appears to indicate with its indentation that if `x` is greater than 5, the `if` structure in its body determines whether `y` is also greater than 5. If so, the body of the nested `if` structure outputs the string "`x and y are > 5`". Otherwise, it *appears* that if `x` is not greater than 5, the `else` part of the `if...else` structure outputs the string "`x is <= 5`".

Beware! The preceding nested `if` statement does not execute as it appears. The interpreter actually interprets the preceding statement as

```
if ( x > 5 )
    if ( y > 5 )
        document.writeln( "x and y are > 5" );
    else
        document.writeln( "x is <= 5" );
```

in which the body of the first `if` statement is a nested `if...else` statement. This statement tests whether `x` is greater than 5. If so, execution continues by testing whether `y` is also greater than 5. If the second condition is true, the proper string—"`x and y are > 5`"—is displayed. However, if the second condition is false, the string "`x is <= 5`" is displayed, even though we know that `x` is greater than 5.

To force the preceding nested `if` statement to execute as it was intended originally, it must be written as follows:

```
if ( x > 5 )
{
    if ( y > 5 )
        document.writeln( "x and y are > 5" );
}
else
    document.writeln( "x is <= 5" );
```

The braces ({}) indicate to the interpreter that the second `if` statement is in the body of the first `if` statement and that the `else` is matched with the first `if` statement. In Exercises 7.21 and 7.22, you will investigate the dangling-`else` problem further.

The `if` selection statement expects only one statement in its body. To include several statements in an `if` statement's body, enclose the statements in braces ({ and }). This can also be done in the `else` section of an `if...else` statement. A set of statements contained within a pair of braces is called a **block**.



Software Engineering Observation 7.3

A block can be placed anywhere in a program that a single statement can be placed.



Software Engineering Observation 7.4

Unlike individual statements, a block does not end with a semicolon. However, each statement within the braces of a block should end with a semicolon.

The following example includes a block in the `else` part of an `if...else` statement:

```
if ( grade >= 60 )
    document.writeln( "Passed" );
else
{
    document.writeln( "Failed<br />" );
    document.writeln( "You must take this course again." );
}
```

In this case, if `grade` is less than 60, the program executes both statements in the body of the `else` and prints

```
Failed.
You must take this course again.
```

Note the braces surrounding the two statements in the `else` clause. These braces are important. Without them, the statement

```
document.writeln( "You must take this course again." );
```

would be outside the body of the `else` part of the `if` and would execute regardless of whether the `grade` is less than 60.



Common Programming Error 7.2

Forgetting one or both of the braces that delimit a block can lead to syntax errors or logic errors.

Syntax errors (e.g., when one brace in a block is left out of the program) are caught by the interpreter when it attempts to interpret the code containing the syntax error. A **logic error** (e.g., the one caused when both braces around a block are left out of the program) also has its effect at execution time. A **fatal logic error** causes a program to fail and terminate prematurely. A **nonfatal logic error** allows a program to continue executing, but the program produces incorrect results.



Good Programming Practice 7.4

Some programmers prefer to type the beginning and ending braces of blocks before typing the individual statements within the braces. This helps avoid omitting one or both of the braces.



Software Engineering Observation 7.5

Just as a block can be placed anywhere a single statement can be placed, it is also possible to have no statement at all (the empty statement) in such places. The empty statement is represented by placing a semicolon (;) where a statement would normally be.



Common Programming Error 7.3

Placing a semicolon after the condition in an if structure leads to a logic error in single-selection if structures and a syntax error in double-selection if structures (if the if part contains a non-empty body statement).

7.7 while Repetition Statement

A repetition structure (also known as a **loop**) allows the programmer to specify that a script is to repeat an action while some condition remains true. The pseudocode statement

*While there are more items on my shopping list
Purchase next item and cross it off my list*

describes the repetition that occurs during a shopping trip. The condition “there are more items on my shopping list” may be true or false. If it’s true, then the action “Purchase next item and cross it off my list” is performed. This action is performed repeatedly while the condition remains true. The statement(s) contained in the *While* repetition structure constitute its body. The body of a loop such as the *While* structure may be a single statement or a block. Eventually, the condition becomes false (i.e., when the last item on the shopping list has been purchased and crossed off the list). At this point, the repetition terminates, and the first pseudocode statement after the repetition structure executes.



Common Programming Error 7.4

If the body of a while statement never causes the while statement’s condition to become true, a logic error occurs. Normally, such a repetition structure will never terminate—an error called an **infinite loop**. Both Internet Explorer and Firefox show a dialog allowing the user to terminate a script that contains an infinite loop.



Common Programming Error 7.5

Remember that JavaScript is a case-sensitive language. In code, spelling the keyword while with an uppercase W, as in **While**, is a syntax error. All of JavaScript’s reserved keywords, such as **while**, **if** and **else**, contain only lowercase letters.

As an example of a **while** statement, consider a program segment designed to find the first power of 2 larger than 1000. Variable **product** begins with the value 2. The statement is as follows:

```
var product = 2;  
while ( product <= 1000 )  
    product = 2 * product;
```

When the **while** statement finishes executing, **product** contains the result 1024. The flowchart in Fig. 7.5 illustrates the flow of control of the preceding **while** repetition statement. Once again, note that (besides small circles and arrows) the flowchart contains only a rectangle symbol and a diamond symbol.

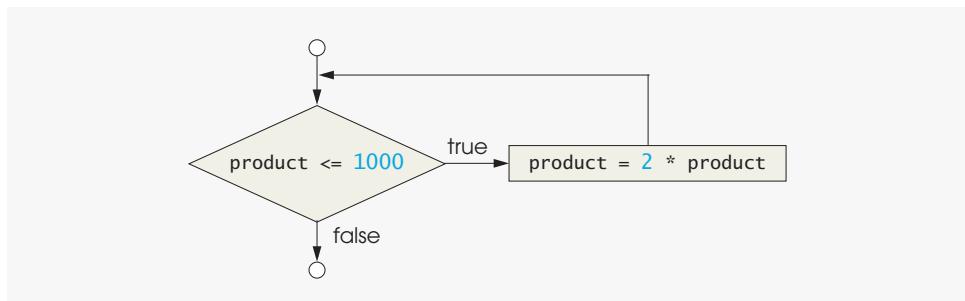


Fig. 7.5 | Flowcharting the `while` repetition statement.

When the script enters the `while` statement, `product` is 2. The script repeatedly multiplies variable `product` by 2, so `product` takes on the values 4, 8, 16, 32, 64, 128, 256, 512 and 1024 successively. When `product` becomes 1024, the condition `product <= 1000` in the `while` statement becomes `false`. This terminates the repetition, with 1024 as `product`'s final value. Execution continues with the next statement after the `while` statement. [Note: If a `while` statement's condition is initially `false`, the body statement(s) will never execute.]

The flowchart clearly shows the repetition. The flowline emerging from the rectangle wraps back to the decision, which the script tests each time through the loop until the decision eventually becomes `false`. At this point, the `while` statement exits, and control passes to the next statement in the program.

7.8 Formulating Algorithms: Counter-Controlled Repetition

To illustrate how to develop algorithms, we solve several variations of a class-averaging problem. Consider the following problem statement:

A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.

The class average is equal to the sum of the grades divided by the number of students (10 in this case). The algorithm for solving this problem on a computer must input each of the grades, perform the averaging calculation and display the result.

Let us use pseudocode to list the actions to execute and specify the order in which the actions should execute. We use **counter-controlled repetition** to input the grades one at a time. This technique uses a variable called a **counter** to control the number of times a set of statements executes. In this example, repetition terminates when the counter exceeds 10. In this section, we present a pseudocode algorithm (Fig. 7.6) and the corresponding program (Fig. 7.7). In the next section, we show how to develop pseudocode algorithms. Counter-controlled repetition often is called **definite repetition**, because the number of repetitions is known before the loop begins executing.

Note the references in the algorithm to a total and a counter. A **total** is a variable in which a script accumulates the sum of a series of values. A counter is a variable a script uses to count—in this case, to count the number of grades entered. Variables that store totals should normally be initialized to zero before they are used in a program.

```
Set total to zero
Set grade counter to one

While grade counter is less than or equal to ten
    Input the next grade
    Add the grade into the total
    Add one to the grade counter

Set the class average to the total divided by ten
Print the class average
```

Fig. 7.6 | Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem.

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.7: average.html -->
6  <!-- Counter-controlled repetition to calculate a class average. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Class Average Program</title>
10         <script type = "text/javascript">
11             <!--
12                 var total; // sum of grades
13                 var gradeCounter; // number of grades entered
14                 var grade; // grade typed by user (as a string)
15                 var gradeValue; // grade value (converted to integer)
16                 var average; // average of all grades
17
18                 // Initialization Phase
19                 total = 0; // clear total
20                 gradeCounter = 1; // prepare to loop
21
22                 // Processing Phase
23                 while ( gradeCounter <= 10 ) // loop 10 times
24                 {
25
26                     // prompt for input and read grade from user
27                     grade = window.prompt( "Enter integer grade:", "0" );
28
29                     // convert grade from a string to an integer
30                     gradeValue = parseInt( grade );
31
32                     // add gradeValue to total
33                     total = total + gradeValue;
34             </script>
35     </head>
36     <body>
37         <h1>Grade Average</h1>
38         <p>The average is: <span id="average"></span></p>
39     </body>
40 </html>
```

Fig. 7.7 | Counter-controlled repetition to calculate a class average. (Part I of 2.)

```

35          // add 1 to gradeCounter
36          gradeCounter = gradeCounter + 1;
37      } // end while
38
39      // Termination Phase
40      average = total / 10; // calculate the average
41
42      // display average of exam grades
43      document.writeln(
44          "<h1>Class average is " + average + "</h1> ");
45      // -->
46  </script>
47  </head>
48  <body>
49      <p>Click Refresh (or Reload) to run the script again<p>
50  </body>
51 </html>

```



This dialog is displayed 10 times. User input is 100, 88, 93, 55, 68, 77, 83, 95, 73 and 62.



Fig. 7.7 | Counter-controlled repetition to calculate a class average. (Part 2 of 2.)

Lines 12–16 declare variables `total`, `gradeCounter`, `grade`, `gradeValue`, `average`. The variable `grade` will store the string the user types into the prompt dialog. The variable `gradeValue` will store the integer value of the grade the user enters into the prompt dialog.

Lines 19–20 are assignment statements that initialize `total` to 0 and `gradeCounter` to 1. Note that variables `total` and `gradeCounter` are initialized before they are used in a calculation.



Common Programming Error 7.6

Not initializing a variable that will be used in a calculation results in a logic error that produces the value NaN—Not a Number. You must initialize the variable before it is used in a calculation.

Line 23 indicates that the `while` statement continues iterating while the value of `gradeCounter` is less than or equal to 10. Line 27 corresponds to the pseudocode statement “*Input the next grade.*” The statement displays a prompt dialog with the prompt “Enter integer grade:” on the screen.

After the user enters the grade, line 30 converts it from a string to an integer. We must convert the string to an integer in this example; otherwise, the addition statement in line 33 will be a string-concatenation statement rather than a numeric sum.

Next, the program updates the `total` with the new `gradeValue` entered by the user. Line 33 adds `gradeValue` to the previous value of `total` and assigns the result to `total`. This statement seems a bit strange, because it does not follow the rules of algebra. Keep in mind that JavaScript operator precedence evaluates the addition (+) operation before the assignment (=) operation. The value of the expression on the right side of the assignment operator always replaces the value of the variable on the left side of the assignment operator.

The program now is ready to increment the variable `gradeCounter` to indicate that a grade has been processed and to read the next grade from the user. Line 36 adds 1 to `gradeCounter`, so the condition in the `while` statement will eventually become `false` and terminate the loop. After this statement executes, the program continues by testing the condition in the `while` statement in line 23. If the condition is still true, the statements in lines 27–36 repeat. Otherwise the program continues execution with the first statement in sequence after the body of the loop (i.e., line 40).

Line 40 assigns the results of the average calculation to variable `average`. Lines 43–44 write a line of XHTML text in the document that displays the string “Class average is” followed by the value of variable `average` as an `<h1>` element in the browser.

Execute the script in a web browser by double clicking the XHTML document (from Windows Explorer). This script parses any user input as an integer. In the sample program execution in Fig. 7.7, the sum of the values entered (100, 88, 93, 55, 68, 77, 83, 95, 73 and 62) is 794. Although the treats all input as integers, the averaging calculation in the program does not produce an integer. Rather, the calculation produces a **floating-point number** (i.e., a number containing a decimal point). The average of the 10 integers input by the user in this example is 79.4.



Software Engineering Observation 7.6

If the string passed to `parseInt` contains a floating-point numeric value, `parseInt` simply truncates the floating-point part. For example, the string “27.95” results in the integer 27, and the string “-123.45” results in the integer -123. If the string passed to `parseInt` is not a numeric value, `parseInt` returns `NaN` (not a number).

JavaScript actually represents all numbers as floating-point numbers in memory. Floating-point numbers often develop through division, as shown in this example. When we divide 10 by 3, the result is 3.3333333..., with the sequence of 3's repeating infinitely. The computer allocates only a fixed amount of space to hold such a value, so the stored floating-point value can be only an approximation. Although floating-point numbers are not always 100 percent precise, they have numerous applications. For example, when we speak of a “normal” body temperature of 98.6, we do not need to be precise to a large number of digits. When we view the temperature on a thermometer and read it as 98.6, it may actually be 98.5999473210643. The point here is that few applications require high-

precision floating-point values, so calling this number simply 98.6 is fine for most applications.



Common Programming Error 7.7

Using floating-point numbers in a manner that assumes they are represented precisely can lead to incorrect results. Real numbers are represented only approximately by computers. For example, no fixed-size floating-point representation of π can ever be precise, because π is a transcendental number whose value cannot be expressed as digits in a finite amount of space.

7.9 Formulating Algorithms: Sentinel-Controlled Repetition

Let us generalize the class-average problem. Consider the following problem:

Develop a class-averaging program that will process an arbitrary number of grades each time the program is run.

In the first class-average example, the number of grades (10) was known in advance. In this example, no indication is given of how many grades the user will enter. The program must process an arbitrary number of grades. How can the program determine when to stop the input of grades? How will it know when to calculate and display the class average?

One way to solve this problem is to use a special value called a **sentinel value** (also called a **signal value**, a **dummy value** or a **flag value**) to indicate the end of data entry. The user types in grades until all legitimate grades have been entered. Then the user types the sentinel value to indicate that the last grade has been entered. Sentinel-controlled repetition is often called **indefinite repetition**, because the number of repetitions is not known before the loop begins executing.

Clearly, one must choose a sentinel value that cannot be confused with an acceptable input value. -1 is an acceptable sentinel value for this problem because grades on a quiz are normally nonnegative integers from 0 to 100. Thus, an execution of the class-average program might process a stream of inputs such as 95, 96, 75, 74, 89 and -1 . The program would compute and print the class average for the grades 95, 96, 75, 74 and 89 (-1 is the sentinel value, so it should not enter into the average calculation).



Common Programming Error 7.8

Choosing a sentinel value that is also a legitimate data value results in a logic error and may prevent a sentinel-controlled loop from terminating properly.

We approach the class-average program with a technique called **top-down, stepwise refinement**, a technique that is essential to the development of well-structured algorithms. We begin with a pseudocode representation of the **top**:

Determine the class average for the quiz

The top is a single statement that conveys the program's overall purpose. As such, the top is, in effect, a complete representation of a program. Unfortunately, the top rarely conveys sufficient detail from which to write the JavaScript algorithm. Therefore we must begin a refinement process. First, we divide the top into a series of smaller tasks and list them in the order in which they need to be performed, creating the following **first refinement**:

Initialize variables

Input, sum up and count the quiz grades

Calculate and print the class average

Here, only the sequence structure is used; the steps listed are to be executed in order, one after the other.



Software Engineering Observation 7.7

Each refinement, as well as the top itself, is a complete specification of the algorithm; only the level of detail varies.

To proceed to the next level of refinement (the **second refinement**), we commit to specific variables. We need a running total of the numbers, a count of how many numbers have been processed, a variable to receive the string representation of each grade as it is input, a variable to store the value of the grade after it is converted to an integer and a variable to hold the calculated average. The pseudocode statement

Initialize variables

may be refined as follows:

Initialize total to zero

Initialize gradeCounter to zero

Note that only the variables *total* and *gradeCounter* are initialized before they are used; the variables *average*, *grade* and *gradeValue* (for the calculated average, the user input and the integer representation of the *grade*, respectively) need not be initialized, because their values are determined as they are calculated or input.

The pseudocode statement

Input, sum up and count the quiz grades

requires a repetition structure (a loop) that successively inputs each grade. We do not know in advance how many grades are to be processed, so we will use sentinel-controlled repetition. The user will enter legitimate grades, one at a time. After entering the last legitimate grade, the user will enter the sentinel value. The program will test for the sentinel value after the user enters each grade and will terminate the loop when the sentinel value is encountered. The second refinement of the preceding pseudocode statement is then

Input the first grade (possibly the sentinel)

While the user has not as yet entered the sentinel

Add this grade into the running total

Add one to the grade counter

Input the next grade (possibly the sentinel)

Note that in pseudocode, we do not use braces around the pseudocode that forms the body of the *While* structure. We simply indent the pseudocode under the *While*, to show that it belongs to the body of the *While*. Remember, pseudocode is only an informal program-development aid.

The pseudocode statement

Calculate and print the class average

may be refined as follows:

If the counter is not equal to zero

Set the average to the total divided by the counter

Print the average

Else

Print "No grades were entered"

Note that we are testing for the possibility of **division by zero**—a logic error that, if undetected, would cause the program to produce invalid output. The complete second refinement of the pseudocode algorithm for the class-average problem is shown in Fig. 7.8.



Error-Prevention Tip 7.1

When performing division by an expression whose value could be zero, explicitly test for this case, and handle it appropriately in your program (e.g., by printing an error message) rather than allowing the division by zero to occur.



Good Programming Practice 7.5

Include completely blank lines in pseudocode programs to make the pseudocode more readable. The blank lines separate pseudocode control structures and separate the program phases.



Software Engineering Observation 7.8

Many algorithms can be divided logically into three phases: an initialization phase that initializes the program variables, a processing phase that inputs data values and adjusts program variables accordingly, and a termination phase that calculates and prints the results.

The pseudocode algorithm in Fig. 7.8 solves the more general class-averaging problem. This algorithm was developed after only two refinements. Sometimes more refinements are necessary.

Initialize total to zero

Initialize gradeCounter to zero

Input the first grade (possibly the sentinel)

While the user has not as yet entered the sentinel

Add this grade into the running total

Add one to the grade counter

Input the next grade (possibly the sentinel)

If the counter is not equal to zero

Set the average to the total divided by the counter

Print the average

Else

Print "No grades were entered"

Fig. 7.8 | Sentinel-controlled repetition to solve the class-average problem.



Software Engineering Observation 7.9

The programmer terminates the top-down, stepwise refinement process after specifying the pseudocode algorithm in sufficient detail for the programmer to convert the pseudocode to a JavaScript program. Then, implementing the JavaScript program will normally be straightforward.



Good Programming Practice 7.6

When converting a pseudocode program to JavaScript, keep the pseudocode in the JavaScript program as comments.



Software Engineering Observation 7.10

Experience has shown that the most difficult part of solving a problem on a computer is developing the algorithm for the solution. Once a correct algorithm is specified, the process of producing a working JavaScript program from the algorithm is normally straightforward.



Software Engineering Observation 7.11

Many experienced programmers write programs without ever using program-development tools like pseudocode. As they see it, their ultimate goal is to solve the problem on a computer, and writing pseudocode merely delays the production of final outputs. Although this approach may work for simple and familiar problems, it can lead to serious errors in large, complex projects.

Figure 7.9 shows the JavaScript program and a sample execution. Although each grade is an integer, the averaging calculation is likely to produce a number with a decimal point (a real number).

In this example, we see that control structures may be stacked on top of one another (in sequence) just as a child stacks building blocks. The `while` statement (lines 31–45) is followed immediately by an `if...else` statement (lines 48–57) in sequence. Much of the code in this program is identical to the code in Fig. 7.7, so we concentrate in this example on the new features.

Line 21 initializes `gradeCounter` to 0, because no grades have been entered yet. Remember that the program uses sentinel-controlled repetition. To keep an accurate record of the number of grades entered, the script increments `gradeCounter` only after processing a valid grade value.

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.9: average2.html -->
6  <!-- Sentinel-controlled repetition to calculate a class average. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Class Average Program: Sentinel-controlled Repetition</title>
10
11         <script type = "text/javascript">
12             <!--
```

Fig. 7.9 | Sentinel-controlled repetition to calculate a class average. (Part I of 3.)

```
13     var total; // sum of grades
14     var gradeCounter; // number of grades entered
15     var grade; // grade typed by user (as a string)
16     var gradeValue; // grade value (converted to integer)
17     var average; // average of all grades
18
19     // Initialization phase
20     total = 0; // clear total
21     gradeCounter = 0; // prepare to loop
22
23     // Processing phase
24     // prompt for input and read grade from user
25     grade = window.prompt(
26         "Enter Integer Grade, -1 to Quit:", "0" );
27
28     // convert grade from a string to an integer
29     gradeValue = parseInt( grade );
30
31     while ( gradeValue != -1 )
32     {
33         // add gradeValue to total
34         total = total + gradeValue;
35
36         // add 1 to gradeCounter
37         gradeCounter = gradeCounter + 1;
38
39         // prompt for input and read grade from user
40         grade = window.prompt(
41             "Enter Integer Grade, -1 to Quit:", "0" );
42
43         // convert grade from a string to an integer
44         gradeValue = parseInt( grade );
45     } // end while
46
47     // Termination phase
48     if ( gradeCounter != 0 )
49     {
50         average = total / gradeCounter;
51
52         // display average of exam grades
53         document.writeln(
54             "<h1>Class average is " + average + "</h1>" );
55     } // end if
56     else
57         document.writeln( "<p>No grades were entered</p>" );
58     // -->
59     </script>
60     </head>
61     <body>
62         <p>Click Refresh (or Reload) to run the script again</p>
63     </body>
64 </html>
```

Fig. 7.9 | Sentinel-controlled repetition to calculate a class average. (Part 2 of 3.)

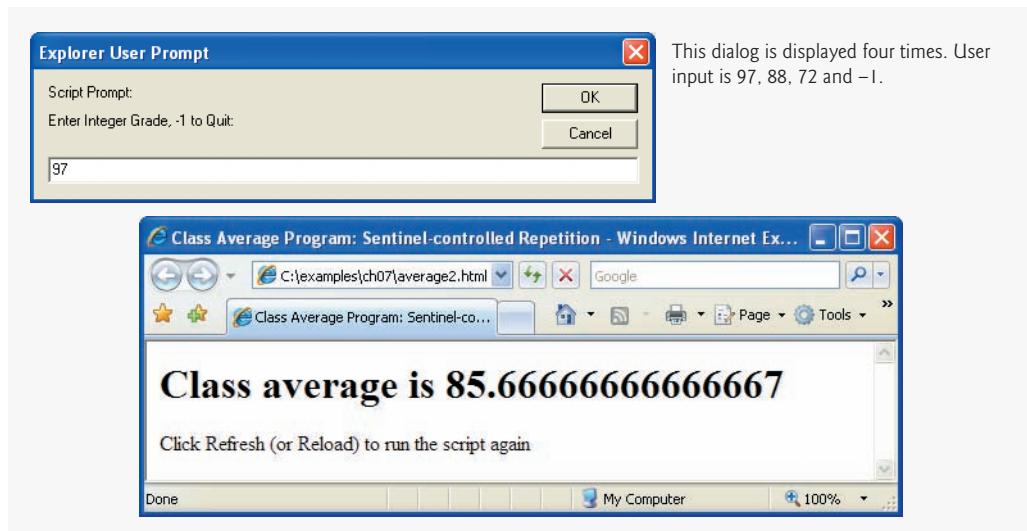


Fig. 7.9 | Sentinel-controlled repetition to calculate a class average. (Part 3 of 3.)

Note the difference in program logic for sentinel-controlled repetition as compared with the counter-controlled repetition in Fig. 7.7. In counter-controlled repetition, we read a value from the user during each iteration of the `while` statement's body for the specified number of iterations. In sentinel-controlled repetition, we read one value (lines 25–26) and convert it to an integer (line 29) before the program reaches the `while` statement. The script uses this value to determine whether the program's flow of control should enter the body of the `while` statement. If the `while` statement's condition is `false` (i.e., the user typed the sentinel as the first grade), the script ignores the body of the `while` statement (i.e., no grades were entered). If the condition is `true`, the body begins execution and processes the value entered by the user (i.e., adds the value to the `total` in line 34). After processing the value, the script increments `gradeCounter` by 1 (line 37), inputs the next grade from the user (lines 40–41) and converts the grade to an integer (line 44), before the end of the `while` statement's body. When the script reaches the closing right brace (`}`) of the body in line 45, execution continues with the next test of the condition of the `while` statement (line 31), using the new value just entered by the user to determine whether the `while` statement's body should execute again. Note that the next value always is input from the user immediately before the script evaluates the condition of the `while` statement. This order allows us to determine whether the value just entered by the user is the sentinel value *before* processing it (i.e., adding it to the `total`). If the value entered is the sentinel value, the `while` statement terminates and the script does not add the value to the `total`.



Good Programming Practice 7.7

In a sentinel-controlled loop, the prompts requesting data entry should explicitly remind the user what the sentinel value is.

Note the block in the `while` loop in Fig. 7.9 (lines 32–45). Without the braces, the last three statements in the body of the loop would fall outside of the loop, causing the computer to interpret the code incorrectly, as follows:

```

while ( gradeValue != -1 )
    // add gradeValue to total
    total = total + gradeValue;

// add 1 to gradeCounter
gradeCounter = gradeCounter + 1;

// prompt for input and read grade from user
grade = window.prompt(
    "Enter Integer Grade, -1 to Quit:", "0" );

// convert grade from a string to an integer
gradeValue = parseInt( grade );

```

This interpretation would cause an infinite loop in the program if the user does not input the sentinel -1 as the first input value in lines 25–26 (i.e., before the `while` statement).



Common Programming Error 7.9

Omitting the braces that delineate a block can lead to logic errors such as infinite loops.

7.10 Formulating Algorithms: Nested Control Statements

Let us work through another complete problem. We once again formulate the algorithm using pseudocode and top-down, stepwise refinement, and write a corresponding JavaScript program.

Consider the following problem statement:

A college offers a course that prepares students for the state licensing exam for real estate brokers. Last year, several of the students who completed this course took the licensing exam. Naturally, the college wants to know how well its students performed. You have been asked to write a program to summarize the results. You have been given a list of these 10 students. Next to each name is written a 1 if the student passed the exam and a 2 if the student failed.

Your program should analyze the results of the exam as follows:

1. *Input each test result (i.e., a 1 or a 2). Display the message “Enter result” on the screen each time the program requests another test result.*
2. *Count the number of test results of each type.*
3. *Display a summary of the test results indicating the number of students who passed and the number of students who failed.*
4. *If more than eight students passed the exam, print the message “Raise tuition.”*

After reading the problem statement carefully, we make the following observations about the problem:

1. The program must process test results for 10 students. A counter-controlled loop will be used.
2. Each test result is a number—either a 1 or a 2. Each time the program reads a test result, the program must determine whether the number is a 1 or a 2. We test for a 1 in our algorithm. If the number is not a 1, we assume that it is a 2. (An exercise at the end of the chapter considers the consequences of this assumption.)

3. Two counters are used to keep track of the exam results—one to count the number of students who passed the exam and one to count the number of students who failed the exam.

After the program processes all the results, it must decide whether more than eight students passed the exam. Let us proceed with top-down, stepwise refinement. We begin with a pseudocode representation of the top:

Analyze exam results and decide whether tuition should be raised

Once again, it is important to emphasize that the top is a complete representation of the program, but that several refinements are necessary before the pseudocode can be evolved naturally into a JavaScript program. Our first refinement is as follows:

Initialize variables

Input the ten exam grades and count passes and failures

Print a summary of the exam results and decide whether tuition should be raised

Here, too, even though we have a complete representation of the entire program, further refinement is necessary. We now commit to specific variables. Counters are needed to record the passes and failures; a counter will be used to control the looping process, and a variable is needed to store the user input. The pseudocode statement

Initialize variables

may be refined as follows:

Initialize passes to zero

Initialize failures to zero

Initialize student to one

Note that only the counters for the number of passes, the number of failures and the number of students are initialized. The pseudocode statement

Input the ten quiz grades and count passes and failures

requires a loop that successively inputs the result of each exam. Here, it is known in advance that there are precisely 10 exam results, so counter-controlled looping is appropriate. Inside the loop (i.e., *nested* within the loop), a double-selection structure will determine whether each exam result is a pass or a failure and will increment the appropriate counter accordingly. The refinement of the preceding pseudocode statement is then

While student counter is less than or equal to ten

Input the next exam result

If the student passed

Add one to passes

Else

Add one to failures

Add one to student counter

Note the use of blank lines to set off the *If...Else* control structure to improve program readability. The pseudocode statement

Print a summary of the exam results and decide whether tuition should be raised

may be refined as follows:

*Print the number of passes
Print the number of failures
If more than eight students passed
 Print "Raise tuition"*

The complete second refinement appears in Fig. 7.10. Note that blank lines are also used to set off the *While* statement for program readability.

This pseudocode is now refined sufficiently for conversion to JavaScript. The JavaScript program and two sample executions are shown in Fig. 7.11.

```
Initialize passes to zero
Initialize failures to zero
Initialize student to one

While student counter is less than or equal to ten
    Input the next exam result
    If the student passed
        Add one to passes
    Else
        Add one to failures
    Add one to student counter
    Print the number of passes
    Print the number of failures
    If more than eight students passed
        Print "Raise tuition"
```

Fig. 7.10 | Examination-results problem pseudocode.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 7.11: analysis.html -->
6 <!-- Examination-results calculation. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Analysis of Examination Results</title>
10    <script type = "text/javascript">
11      <!--
12      // initializing variables in declarations
13      var passes = 0; // number of passes
14      var failures = 0; // number of failures
15      var student = 1; // student counter
16      var result; // one exam result
17
```

Fig. 7.11 | Examination-results calculation. (Part I of 3.)

```

18     // process 10 students; counter-controlled loop
19     while ( student <= 10 )
20     {
21         result = window.prompt( "Enter result (1=pass,2=fail)", "0" );
22
23         if ( result == "1" )
24             passes = passes + 1;
25         else
26             failures = failures + 1;
27
28         student = student + 1;
29     } // end while
30
31     // termination phase
32     document.writeln( "<h1>Examination Results</h1>" );
33     document.writeln(
34         "Passed: " + passes + "<br />Failed: " + failures );
35
36     if ( passes > 8 )
37         document.writeln( "<br />Raise Tuition" );
38     // -->
39     </script>
40 </head>
41 <body>
42     <p>Click Refresh (or Reload) to run the script again</p>
43 </body>
44 </html>

```



This dialog is displayed 10 times. User input is 1, 2, 1, 1, 1, 1, 1, 1 and 1.

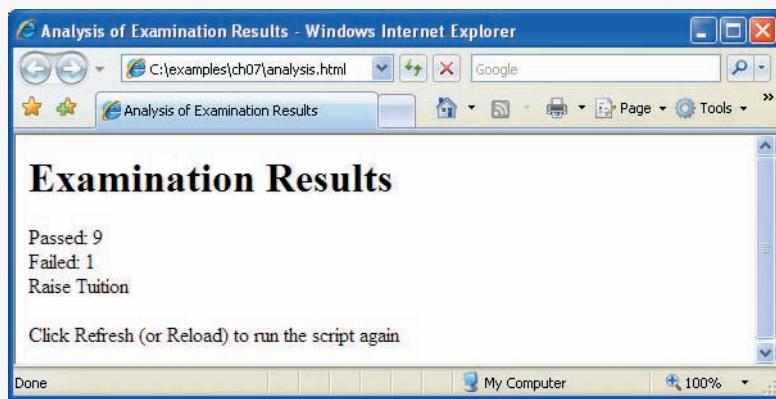


Fig. 7.11 | Examination-results calculation. (Part 2 of 3.)

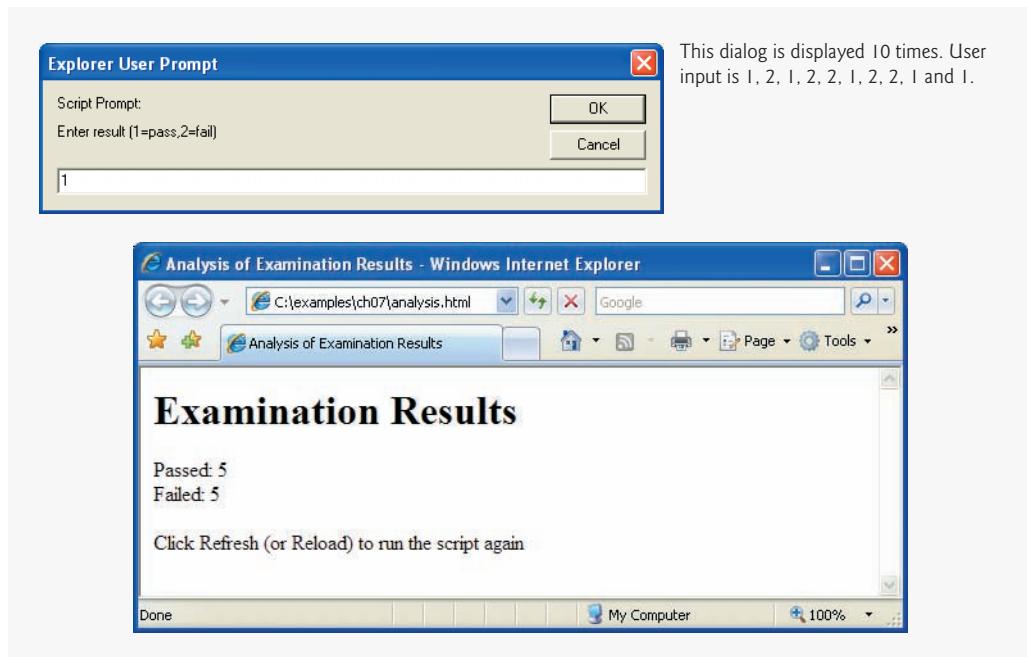


Fig. 7.11 | Examination-results calculation. (Part 3 of 3.)

Lines 13–16 declare the variables used to process the examination results. Note that JavaScript allows variable initialization to be incorporated into declarations (`passes` is assigned 0, `failures` is assigned 0 and `student` is assigned 1). Some programs may require reinitialization at the beginning of each repetition; such reinitialization would normally occur in assignment statements.

The processing of the exam results occurs in the `while` statement in lines 19–29. Note that the `if...else` statement in lines 23–26 in the loop tests only whether the exam result was 1; it assumes that all other exam results are 2. Normally, you should validate the values input by the user (i.e., determine whether the values are correct). In the exercises, we ask you to modify this example to validate the input values to ensure that they are either 1 or 2.



Good Programming Practice 7.8

When inputting values from the user, validate the input to ensure that it is correct. If an input value is incorrect, prompt the user to input the value again.

7.11 Assignment Operators

JavaScript provides several assignment operators (called **compound assignment operators**) for abbreviating assignment expressions. For example, the statement

```
c = c + 3;
```

can be abbreviated with the **addition assignment operator**, `+=`, as

```
c += 3;
```

The `+=` operator adds the value of the expression on the right of the operator to the value of the variable on the left of the operator and stores the result in the variable on the left of the operator. Any statement of the form

```
variable = variable operator expression;
```

where *operator* is one of the binary operators `+`, `-`, `*`, `/` or `%` (or others we will discuss later in the text), can be written in the form

```
variable operator = expression;
```

Thus, the assignment `c += 3` adds 3 to `c`. Figure 7.12 shows the arithmetic assignment operators, sample expressions using these operators and explanations of the meaning of the operators.



Performance Tip 7.1

Programmers can write programs that execute a bit faster when the arithmetic assignment operators are used, because the variable on the left side of the assignment does not have to be evaluated twice.



Performance Tip 7.2

Many of the performance tips we mention in this text result in only nominal improvements, so the reader may be tempted to ignore them. Significant performance improvement often is realized when a supposedly nominal improvement is placed in a loop that may repeat a large number of times.

Assignment operator	Initial value of variable	Sample expression	Explanation	Assigns
<code>+=</code>	<code>c = 3</code>	<code>c += 7</code>	<code>c = c + 7</code>	10 to <code>c</code>
<code>-=</code>	<code>d = 5</code>	<code>d -= 4</code>	<code>d = d - 4</code>	1 to <code>d</code>
<code>*=</code>	<code>e = 4</code>	<code>e *= 5</code>	<code>e = e * 5</code>	20 to <code>e</code>
<code>/=</code>	<code>f = 6</code>	<code>f /= 3</code>	<code>f = f / 3</code>	2 to <code>f</code>
<code>%=</code>	<code>g = 12</code>	<code>g %= 9</code>	<code>g = g % 9</code>	3 to <code>g</code>

Fig. 7.12 | Arithmetic assignment operators.

7.12 Increment and Decrement Operators

JavaScript provides the unary **increment operator** (`++`) and **decrement operator** (`--`) (summarized in Fig. 7.13). If a variable `c` is incremented by 1, the increment operator, `++`, can be used rather than the expression `c = c + 1` or `c += 1`. If an increment or decrement operator is placed before a variable, it is referred to as the **preincrement** or **predecrement operator**, respectively. If an increment or decrement operator is placed after a variable, it is referred to as the **postincrement** or **postdecrement operator**, respectively.

Operator	Example	Called	Explanation
<code>++</code>	<code>++a</code>	preincrement	Increment <code>a</code> by 1, then use the new value of <code>a</code> in the expression in which <code>a</code> resides.
<code>++</code>	<code>a++</code>	postincrement	Use the current value of <code>a</code> in the expression in which <code>a</code> resides, then increment <code>a</code> by 1.
<code>--</code>	<code>--b</code>	predecrement	Decrement <code>b</code> by 1, then use the new value of <code>b</code> in the expression in which <code>b</code> resides.
<code>--</code>	<code>b--</code>	postdecrement	Use the current value of <code>b</code> in the expression in which <code>b</code> resides, then decrement <code>b</code> by 1.

Fig. 7.13 | Increment and decrement operators.



Error-Prevention Tip 7.2

The predecrement and postdecrement JavaScript operators cause the W3C XHTML Validator to incorrectly report errors. The validator attempts to interpret the decrement operator as part of an XHTML comment tag (`<!--` or `-->`). You can avoid this problem by using the subtraction assignment operator (`-=`) to subtract one from a variable. Note that the validator may report many more (nonexistent) errors once it improperly parses the decrement operator.

Preincrementing (or predecrementing) a variable causes the program to increment (decrement) the variable by 1, then use the new value of the variable in the expression in which it appears. Postincrementing (postdecrementing) the variable causes the program to use the current value of the variable in the expression in which it appears, then increment (decrement) the variable by 1.

The script in Fig. 7.14 demonstrates the difference between the preincrementing version and the postincrementing version of the `++` increment operator. Postincrementing the variable `c` causes it to be incremented after it is used in the `document.writeln` method call (line 18). Preincrementing the variable `c` causes it to be incremented before it is used in the `document.writeln` method call (line 25). The program displays the value of `c` before and after the `++` operator is used. The decrement operator (`--`) works similarly.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 7.14: increment.html -->
6 <!-- Preincrementing and Postincrementing. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Preincrementing and Postincrementing</title>
10    <script type = "text/javascript">
11      <!--
12        var c;
```

Fig. 7.14 | Preincrementing and postincrementing. (Part 1 of 2.)

```

13
14     c = 5;
15     document.writeln( "<h3>Postincrementing</h3>" );
16     document.writeln( c ); // prints 5
17     // prints 5 then increments
18     document.writeln( "<br />" + c++ );
19     document.writeln( "<br />" + c ); // prints 6
20
21     c = 5;
22     document.writeln( "<h3>Preincrementing</h3>" );
23     document.writeln( c ); // prints 5
24     // increments then prints 6
25     document.writeln( "<br />" + ++c );
26     document.writeln( "<br />" + c ); // prints 6
27     // -->
28   </script>
29 </head><body></body>
30 </html>
```

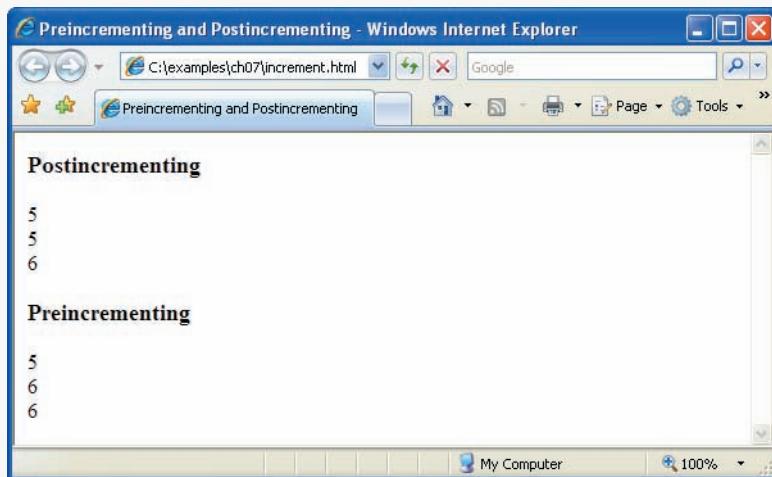


Fig. 7.14 | Preincrementing and postincrementing. (Part 2 of 2.)



Good Programming Practice 7.9

For readability, unary operators should be placed next to their operands, with no intervening spaces.

The three assignment statements in Fig. 7.11 (lines 24, 26 and 28, respectively),

```

passes = passes + 1;
failures = failures + 1;
student = student + 1;
```

can be written more concisely with assignment operators as

```

passes += 1;
failures += 1;
student += 1;
```

with preincrement operators as

```
++passes;
++failures;
++student;
```

or with postincrement operators as

```
passes++;
failures++;
student++;
```

It is important to note here that when incrementing or decrementing a variable in a statement by itself, the preincrement and postincrement forms have the same effect, and the predecrement and postdecrement forms have the same effect. It is only when a variable appears in the context of a larger expression that preincrementing the variable and postincrementing the variable have different effects. Predecrementing and postdecrementing behave similarly.



Common Programming Error 7.10

Attempting to use the increment or decrement operator on an expression other than a **left-hand-side expression**—commonly called an **lvalue**—is a syntax error. A left-hand-side expression is a variable or expression that can appear on the left side of an assignment operation. For example, writing `++(x + 1)` is a syntax error, because `(x + 1)` is not a left-hand-side expression.

Figure 7.15 lists the precedence and associativity of the operators introduced to this point. The operators are shown top to bottom in decreasing order of precedence. The second column describes the associativity of the operators at each level of precedence. Notice that the conditional operator (`?:`), the unary operators increment (`++`) and decrement (`--`) and the assignment operators `=`, `+=`, `-=`, `*=`, `/=` and `%=` associate from right to left. All other operators in the operator precedence table (Fig. 7.15) associate from left to right. The third column names the groups of operators.

Operator	Associativity	Type
<code>++ --</code>	right to left	unary
<code>* / %</code>	left to right	multiplicative
<code>+ -</code>	left to right	additive
<code>< <= > >=</code>	left to right	relational
<code>== !=</code>	left to right	equality
<code>?:</code>	right to left	conditional
<code>= += -= *= /= %=</code>	right to left	assignment

Fig. 7.15 | Precedence and associativity of the operators discussed so far.

7.13 Wrap-Up

In this chapter, we introduced the concept of algorithms, and explained how to introduce structure into your programs. We used pseudocode and flowcharts to represent algorithms and demonstrated how to translate them into control structures, which form the basis of all programs. We explored selection and repetition statements, and how to integrate them to make decisions and repeat statements in your code. We also introduced assignment, increment and decrement operators, as well as different types of errors that can result from incorrect implementation of control statements.

7.14 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 and more. Also check out the tutorials for all skill levels, from introductory to advanced. Be sure to visit the related Resource Centers on XHTML (www.deitel.com/xhtml1/) and CSS 2.1 (www.deitel.com/css21/).

Summary

Section 7.2 Algorithms

- Any computable problem can be solved by executing a series of actions in a specific order.
- A procedure for solving a problem in terms of the actions to execute and the order in which the actions are to execute is called an algorithm.
- Specifying the order in which statements are to be executed in a computer program is called program control.

Section 7.3 Pseudocode

- Pseudocode is an artificial and informal language that helps programmers develop algorithms.
- Carefully prepared pseudocode may be converted easily to a corresponding JavaScript program.
- Pseudocode normally describes only executable statements—the actions that are performed when the program is converted from pseudocode to JavaScript and executed.

Section 7.4 Control Structures

- Normally, statements in a program execute one after the other, in the order in which they are written. This process is called sequential execution.
- Various JavaScript statements enable the programmer to specify that the next statement to be executed may be other than the next one in sequence. This process is called transfer of control.
- All programs can be written in terms of only three control structures, namely, the sequence structure, the selection structure and the repetition structure.
- A flowchart is a graphical representation of an algorithm or of a portion of an algorithm. Flowcharts are drawn using certain special-purpose symbols, such as rectangles, diamonds, ovals and small circles; these symbols are connected by arrows called flowlines, which indicate the order in which the actions of the algorithm execute.
- JavaScript provides three selection structures. The `if` statement either performs (selects) an action if a condition is true or skips the action if the condition is false. The `if...else` statement performs an action if a condition is true and performs a different action if the condition is false.

The `switch` statement performs one of many different actions, depending on the value of an expression.

- JavaScript provides four repetition statements, namely, `while`, `do...while`, `for` and `for...in`.
- Keywords cannot be used as identifiers (e.g., for variable names).
- Single-entry/single-exit control structures make it easy to build programs. Control structures are attached to one another by connecting the exit point of one control structure to the entry point of the next. This procedure is called control-structure stacking. There is only one other way control structures may be connected: control-structure nesting.

Section 7.5 if Selection Statement

- The JavaScript interpreter ignores white-space characters: blanks, tabs and newlines used for indentation and vertical spacing. Programmers insert white-space characters to enhance program clarity.
- A decision can be made on any expression that evaluates to a value of JavaScript's boolean type (i.e., any expression that evaluates to `true` or `false`).
- The indentation convention you choose should be carefully applied throughout your programs. It is difficult to read programs that do not use uniform spacing conventions.

Section 7.6 if...else Selection Statement

- The conditional operator (`?:`) is closely related to the `if...else` statement. Operator `?:` is JavaScript's only ternary operator—it takes three operands. The operands together with the `?:` operator form a conditional expression. The first operand is a boolean expression, the second is the value for the conditional expression if the boolean expression evaluates to `true` and the third is the value for the conditional expression if the boolean expression evaluates to `false`.
- Nested `if...else` statements test for multiple cases by placing `if...else` statements inside other `if...else` structures.
- The JavaScript interpreter always associates an `else` with the previous `if`, unless told to do otherwise by the placement of braces (`{}`).
- The `if` selection statement expects only one statement in its body. To include several statements in the body of an `if` statement, enclose the statements in braces (`{` and `}`). A set of statements contained within a pair of braces is called a block.
- A logic error has its effect at execution time. A fatal logic error causes a program to fail and terminate prematurely. A nonfatal logic error allows a program to continue executing, but the program produces incorrect results.

Section 7.7 while Repetition Statement

- The `while` repetition structure allows the programmer to specify that an action is to be repeated while some condition remains true.

Section 7.8 Formulating Algorithms: Counter-Controlled Repetition

- Counter-controlled repetition is often called definite repetition, because the number of repetitions is known before the loop begins executing.
- Uninitialized variables used in mathematical calculations result in logic errors and produce the value `NaN` (not a number).
- JavaScript represents all numbers as floating-point numbers in memory. Floating-point numbers often develop through division. The computer allocates only a fixed amount of space to hold such a value, so the stored floating-point value can only be an approximation.

Section 7.9 Formulating Algorithms: Sentinel-Controlled Repetition

- In sentinel-controlled repetition, a special value called a sentinel value (also called a signal value, a dummy value or a flag value) indicates the end of data entry. Sentinel-controlled repetition often is called indefinite repetition, because the number of repetitions is not known in advance.
- It is necessary to choose a sentinel value that cannot be confused with an acceptable input value.

Section 7.10 Formulating Algorithms: Nested Control Statements.

- Top-down, stepwise refinement is a technique essential to the development of well-structured algorithms. The top is a single statement that conveys the overall purpose of the program. As such, the top is, in effect, a complete representation of a program. The stepwise refinement process divides the top into a series of smaller tasks. The programmer terminates the top-down, stepwise refinement process when the pseudocode algorithm is specified in sufficient detail for the programmer to be able to convert the pseudocode to a JavaScript program.

Section 7.11 Assignment Operators

- JavaScript provides the arithmetic assignment operators `+=`, `-=`, `*=`, `/=` and `%=`, which abbreviate certain common types of expressions.

Section 7.12 Increment and Decrement Operators

- The increment operator, `++`, and the decrement operator, `--`, increment or decrement a variable by 1, respectively. If the operator is prefixed to the variable, the variable is incremented or decremented by 1, then used in its expression. If the operator is postfix to the variable, the variable is used in its expression, then incremented or decremented by 1.

Terminology

-- operator	definite repetition
? : operator	diamond symbol
++ operator	division by zero
action symbol	double-selection structure
action/decision model	dummy value
algorithm	empty statement (;)
arithmetic assignment operators:	fatal logic error
<code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> and <code>%=</code>	first refinement
arrow	flag value
block	floating-point number
body of a loop	flowchart
boolean expression	flowlines
braces ({})	goto elimination
compound assignment operator	goto statement
conditional expression	if selection statement
conditional operator (?:)	if...else selection statement
connector symbol	increment operator (++)
control structure	indefinite repetition
control-structure nesting	infinite loop
control-structure stacking	initialization
counter	initialization phase
counter-controlled repetition	keyword
dangling-else problem	left-hand-side expression
decision symbol	logic error
decrement operator (--)	loop

loop-continuation condition	sentinel value
<i>lvalue</i>	sentinel-controlled repetition
multiple-selection structure	sequence structure
NaN (not a number)	sequential execution
nested control structures	signal value
nonfatal logic error	single-entry/single-exit control structure
<code>null</code>	single-selection structure
oval symbol	small circle symbol
postdecrement operator	stacked control structures
postincrement operator	statement
predecrement operator	structured programming
preincrement operator	syntax error
procedure	terminate a loop
processing phase	termination phase
program control	ternary operator
pseudocode	top
rectangle symbol	top-down, stepwise refinement
repetition	transfer of control
repetition structure	unary operator
second refinement	<code>while</code> repetition statement
selection	white-space character
selection structure	

Self-Review Exercises

- 7.1** Fill in the blanks in each of the following statements:
- All programs can be written in terms of three types of control structures: _____, _____ and _____.
 - The _____ double-selection statement is used to execute one action when a condition is true and another action when that condition is false.
 - Repeating a set of instructions a specific number of times is called _____ repetition.
 - When it is not known in advance how many times a set of statements will be repeated, a(n) _____ (or a(n) _____, _____ or _____) value can be used to terminate the repetition.
- 7.2** Write four JavaScript statements that each add 1 to variable `x`, which contains a number.
- 7.3** Write JavaScript statements to accomplish each of the following tasks:
- Assign the sum of `x` and `y` to `z`, and increment the value of `x` by 1 after the calculation. Use only one statement.
 - Test whether the value of the variable `count` is greater than 10. If it is, print "Count is greater than 10".
 - Decrement the variable `x` by 1, then subtract it from the variable `total`. Use only one statement.
 - Calculate the remainder after `q` is divided by `divisor`, and assign the result to `q`. Write this statement in two different ways.
- 7.4** Write a JavaScript statement to accomplish each of the following tasks:
- Declare variables `sum` and `x`.
 - Assign 1 to variable `x`.
 - Assign 0 to variable `sum`.
 - Add variable `x` to variable `sum`, and assign the result to variable `sum`.
 - Print "The sum is: ", followed by the value of variable `sum`.

7.5 Combine the statements that you wrote in Exercise 7.4 into a JavaScript program that calculates and prints the sum of the integers from 1 to 10. Use the `while` statement to loop through the calculation and increment statements. The loop should terminate when the value of `x` becomes 11.

7.6 Determine the value of each variable after the calculation is performed. Assume that, when each statement begins executing, all variables have the integer value 5.

- `product *= x++;`
- `quotient /= ++x;`

7.7 Identify and correct the errors in each of the following segments of code:

- `while (c <= 5) {
 product *= c;
 ++c;
}
if (gender == 1)
 document.writeln("Woman");
else;
 document.writeln("Man");`

7.8 What is wrong with the following `while` repetition statement?

```
while ( z >= 0 )  
    sum += z;
```

Answers to Self-Review Exercises

7.1 a) Sequence, selection and repetition. b) `if...else`. c) Counter-controlled (or definite). d) Sentinel, signal, flag or dummy.

7.2 `x = x + 1;
x += 1;
++x;
x++;`

7.3 a) `z = x++ + y;`
b) `if (count > 10)
 document.writeln("Count is greater than 10");`
c) `total -= --x;`
d) `q %= divisor;
q = q % divisor;`

7.4 a) `var sum, x;`
b) `x = 1;`
c) `sum = 0;`
d) `sum += x; or sum = sum + x;`
e) `document.writeln("The sum is: " + sum);`

7.5 The solution is as follows:

```
1  <?xml version = "1.0" encoding = "utf-8"?>  
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
4  
5  <!-- Exercise 7.5: ex07_05.html -->  
6  <html xmlns = "http://www.w3.org/1999/xhtml">  
7      <head><title>Sum the Integers from 1 to 10</title>  
8      <script type = "text/javascript">  
9          <!--
```

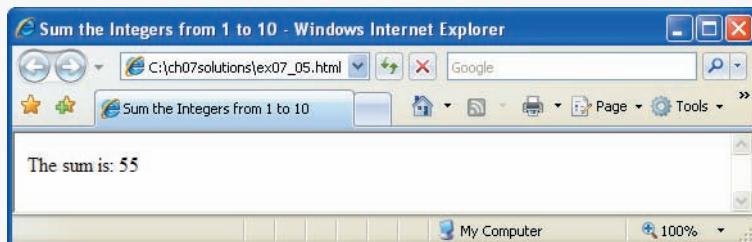
```

10     var sum; // stores the total
11     var x; // counter control variable
12
13     x = 1;
14     sum = 0;
15
16     while ( x <= 10 )
17     {
18         sum += x;
19         ++x;
20     } // end while
21
22     document.writeln( "The sum is: " + sum );
23     // -->
24 
```

</script>

</head><body></body>

</html>



- 7.6**
- a) `product = 25, x = 6;`
 - b) `quotient = 0.833333..., x = 6;`
- 7.7**
- a) Error: Missing the closing right brace of the `while` body.
Correction: Add closing right brace after the statement `++c;`.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 7.7a: ex07_07a.html -->
6  <html xmlns = "http://www.w3.org/1999/xhtml">
7      <head>
8          <title>Finding Code Errors</title>
9          <script type = "text/javascript">
10             <!--
11             var c;
12             var product;
13
14             c = 1;
15             product = 1;
16
17             while ( c <= 5 )
18             {
19                 product *= c;
20                 ++c;
21             } // end while
22
23             document.writeln( "The product is: " + product );

```

```

24      // -->
25      </script>
26      </head><body></body>
27  </html>

```



- b) Error: The semicolon after `else` results in a logic error. The second output statement will always be executed.

Correction: Remove the semicolon after `else`.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 7.7b: ex07_07b.html -->
6  <html xmlns = "http://www.w3.org/1999/xhtml">
7      <head>
8          <title>Finding Code Errors</title>
9          <script type = "text/javascript">
10             <!--
11             var gender;
12             gender = window.prompt( "Enter gender"
13                 + "(1=Woman,2=Man)", "1" );
14
15             if ( gender == 1 )
16                 document.writeln( "Woman" );
17             else
18                 document.writeln( "Man" );
19             // -->
20         </script>
21     </head><body></body>
22  </html>

```



7.8 The value of the variable *z* is never changed in the body of the `while` statement. Therefore, if the loop-continuation condition (`z >= 0`) is true, an infinite loop is created. To prevent the creation of the infinite loop, *z* must be decremented so that it eventually becomes less than 0.

Exercises

7.9 Identify and correct the errors in each of the following segments of code [Note: There may be more than one error in each piece of code]:

- `if (age >= 65);
 document.writeln("Age greater than or equal to 65");
else
 document.writeln("Age is less than 65");`
- `var x = 1, total;
while (x <= 10)
{
 total += x;
 ++x;
}
var x = 1;
var total = 0;
While (x <= 100)
 total += x;
 ++x;`
- `var y = 5;
while (y > 0)
{
 document.writeln(y);
 ++y;`

7.10 What does the following program print?

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 7.10: ex07_10.html -->
6  <html xmlns="http://www.w3.org/1999/xhtml">
7      <head><title>Mystery Script</title>
8          <script type = "text/javascript">
9              <!--
10                 var y;
11                 var x = 1;
12                 var total = 0;
13
14                 while ( x <= 10 )
15                 {
16                     y = x * x;
17                     document.writeln( y + "<br />" );
18                     total += y;
19                     ++x;
20                 } // end while
21
22                 document.writeln( "<br />Total is " + total );
23             // -->

```

```

24      </script>
25      </head><body></body>
26  </html>

```

For Exercises 7.11–7.14, perform each of the following steps:

- Read the problem statement.
- Formulate the algorithm using pseudocode and top-down, stepwise refinement.
- Write a JavaScript program.
- Test, debug and execute the JavaScript program.
- Process three complete sets of data.

7.11 Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several tankfuls of gasoline by recording the number of miles driven and the number of gallons used for each tankful. Develop a JavaScript program that will take as input the miles driven and gallons used (both as integers) for each tankful. The program should calculate and output XHTML text that displays the number of miles per gallon obtained for each tankful and prints the combined number of miles per gallon obtained for all tankfuls up to this point. Use `prompt` dialogs to obtain the data from the user.

7.12 Develop a JavaScript program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following facts are available:

- Account number
- Balance at the beginning of the month
- Total of all items charged by this customer this month
- Total of all credits applied to this customer's account this month
- Allowed credit limit

The program should input each of these facts from a `prompt` dialog as an integer, calculate the new balance ($= \text{beginning balance} + \text{charges} - \text{credits}$), display the new balance and determine whether the new balance exceeds the customer's credit limit. For customers whose credit limit is exceeded, the program should output XHTML text that displays the message "Credit limit exceeded."

7.13 A large company pays its salespeople on a commission basis. The salespeople receive \$200 per week, plus 9 percent of their gross sales for that week. For example, a salesperson who sells \$5000 worth of merchandise in a week receives \$200 plus 9 percent of \$5000, or a total of \$650. You have been supplied with a list of the items sold by each salesperson. The values of these items are as follows:

Item	Value
1	239.99
2	129.75
3	99.95
4	350.89

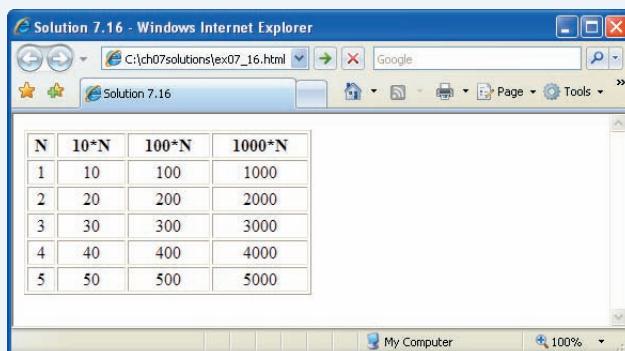
Develop a program that inputs one salesperson's items sold for last week, calculates the salesperson's earnings and outputs XHTML text that displays the salesperson's earnings.

7.14 Develop a JavaScript program that will determine the gross pay for each of three employees. The company pays "straight time" for the first 40 hours worked by each employee and pays "time and a half" for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee, determine the employee's gross pay and output XHTML text that displays the employee's gross pay. Use `prompt` dialogs to input the data.

7.15 The process of finding the largest value (i.e., the maximum of a group of values) is used frequently in computer applications. For example, a program that determines the winner of a sales contest would input the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a pseudocode program and then a JavaScript program that inputs a series of 10 single-digit numbers as characters, determines the largest of the numbers and outputs a message that displays the largest number. Your program should use three variables as follows:

- counter: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed);
- number: The current digit input to the program;
- largest: The largest number found so far.

7.16 Write a JavaScript program that uses looping to print the following table of values. Output the results in an XHTML table. Use CSS to center the data in each column.



The screenshot shows a Microsoft Internet Explorer window titled "Solution 7.16 - Windows Internet Explorer". The address bar shows the URL "C:\ch07\solutions\ex07_16.html". The page content displays a table with four columns: N, 10*N, 100*N, and 1000*N. The table has 6 rows, including the header row. The data is as follows:

N	10*N	100*N	1000*N
1	10	100	1000
2	20	200	2000
3	30	300	3000
4	40	400	4000
5	50	500	5000

7.17 Using an approach similar to that in Exercise 7.15, find the *two* largest values among the 10 digits entered. [Note: You may input each number only once.]

7.18 Modify the program in Fig. 7.11 to validate its inputs. For every value input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

7.19 What does the following program print?

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 7.19: ex07_19.html -->
6  <html xmlns = "http://www.w3.org/1999/xhtml">
7      <head><title>Mystery Script</title>
8          <script type = "text/javascript">
9              <!--
10                  var count = 1;
11
12                  while ( count <= 10 )
13                  {
14                      document.writeln(
15                          count % 2 == 1 ? "****<br />" : "++++++<br />");
16                      ++count;
17                  } // end while
18                  // --
19          </script>
20      </head><body></body>
21  </html>

```

7.20 What does the following program print?

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 7.20: ex07_20.html -->
6  <html xmlns = "http://www.w3.org/1999/xhtml">
7      <head><title>Mystery Script</title>
8          <script type = "text/javascript">
9              <!--
10                  var row = 10;
11                  var column;
12
13                  while ( row >= 1 )
14                  {
15                      column = 1;
16
17                      while ( column <= 10 )
18                      {
19                          document.write( row % 2 == 1 ? "<" : ">" );
20                          ++column;
21                      } // end while
22
23                      --row;
24                      document.writeln( "<br />" );
25                  } // end while
26                  // -->
27          </script>
28      </head><body></body>
29  </html>

```

7.21 (*Dangling-Else Problem*) Determine the output for each of the given segments of code when *x* is 9 and *y* is 11, and when *x* is 11 and *y* is 9. Note that the interpreter ignores the indentation in a JavaScript program. Also, the JavaScript interpreter always associates an *else* with the previous *if*, unless told to do otherwise by the placement of braces ({}). You may not be sure at first glance which *if* an *else* matches. This situation is referred to as the “dangling-else” problem. We have eliminated the indentation from the given code to make the problem more challenging. [Hint: Apply the indentation conventions you have learned.]

- a)

```

if ( x < 10 )
    if ( y > 10 )
        document.writeln( "*****<br />" );
    else
        document.writeln( "#####<br />" );
        document.writeln( "$$$$$<br />" );

```
- b)

```

if ( x < 10 )
{
    if ( y > 10 )
        document.writeln( "*****<br />" );
}
else
{
    document.writeln( "#####<br />" );
    document.writeln( "$$$$$<br />" );
}

```

7.22 (*Another Dangling-Else Problem*) Modify the given code to produce the output shown in each part of this problem. Use proper indentation techniques. You may not make any changes other than inserting braces, changing the code indentation and inserting blank lines. The interpreter ignores indentation in JavaScript. We have eliminated the indentation from the given code to make the problem more challenging. [Note: It is possible that no modification is necessary for some of the segments of code.]

```
if ( y == 8 )
if ( x == 5 )
document.writeln( "AAAAA<br />" );
else
document.writeln( "####<br />" );
document.writeln( "$$$$<br />" );
document.writeln( "&&&&&<br />" );
```

- a) Assuming that $x = 5$ and $y = 8$, the following output is produced:

```
AAAAA
$$$$
&&&&
```

- b) Assuming that $x = 5$ and $y = 8$, the following output is produced:

```
AAAAA
```

- c) Assuming that $x = 5$ and $y = 8$, the following output is produced:

```
AAAAA
&&&&
```

- d) Assuming that $x = 5$ and $y = 7$, the following output is produced [Note: The last three output statements after the `else` statements are all part of a block]:

```
#####
$$$$
&&&&
```

7.23 Write a script that reads in the size of the side of a square and outputs XHTML text that displays a hollow square of that size constructed of asterisks. Use a `prompt` dialog to read the size from the user. Your program should work for squares of all side sizes between 1 and 20.

7.24 A palindrome is a number or a text phrase that reads the same backward and forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write a script that reads in a five-digit integer and determines whether it is a palindrome. If the number is not five digits long, display an `alert` dialog indicating the problem to the user. Allow the user to enter a new value after dismissing the `alert` dialog. [Hint: It is possible to do this exercise with the techniques learned in this chapter. You will need to use both division and remainder operations to “pick off” each digit.]

7.25 Write a script that outputs XHTML text that displays the following checkerboard pattern:

```
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
```

Your program may use only three output statements to display the pattern, one of the form

```
document.write( "* " );
```

one of the form

```
document.write( " " );
```

and one of the form

```
document.writeln(); //writes a newline character
```

You may use XHTML tags (e.g., `<pre>`) for alignment purposes. [Hint: Repetition structures are required in this exercise.]

7.26 Write a script that outputs XHTML text that keeps displaying in the browser window the multiples of the integer 2, namely 2, 4, 8, 16, 32, 64, etc. Your loop should *not terminate* (i.e., you should create an infinite loop). What happens when you run this program?

7.27 A company wants to transmit data over the telephone, but it is concerned that its phones may be tapped. All of its data is transmitted as four-digit integers. It has asked you to write a program that will encrypt its data so that the data may be transmitted more securely. Your script should read a four-digit integer entered by the user in a prompt dialog and encrypt it as follows: Replace each digit by (*the sum of that digit plus 7*) *modulus 10*. Then swap the first digit with the third, and swap the second digit with the fourth. Then output XHTML text that displays the encrypted integer.

7.28 Write a program that inputs an encrypted four-digit integer (from Exercise 7.27) and decrypts it to form the original number.

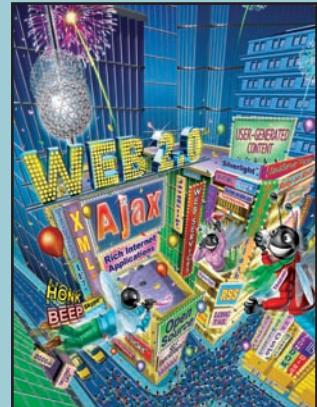
8

JavaScript: Control Statements II

OBJECTIVES

In this chapter you will learn:

- The essentials of counter-controlled repetition
- To use the `for` and `do...while` repetition statements to execute statements in a program repeatedly.
- To perform multiple selection using the `switch` selection statement.
- To use the `break` and `continue` program-control statements
- To use the logical operators.



Not everything that can be counted counts, and not every thing that counts can be counted.

—Albert Einstein

Who can control his fate?

—William Shakespeare

The used key is always bright.

—Benjamin Franklin

Intelligence ... is the faculty of making artificial objects, especially tools to make tools.

—Henri Bergson

Every advantage in the past is judged in the light of the final issue.

—Demosthenes

Outline

- 8.1** Introduction
- 8.2** Essentials of Counter-Controlled Repetition
- 8.3** `for` Repetition Statement
- 8.4** Examples Using the `for` Statement
- 8.5** `switch` Multiple-Selection Statement
- 8.6** `do...while` Repetition Statement
- 8.7** `break` and `continue` Statements
- 8.8** Labeled `break` and `continue` Statements
- 8.9** Logical Operators
- 8.10** Summary of Structured Programming
- 8.11** Wrap-Up
- 8.12** Web Resources

[Summary](#) | [Terminology](#) | [Self-Review Exercises](#) | [Answers to Self-Review Exercises](#) | [Exercises](#)

8.1 Introduction

Chapter 7 began our introduction to the types of building blocks that are available for problem solving and used them to employ proven program-construction principles. In this chapter, we continue our presentation of the theory and principles of structured programming by introducing JavaScript’s remaining control statements (with the exception of `for...in`, which is presented in Chapter 10). As in Chapter 7, the JavaScript techniques you will learn here are applicable to most high-level languages. In later chapters, you’ll see that control structures are helpful in manipulating objects.

8.2 Essentials of Counter-Controlled Repetition

Counter-controlled repetition requires:

1. The *name* of a control variable (or loop counter).
2. The *initial value* of the control variable.
3. The *increment* (or *decrement*) by which the control variable is modified each time through the loop (also known as *each iteration of the loop*).
4. The condition that tests for the *final value* of the control variable to determine whether looping should continue.

To see the four elements of counter-controlled repetition, consider the simple script shown in Fig. 8.1, which displays lines of XHTML text that illustrate the seven different font sizes supported by XHTML. The declaration in line 12 *names* the control variable (`counter`), reserves space for it in memory and sets it to an *initial value* of 1. The declaration and initialization of `counter` could also have been accomplished by the following declaration and assignment statement:

```
var counter; // declare counter
counter = 1; // initialize counter to 1
```

Lines 16–18 in the `while` statement write a paragraph element consisting of the string “XHTML font size” concatenated with the control variable `counter`’s value, which repre-

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.1: WhileCounter.html -->
6 <!-- Counter-controlled repetition. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Counter-Controlled Repetition</title>
10    <script type = "text/javascript">
11      <!--
12        var counter = 1; // initialization
13
14        while ( counter <= 7 ) // repetition condition
15        {
16          document.writeln( "<p style = \"font-size: " +
17            counter + "ex\">XHTML font size " + counter +
18            "ex</p>" );
19          ++counter; // increment
20        } //end while
21        // --
22      </script>
23    </head><body></body>
24 </html>

```

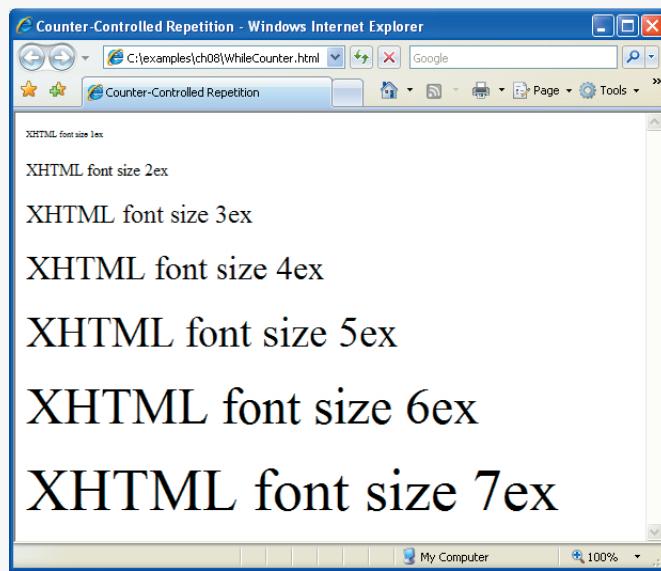


Fig. 8.1 | Counter-controlled repetition.

sents the font size. An inline CSS `style` attribute sets the `font-size` property to the value of `counter` concatenated to `ex`. Note the use of the escape sequence `\"`, which is placed around attribute `style`'s value. Because the double-quote character delimits the beginning and end of a string literal in JavaScript, it cannot be used in the contents of the string

unless it is preceded by a \ to create the escape sequence \". For example, if counter is 5, the preceding statement produces the markup

```
<p style = "font-size: 5ex">XHTML font size 5ex</p>
```

XHTML allows either single quotes (') or double quotes (") to be placed around the value specified for an attribute. JavaScript allows single quotes to be placed in a string literal. Thus, we could have placed single quotes around the font-size property to produce equivalent XHTML output without the use of escape sequences.



Common Programming Error 8.1

Placing a double-quote (") character inside a string literal that is delimited by double quotes causes a runtime error when the script is interpreted. To be displayed as part of a string literal, a double-quote (") character must be preceded by a \ to form the escape sequence \".

Line 19 in the while statement increments the control variable by 1 for each iteration of the loop (i.e., each time the body of the loop is performed). The loop-continuation condition (line 14) in the while statement tests whether the value of the control variable is less than or equal to 7 (the *final value* for which the condition is true). Note that the body of this while statement executes even when the control variable is 7. The loop terminates when the control variable exceeds 7 (i.e., counter becomes 8).



Good Programming Practice 8.1

Use integer values to control loop counting.



Good Programming Practice 8.2

Indent the statements in the body of each control structure.



Good Programming Practice 8.3

Put a blank line before and after each control structure, to make it stand out in the program.



Good Programming Practice 8.4

Too many levels of nesting can make a program difficult to understand. As a general rule, try to avoid using more than three levels of nesting.



Good Programming Practice 8.5

Vertical spacing above and below control structures and indentation of the bodies of control structures in the headers of the control structure give programs a two-dimensional appearance that enhances readability.

8.3 for Repetition Statement

The **for repetition statement** handles all the details of counter-controlled repetition. Figure 8.2 illustrates the power of the for statement by reimplementing the script of Fig. 8.1.

When the for statement begins executing (line 15), the control variable counter is declared and is initialized to 1 (i.e., the first statement of the for statement declares the control variable's *name* and provides the control variable's *initial value*). Next, the loop-

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.2: ForCounter.html -->
6 <!-- Counter-controlled repetition with the for statement. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Counter-Controlled Repetition</title>
10    <script type = "text/javascript">
11      <!--
12        // Initialization, repetition condition and
13        // incrementing are all included in the for
14        // statement header.
15        for ( var counter = 1; counter <= 7; ++counter )
16          document.writeln( "<p style = \"font-size: " +
17            counter + "ex\">XHTML font size " + counter +
18            "ex</p>" );
19        // -->
20      </script>
21    </head><body></body>
22 </html>

```

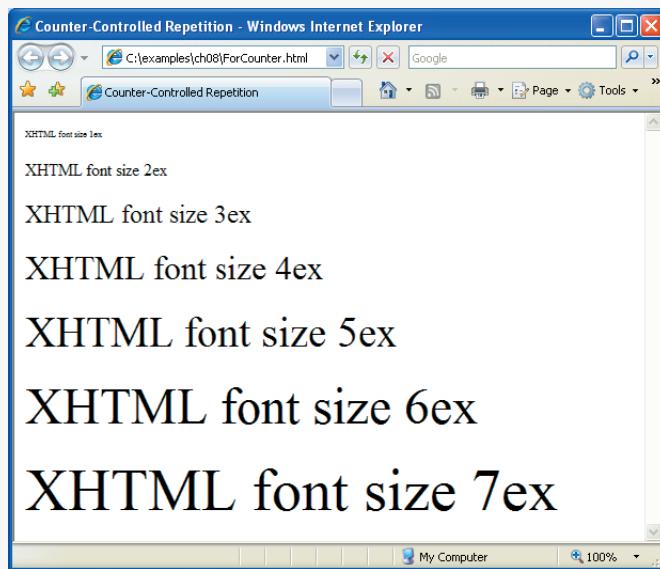


Fig. 8.2 | Counter-controlled repetition with the for statement.

continuation condition, `counter <= 7`, is checked. The condition contains the *final value* (7) of the control variable. The initial value of `counter` is 1. Therefore, the condition is satisfied (i.e., true), so the body statement (lines 16–18) writes a paragraph element in the XHTML document. Then, variable `counter` is incremented in the expression `++counter` and the loop continues execution with the loop-continuation test. The control variable is now equal to 2, so the final value is not exceeded and the program performs the body statement again (i.e., performs the next iteration of the loop). This process continues until the

control variable counter becomes 8, at which point the loop-continuation test fails and the repetition terminates.

The program continues by performing the first statement after the `for` statement. (In this case, the script terminates, because the interpreter reaches the end of the script.)

Figure 8.3 takes a closer look at the `for` statement at line 15 of Fig. 8.2. The `for` statement's first line (including the keyword `for` and everything in parentheses after `for`) is often called the **for statement header**. Note that the `for` statement "does it all"—it specifies each of the items needed for counter-controlled repetition with a control variable. Remember that a block is a group of statements enclosed in curly braces that can be placed anywhere that a single statement can be placed, so you can use a block to put multiple statements into the body of a `for` statement.

Note that Fig. 8.3 uses the loop-continuation condition `counter <= 7`. If you incorrectly write `counter < 7`, the loop will execute only six times. This is an example of the common logic error called an **off-by-one error**.



Common Programming Error 8.2

Using an incorrect relational operator or an incorrect final value of a loop counter in the condition of a while, for or do...while statement can cause an off-by-one error or an infinite loop.



Good Programming Practice 8.6

*Using the final value in the condition of a while or for statement and using the `<=` relational operator will help avoid off-by-one errors. For a loop used to print the values 1 to 10, for example, the initial value of counter should be 1, and the loop-continuation condition should be `counter <= 10` rather than `counter < 10` (which is an off-by-one error) or `counter < 11` (which is correct). Many programmers, however, prefer so-called **zero-based counting**, in which, to count 10 times through the loop, counter would be initialized to zero and the loop-continuation test would be `counter < 10`.*

The general format of the `for` statement is

```
for ( initialization; loopContinuationTest; increment )
    statements
```

where the *initialization* expression names the loop's control variable and provides its initial value, *loopContinuationTest* is the expression that tests the loop-continuation condition (containing the final value of the control variable for which the condition is true), and

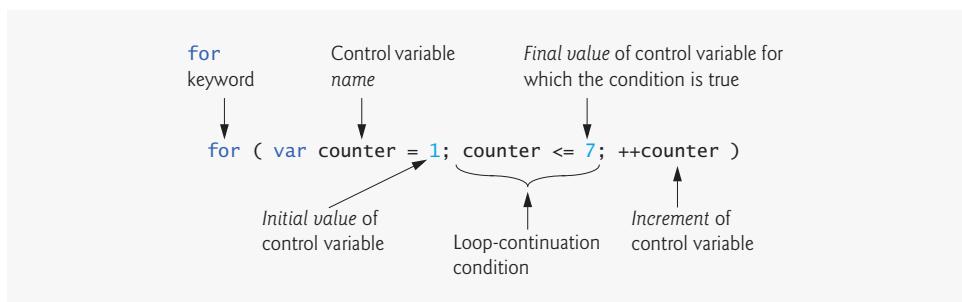


Fig. 8.3 | `for` statement header components.

increment is an expression that increments the control variable. The `for` statement can be represented by an equivalent `while` statement, with *initialization*, *loopContinuationTest* and *increment* placed as follows:

```
initialization;
while ( loopContinuationTest )
{
    statements
    increment;
}
```

In Section 8.7 we discuss an exception to this rule.

If the *initialization* expression in the `for` statement's header is the first definition of the control variable, the control variable can still be used after the `for` statement in the script. The part of a script in which a variable name can be used is known as the variable's **scope**. Scope is discussed in detail in Chapter 9, JavaScript: Functions.



Good Programming Practice 8.7

Place only expressions involving the control variable in the initialization and increment sections of a for statement. Manipulations of other variables should appear either before the loop (if they execute only once, like initialization statements) or in the loop body (if they execute once per iteration of the loop, like incrementing or decrementing statements).

The three expressions in the `for` statement are optional. If *loopContinuationTest* is omitted, JavaScript assumes that the loop-continuation condition is `true`, thus creating an infinite loop. One might omit the *initialization* expression if the control variable is initialized before the loop. One might omit the *increment* expression if the increment is calculated by statements in the body of the `for` statement or if no increment is needed. The *increment* expression in the `for` statement acts like a stand-alone statement at the end of the body of the `for` statement. Therefore, the expressions

```
counter = counter + 1
counter += 1
++counter
counter++
```

are all equivalent in the incrementing portion of the `for` statement. Many programmers prefer the form `counter++`. This is because the incrementing of the control variable occurs after the body of the loop is executed, and therefore the postincrementing form seems more natural. Preincrementing and postincrementing both have the same effect in our example, because the variable being incremented does not appear in a larger expression. The two semicolons in the `for` statement header are required.



Common Programming Error 8.3

Using commas instead of the two required semicolons in the header of a for statement is a syntax error.



Common Programming Error 8.4

Placing a semicolon immediately to the right of the right parenthesis of the header of a for statement makes the body of that for statement an empty statement. This code is normally a logic error.

The initialization, loop-continuation condition and increment portions of a `for` statement can contain arithmetic expressions. For example, assume that $x = 2$ and $y = 10$. If x and y are not modified in the body of the loop, then the statement

```
for ( var j = x; j <= 4 * x * y; j += y / x )
```

is equivalent to the statement

```
for ( var j = 2; j <= 80; j += 5 )
```

The “increment” of a `for` statement may be negative, in which case it is really a decrement and the loop actually counts downward.

If the loop-continuation condition initially is `false`, the `for` statement’s body is not performed. Instead, execution proceeds with the statement following the `for` statement.

The control variable frequently is printed or used in calculations in the body of a `for` statement, but it does not have to be. Other times, the control variable is used for controlling repetition but never mentioned in the body of the `for` statement.



Error-Prevention Tip 8.1

Although the value of the control variable can be changed in the body of a `for` statement, avoid changing it, because doing so can lead to subtle errors.

The `for` statement is flowcharted much like the `while` statement. For example, Fig. 8.4 shows the flowchart of the `for` statement

```
for ( var counter = 1; counter <= 7; ++counter )
    document.writeln( "<p style = \"font-size: " +
        counter + "ex\">XHTML font size " + counter +
        "</p>" );
```

This flowchart makes it clear that the initialization occurs only once and that incrementing occurs *after* each execution of the body statement. Note that, besides small circles and arrows, the flowchart contains only rectangle symbols and a diamond symbol.

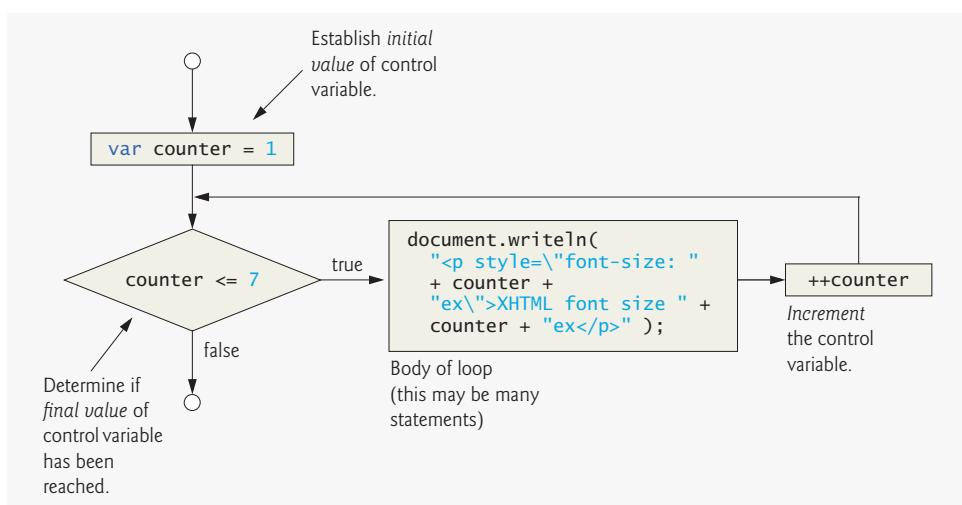


Fig. 8.4 | `for` repetition statement flowchart.

8.4 Examples Using the for Statement

The examples in this section show methods of varying the control variable in a `for` statement. In each case, we write the appropriate `for` header. Note the change in the relational operator for loops that decrement the control variable.

- a) Vary the control variable from 1 to 100 in increments of 1.

```
for ( var i = 1; i <= 100; ++i )
```

- b) Vary the control variable from 100 to 1 in increments of -1 (i.e., decrements of 1).

```
for ( var i = 100; i >= 1; --i )
```

- c) Vary the control variable from 7 to 77 in steps of 7.

```
for ( var i = 7; i <= 77; i += 7 )
```

- d) Vary the control variable from 20 to 2 in steps of -2.

```
for ( var i = 20; i >= 2; i -= 2 )
```

- e) Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17, 20.

```
for ( var j = 2; j <= 20; j += 3 )
```

- f) Vary the control variable over the following sequence of values: 99, 88, 77, 66, 55, 44, 33, 22, 11, 0.

```
for ( var j = 99; j >= 0; j -= 11 )
```



Common Programming Error 8.5

Not using the proper relational operator in the loop-continuation condition of a loop that counts downward (e.g., using `i <= 1` in a loop that counts down to 1) is usually a logic error that will yield incorrect results when the program runs.

The next two scripts demonstrate the `for` repetition statement. Figure 8.5 uses the `for` statement to sum the even integers from 2 to 100. Note that the increment expression adds 2 to the control variable `number` after the body executes during each iteration of the loop. The loop terminates when `number` has the value 102 (which is not added to the sum).

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.5: Sum.html -->
6  <!-- Summation with the for repetition structure. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8    <head>
9      <title>Sum the Even Integers from 2 to 100</title>
10     <script type = "text/javascript">
11       <!--
12         var sum = 0;
```

Fig. 8.5 | Summation with the `for` repetition structure. (Part 1 of 2.)

```

13
14     for ( var number = 2; number <= 100; number += 2 )
15         sum += number;
16
17     document.writeln( "The sum of the even integers " +
18         "from 2 to 100 is " + sum );
19     // -->
20     </script>
21 </head><body></body>
22 </html>

```

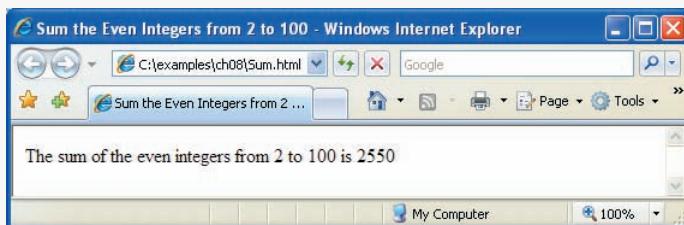


Fig. 8.5 | Summation with the for repetition structure. (Part 2 of 2.)

Note that the body of the for statement in Fig. 8.5 actually could be merged into the rightmost (increment) portion of the for header by using a comma, as follows:

```
for ( var number = 2; number <= 100; sum += number, number += 2 );
```

Similarly, the initialization `sum = 0` could be merged into the initialization section of the for statement.



Good Programming Practice 8.8

Although statements preceding a for statement and in the body of a for statement can often be merged into the for header, avoid doing so, because it makes the program more difficult to read.



Good Programming Practice 8.9

For clarity, limit the size of control-statement headers to a single line, if possible.

The next example computes compound interest (compounded yearly) using the for statement. Consider the following problem statement:

A person invests \$1000.00 in a savings account yielding 5 percent interest. Assuming that all the interest is left on deposit, calculate and print the amount of money in the account at the end of each year for 10 years. Use the following formula to determine the amounts:

$$a = p (1 + r)^n$$

where

p is the original amount invested (i.e., the principal)

r is the annual interest rate

n is the number of years

a is the amount on deposit at the end of the *n*th year.

This problem involves a loop that performs the indicated calculation for each of the 10 years the money remains on deposit. Figure 8.6 presents the solution to this problem, displaying the results in a table.

Lines 16–18 declare three variables and initialize `principal` to 1000.0 and `rate` to .05. Lines 20–21 write an XHTML `<table>` tag, and lines 22–23 write the caption that summarizes the table's content. Lines 24–25 create the table's header section (`<thead>`), a

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.6: Interest.html -->
6 <!-- Compound interest calculation with a for loop. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Calculating Compound Interest</title>
10    <style type = "text/css">
11      table { width: 100% }
12      th { text-align: left }
13    </style>
14    <script type = "text/javascript">
15      <!--
16      var amount; // current amount of money
17      var principal = 1000.0; // principal amount
18      var rate = .05; // interest rate
19
20      document.writeln(
21        "<table border = \"1\">"); // begin the table
22      document.writeln(
23        "<caption>Calculating Compound Interest</caption>");
24      document.writeln(
25        "<thead><tr><th>Year</th>"); // year column heading
26      document.writeln(
27        "<th>Amount on deposit</th>"); // amount column heading
28      document.writeln(")</tr></thead><tbody>";
29
30      // output a table row for each year
31      for ( var year = 1; year <= 10; ++year )
32      {
33        amount = principal * Math.pow( 1.0 + rate, year );
34        document.writeln( "<tr><td>" + year +
35          "</td><td>" + amount.toFixed(2) +
36          "</td></tr>" );
37      } //end for
38
39      document.writeln( "</tbody></table>" );
40      // --
41    </script>
42  </head><body></body>
43 </html>

```

Fig. 8.6 | Compound interest calculation with a for loop. (Part I of 2.)

The screenshot shows a Microsoft Internet Explorer window with the title "Calculating Compound Interest - Windows Internet Explorer". The address bar displays "C:\examples\ch08\Interest.html". The main content area contains a table titled "Calculating Compound Interest". The table has two columns: "Year" and "Amount on deposit". The data rows show the following values:

Year	Amount on deposit
1	1050.00
2	1102.50
3	1157.63
4	1215.51
5	1276.28
6	1340.10
7	1407.10
8	1477.46
9	1551.33
10	1628.89

Fig. 8.6 | Compound interest calculation with a `for` loop. (Part 2 of 2.)

row (`<tr>`) and a column heading (`<th>`) containing “Year.” Lines 26–28 create a table heading for “Amount on deposit” and write the closing `</tr>` and `</thead>` tags.

The `for` statement (lines 31–37) executes its body 10 times, incrementing control variable `year` from 1 to 10 (note that `year` represents n in the problem statement). JavaScript does not include an exponentiation operator. Instead, we use the `Math` object’s `pow` method for this purpose. `Math.pow(x, y)` calculates the value of `x` raised to the `y`th power. Method `Math.pow` takes two numbers as arguments and returns the result.

Line 33 performs the calculation using the formula given in the problem statement. Lines 34–36 write a line of XHTML markup that creates another row in the table. The first column is the current `year` value. The second column displays the value of `amount`. Line 39 writes the closing `</tbody>` and `</table>` tags after the loop terminates.

Line 35 introduces the **Number object** and its **toFixed method**. The variable `amount` contains a numerical value, so JavaScript represents it as a `Number` object. The `toFixed` method of a `Number` object formats the value by rounding it to the specified number of decimal places. On line 35, `amount.toFixed(2)` outputs the value of `amount` with two decimal places.

Variables `amount`, `principal` and `rate` represent numbers in this script. Remember that JavaScript represents all numbers as floating-point numbers. This feature is convenient in this example, because we are dealing with fractional parts of dollars and need a type that allows decimal points in its values.

Unfortunately, floating-point numbers can cause trouble. Here is a simple example of what can go wrong when using floating-point numbers to represent dollar amounts (assuming that dollar amounts are displayed with two digits to the right of the decimal point): Two dollar amounts stored in the machine could be 14.234 (which would normally be rounded to 14.23 for display purposes) and 18.673 (which would normally be rounded to 18.67 for display purposes). When these amounts are added, they produce the

internal sum 32.907, which would normally be rounded to 32.91 for display purposes. Thus your printout could appear as

$$\begin{array}{r} 14.23 \\ + 18.67 \\ \hline 32.91 \end{array}$$

but a person adding the individual numbers as printed would expect the sum to be 32.90. You have been warned!

8.5 switch Multiple-Selection Statement

Previously, we discussed the `if` single-selection statement and the `if...else` double-selection statement. Occasionally, an algorithm will contain a series of decisions in which a variable or expression is tested separately for each of the values it may assume, and different actions are taken for each value. JavaScript provides the `switch` multiple-selection statement to handle such decision making. The script in Fig. 8.7 demonstrates three different CSS list formats determined by the value the user enters.

Line 12 in the script declares the variable `choice`. This variable stores the user's choice, which determines what type of XHTML list to display. Lines 13–14 declare variables `startTag` and `endTag`, which will store the XHTML tags that will be used to create the list element. Line 15 declares variable `validInput` and initializes it to `true`. The script uses this variable to determine whether the user made a valid choice (indicated by the value of `true`). If a choice is invalid, the script sets `validInput` to `false`. Line 16 declares variable `listType`, which will store an `h1` element indicating the list type. This heading appears before the list in the XHTML document.

Lines 18–19 prompt the user to enter a 1 to display a numbered list, a 2 to display a lettered list and a 3 to display a list with roman numerals. Lines 21–40 define a `switch statement` that assigns to the variables `startTag`, `endTag` and `listType` values based on the value input by the user in the `prompt` dialog. We create these different lists using the CSS property `list-style-type`, which allows us to set the numbering system for the list. Possible values include `decimal` (numbers—the default), `lower-roman` (lowercase Roman numerals), `upper-roman` (uppercase Roman numerals), `lower-alpha` (lowercase letters), `upper-alpha` (uppercase letters), and several others.

The `switch` statement consists of a series of `case labels` and an optional `default case`. When the flow of control reaches the `switch` statement, the script evaluates the `controlling expression` (`choice` in this example) in the parentheses following keyword `switch`. The value of this expression is compared with the value in each of the `case labels`, starting with the first `case label`. Assume that the user entered 2. Remember that the value typed

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.7: SwitchTest.html -->
6 <!-- Using the switch multiple-selection statement. -->
```

Fig. 8.7 | Using the `switch` multiple-selection statement. (Part I of 4.)

```
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9     <title>Switching between XHTML List Formats</title>
10    <script type = "text/javascript">
11        <!--
12        var choice; // user's choice
13        var startTag; // starting list item tag
14        var endTag; // ending list item tag
15        var validInput = true; // indicates if input is valid
16        var listType; // type of list as a string
17
18        choice = window.prompt( "Select a list style:\n" +
19                            "1 (numbered), 2 (lettered), 3 (roman)", "1" );
20
21        switch ( choice )
22        {
23            case "1":
24                startTag = "<ol>";
25                endTag = "</ol>";
26                listType = "<h1>Numbered List</h1>";
27                break;
28            case "2":
29                startTag = "<ol style = \"list-style-type: upper-alpha\>" ;
30                endTag = "</ol>";
31                listType = "<h1>Lettered List</h1>";
32                break;
33            case "3":
34                startTag = "<ol style = \"list-style-type: upper-roman\>" ;
35                endTag = "</ol>";
36                listType = "<h1>Roman Numbered List</h1>";
37                break;
38            default:
39                validInput = false;
40        } //end switch
41
42        if ( validInput == true )
43        {
44            document.writeln( listType + startTag );
45
46            for ( var i = 1; i <= 3; ++i )
47                document.writeln( "<li>List item " + i + "</li>" );
48
49            document.writeln( endTag );
50        } //end if
51        else
52            document.writeln( "Invalid choice: " + choice );
53        // -->
54    </script>
55 </head>
56 <body>
57     <p>Click Refresh (or Reload) to run the script again</p>
58 </body>
59 </html>
```

Fig. 8.7 | Using the `switch` multiple-selection statement. (Part 2 of 4.)

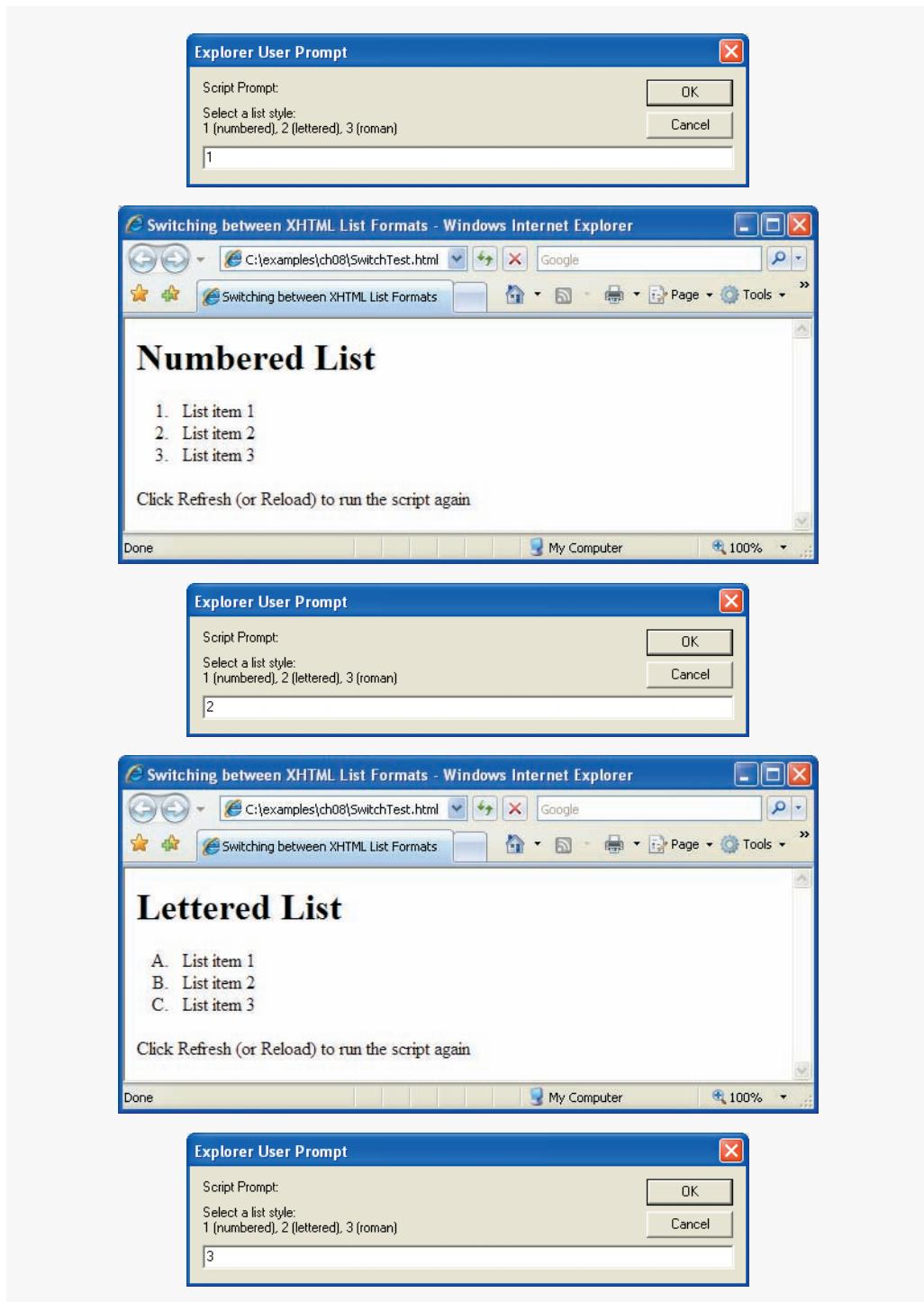


Fig. 8.7 | Using the `switch` multiple-selection statement. (Part 3 of 4.)

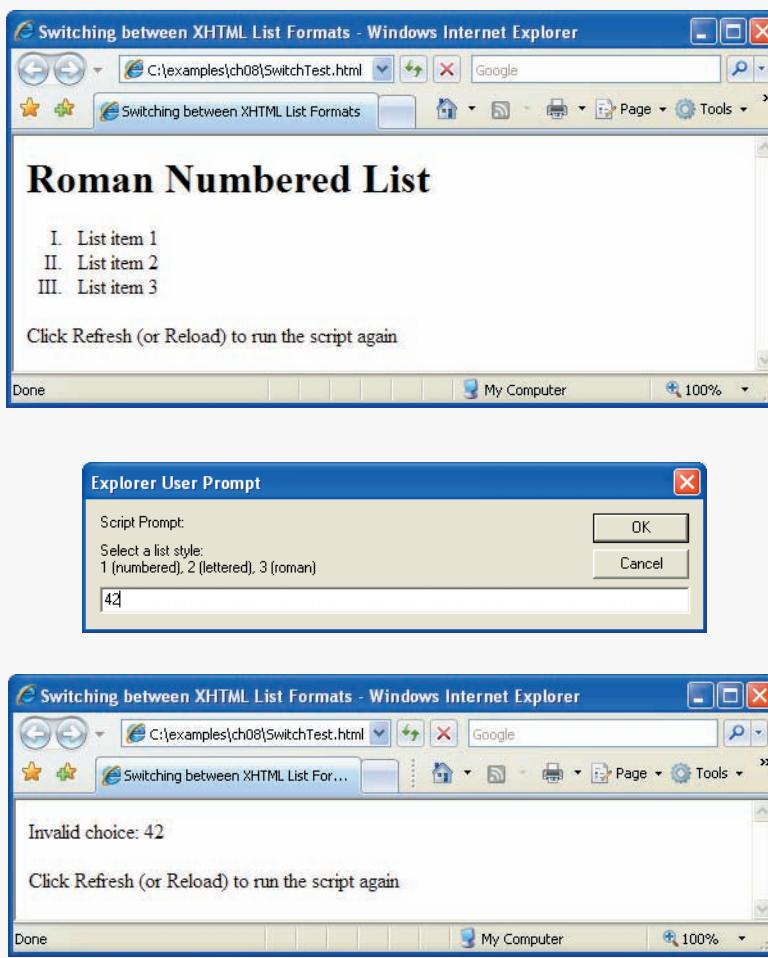


Fig. 8.7 | Using the `switch` multiple-selection statement. (Part 4 of 4.)

by the user in a prompt dialog is returned as a string. So, the string 2 is compared to the string in each case in the `switch` statement. If a match occurs (case "2":), the statements for that case execute. For the string 2 (lines 29–32), we set `startTag` to an opening `ol` tag with the style property `list-style-type` set to `upper-alpha`, set `endTag` to "``" to indicate the end of an ordered list and set `listType` to "`<h1>Lettered List</h1>`". If no match occurs between the controlling expression's value and a case label, the `default` case executes and sets variable `validInput` to `false`.

The `break` statement in line 32 causes program control to proceed with the first statement after the `switch` statement. The `break` statement is used because the cases in a `switch` statement would otherwise run together. If `break` is not used anywhere in a `switch` statement, then each time a match occurs in the statement, the statements for all the remaining cases execute.

Next, the flow of control continues with the `if` statement in line 42, which tests variable `validInput` to determine whether its value is true. If so, lines 44–49 write the `listType`, the `startTag`, three list items (``) and the `endTag`. Otherwise, the script writes text in the XHTML document indicating that an invalid choice was made (line 52).

Each case can have multiple actions (statements). The `switch` statement is different from others in that braces are not required around multiple actions in a case of a `switch`. The general `switch` statement (i.e., using a `break` in each case) is flowcharted in Fig. 8.8. [Note: As an exercise, flowchart the general `switch` statement without `break` statements.]

The flowchart makes it clear that each `break` statement at the end of a case causes control to exit from the `switch` statement immediately. The `break` statement is not required for the last case in the `switch` statement (or the `default` case, when it appears last), because program control automatically continues with the next statement after the `switch` statement.



Common Programming Error 8.6

Forgetting a `break` statement when one is needed in a `switch` statement is a logic error.



Software Engineering Observation 8.1

Provide a default case in `switch` statements. Cases not explicitly tested in a `switch` statement without a default case are ignored. Including a default case focuses the programmer on processing exceptional conditions. However, there are situations in which no default processing is needed.



Good Programming Practice 8.10

Although the case clauses and the default case clause in a `switch` statement can occur in any order, it is clearer (and more common) to place the default clause last.

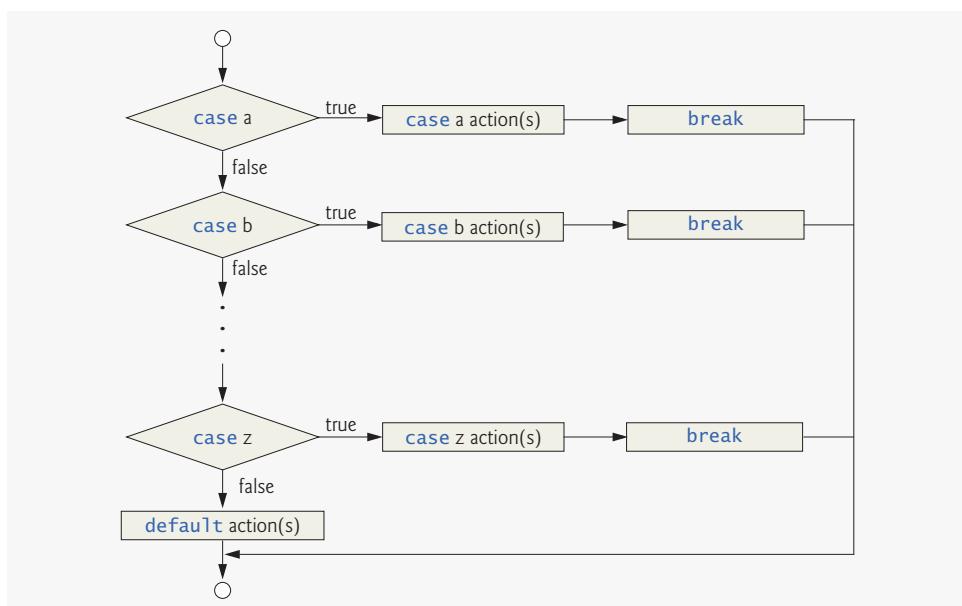


Fig. 8.8 | `switch` multiple-selection statement.



Good Programming Practice 8.11

In a switch statement, when the default clause is listed last, the break for that case statement is not required. Some programmers include this break for clarity and for symmetry with other cases.

Note that having several case labels listed together (e.g., case 1: case 2: with no statements between the cases) simply means that the same set of actions is to occur for each case. Again, note that, besides small circles and arrows, the flowchart contains only rectangle symbols and diamond symbols.

8.6 do...while Repetition Statement

The **do...while repetition statement** is similar to the **while** statement. In the **while** statement, the loop-continuation test occurs at the beginning of the loop, before the body of the loop executes. The **do...while** statement tests the loop-continuation condition *after* the loop body executes—therefore, *the loop body always executes at least once*. When a **do...while** terminates, execution continues with the statement after the **while** clause. Note that it is not necessary to use braces in a **do...while** statement if there is only one statement in the body. However, the braces usually are included, to avoid confusion between the **while** and **do...while** statements. For example,

```
while ( condition )
```

normally is regarded as the header to a **while** statement. A **do...while** statement with no braces around a single-statement body appears as

```
do
    statement
while ( condition );
```

which can be confusing. The last line—`while(condition);`—may be misinterpreted by the reader as a **while** statement containing an empty statement (the semicolon by itself). Thus, to avoid confusion, the **do...while** statement with a one-statement body is often written as follows:

```
do
{
    statement
} while ( condition );
```



Good Programming Practice 8.12

Some programmers always include braces in a do...while statement even if they are not necessary. This helps eliminate ambiguity between the while statement and the do...while statement containing a one-statement body.



Common Programming Error 8.7

*Infinite loops are caused when the loop-continuation condition never becomes false in a **while**, **for** or **do...while** statement. To prevent this, make sure that there is not a semicolon immediately after the header of a **while** or **for** statement. In a counter-controlled loop, make sure that the control variable is incremented (or decremented) in the body of the loop. In a sentinel-controlled loop, make sure that the sentinel value is eventually input.*

The script in Fig. 8.9 uses a do...while statement to display each of the six different XHTML heading types (h1 through h6). Line 12 declares control variable counter and initializes it to 1. Upon entering the do...while statement, lines 15–17 write a line of XHTML text in the document. The value of control variable counter is used to create the starting and ending header tags (e.g., <h1> and </h1>) and to create the line of text to display (e.g., This is an h1 level head). Line 18 increments the counter before the loop-continuation test occurs at the bottom of the loop.

The do...while flowchart in Fig. 8.10 makes it clear that the loop-continuation test does not occur until the action executes at least once.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.9: DoWhileTest.html -->
6  <!-- Using the do...while repetition statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Using the do...while Repetition Statement</title>
10         <script type = "text/javascript">
11             <!--
12                 var counter = 1;
13
14             do {
15                 document.writeln( "<h" + counter + ">This is " +
16                     "an h" + counter + " level head" + "</h" +
17                     counter + ">" );
18                 ++counter;
19             } while ( counter <= 6 );
20             // --
21         </script>
22
23     </head><body></body>
24 </html>
```

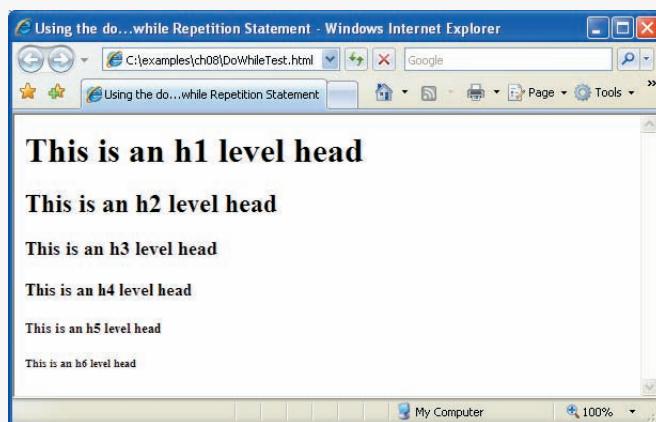


Fig. 8.9 | Using the do...while repetition statement.

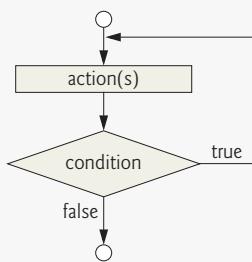


Fig. 8.10 | do...while repetition statement flowchart.

8.7 break and continue Statements

The **break** and **continue statements** alter the flow of control. The **break** statement, when executed in a **while**, **for**, **do...while** or **switch** statement, causes immediate exit from the statement. Execution continues with the first statement after the structure. The **break** statement is commonly used to escape early from a loop or to skip the remainder of a **switch** statement (as in Fig. 8.7). Figure 8.11 demonstrates the **break** statement in a **for** repetition statement.

During each iteration of the **for** statement in lines 14–20, the script writes the value of **count** in the XHTML document. When the **if** statement in line 16 detects that **count**

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.11: BreakTest.html -->
6  <!-- Using the break statement in a for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>
10             Using the break Statement in a for Statement
11         </title>
12         <script type = "text/javascript">
13             <!--
14             for ( var count = 1; count <= 10; ++count )
15             {
16                 if ( count == 5 )
17                     break; // break loop only if count == 5
18
19                 document.writeln( "Count is: " + count + "<br />" );
20             } //end for
21
22             document.writeln(
23                 "Broke out of loop at count = " + count );
24             // --
25         </script>
26     </head><body></body>
27 </html>
```

Fig. 8.11 | Using the **break** statement in a **for** statement. (Part 1 of 2.)

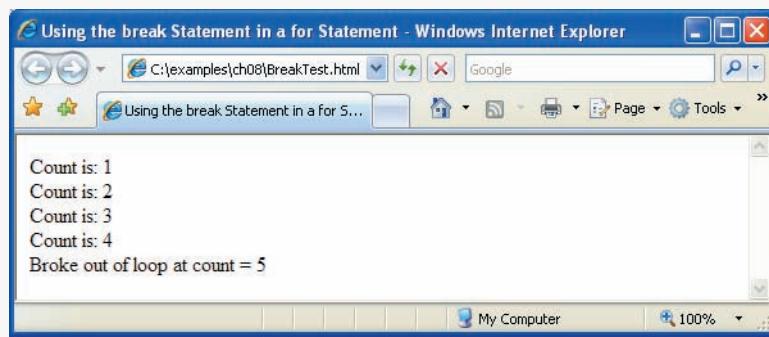


Fig. 8.11 | Using the break statement in a for statement. (Part 2 of 2.)

is 5, the break in line 17 executes. This statement terminates the for statement, and the program proceeds to line 22 (the next statement in sequence immediately after the for statement), where the script writes the value of count when the loop terminated (i.e., 5). The loop executes line 19 only four times.

The continue statement, when executed in a while, for or do...while statement, skips the remaining statements in the body of the statement and proceeds with the next iteration of the loop. In while and do...while statements, the loop-continuation test evaluates immediately after the continue statement executes. In for statements, the increment expression executes, then the loop-continuation test evaluates. This is the one case in which for and while differ. Improper placement of continue before the increment in a while may result in an infinite loop.

Figure 8.12 uses continue in a for statement to skip the document.writeln statement in line 20 when the if statement in line 17 determines that the value of count is 5. When the continue statement executes, the script skips the remainder of the for statement's body. Program control continues with the increment of the for statement's control variable, followed by the loop-continuation test to determine whether the loop should continue executing.



Software Engineering Observation 8.2

Some programmers feel that break and continue violate structured programming. They do not use break and continue, because the effects of these statements can be achieved by structured programming techniques.



Performance Tip 8.1

The break and continue statements, when used properly, perform faster than the corresponding structured techniques.



Software Engineering Observation 8.3

There is a tension between achieving quality software engineering and achieving the best-performing software. Often, one of these goals is achieved at the expense of the other. For all but the most performance-intensive situations, the following rule of thumb should be followed: First make your code simple, readable and correct; then make it fast and small, but only if necessary.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.12: ContinueTest.html -->
6 <!-- Using the continue statement in a for statement. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>
10    Using the continue Statement in a for Statement
11  </title>
12
13  <script type = "text/javascript">
14    <!--
15      for ( var count = 1; count <= 10; ++count )
16    {
17      if ( count == 5 )
18        continue; // skip remaining loop code only if count == 5
19
20      document.writeln( "Count is: " + count + "<br />" );
21    } //end for
22
23    document.writeln( "Used continue to skip printing 5" );
24    // --
25  </script>
26
27  </head><body></body>
28 </html>

```

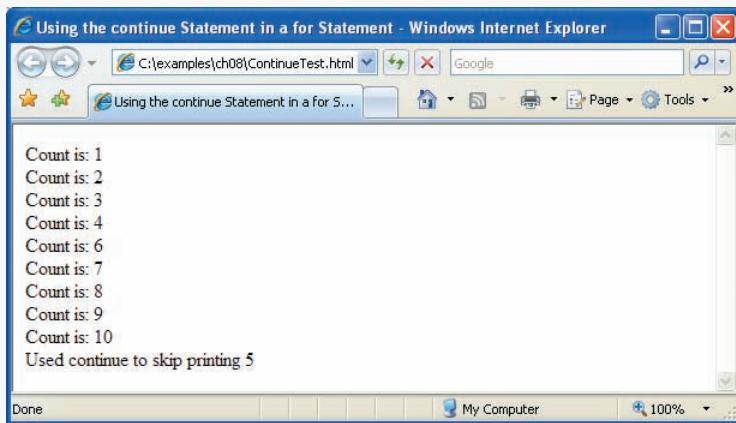


Fig. 8.12 | Using the continue statement in a for statement.

8.8 Labeled break and continue Statements

The **break** statement can break out of an immediately enclosing **while**, **for**, **do...while** or **switch** statement. To break out of a nested set of structures, you can use the **labeled break statement**. This statement, when executed in a **while**, **for**, **do...while** or **switch** statement, causes immediate exit from that statement and any number of enclosing repetition

statements; program execution resumes with the first statement after the enclosing **labeled statement** (a statement preceded by a label). The labeled statement can be a block (a set of statements enclosed in curly braces, {}). Labeled break statements commonly are used to terminate nested looping structures containing while, for, do...while or switch statements. Figure 8.13 demonstrates the labeled break statement in a nested for statement.

```

1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8.13: BreakLabelTest.html -->
6 <!-- Labeled break statement in a nested for statement. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9     <title>Using the break Statement with a Label</title>
10    <script type = "text/javascript">
11      <!--
12        stop: { // labeled block
13          for ( var row = 1; row <= 10; ++row )
14          {
15            for ( var column = 1; column <= 5 ; ++column )
16            {
17              if ( row == 5 )
18                break stop; // jump to end of stop block
19
20              document.write( "*" );
21            } //end for
22
23            document.writeln( "<br />" );
24          } //end for
25
26          // the following line is skipped
27          document.writeln( "This line should not print" );
28        } // end block labeled stop
29
30        document.writeln( "End of script" );
31        // -->
32      </script>
33    </head><body></body>
34 </html>
```

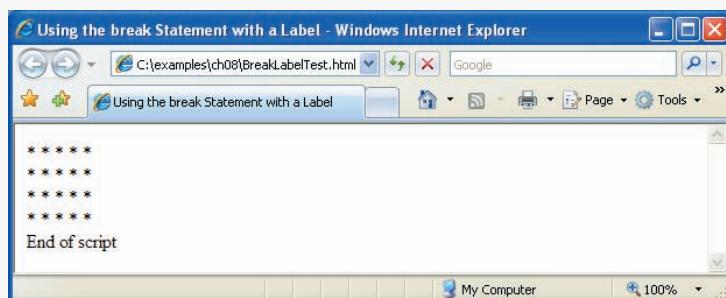


Fig. 8.13 | Labeled break statement in a nested for statement.

The labeled block (lines 12–28) begins with a **label** (an identifier followed by a colon). Here, we use the label `stop:`. The block is enclosed between the braces at the end of line 12 and in line 28, and includes both the nested `for` statement starting in line 13 and the `document.writeln` statement in line 27. When the `if` statement in line 17 detects that `row` is equal to 5, the statement in line 18 executes. This statement terminates both the `for` statement in line 15 and its enclosing `for` statement in line 13, and the program proceeds to the statement in line 30 (the first statement in sequence after the labeled block). The inner `for` statement executes its body only four times. Note that the `document.writeln` statement in line 27 never executes, because it is included in the labeled block and the outer `for` statement never completes.

The `continue` statement proceeds with the next iteration (repetition) of the immediately enclosing `while`, `for` or `do...while` statement. The **labeled continue statement**, when executed in a repetition statement (`while`, `for` or `do...while`), skips the remaining statements in the structure's body and any number of enclosing repetition statements, then proceeds with the next iteration of the enclosing **labeled repetition statement** (a repetition statement preceded by a label). In labeled `while` and `do...while` statements, the loop-continuation test evaluates immediately after the `continue` statement executes. In a labeled `for` statement, the increment expression executes, then the loop-continuation test evaluates. Figure 8.14 uses the labeled `continue` statement in a nested `for` statement to cause execution to continue with the next iteration of the outer `for` statement.

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.14: ContinueLabelTest.html -->
6  <!-- Labeled continue statement in a nested for statement. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Using the continue Statement with a Label</title>
10         <script type = "text/javascript">
11             <!--
12                 nextRow: // target label of continue statement
13                 for ( var row = 1; row <= 5; ++row )
14                 {
15                     document.writeln( "<br />" );
16
17                     for ( var column = 1; column <= 10; ++column )
18                     {
19                         if ( column > row )
20                             continue nextRow; // next iteration of labeled loop
21
22                         document.write( "* " );
23                     } //end for
24                 } //end for
25             // -->
26         </script>
27     </head><body></body>
28 </html>
```

Fig. 8.14 | Labeled `continue` statement in a nested `for` statement. (Part I of 2.)

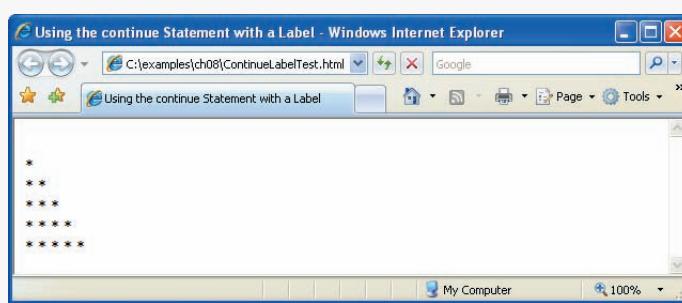


Fig. 8.14 | Labeled continue statement in a nested for statement. (Part 2 of 2.)

The labeled for statement (lines 13–24) starts with the `nextRow` label in line 12. When the if statement in line 19 in the inner for statement detects that `column` is greater than `row`, line 20 executes and program control continues with the increment of the control variable of the outer for statement. Even though the inner for statement counts from 1 to 10, the number of * characters output on a row never exceeds the value of `row`.

8.9 Logical Operators

So far, we have studied only such **simple conditions** as `count <= 10`, `total > 1000` and `number != sentinelValue`. These conditions were expressed in terms of the relational operators `>`, `<`, `>=` and `<=`, and in terms of the equality operators `==` and `!=`. Each decision tested one condition. To make a decision based on multiple conditions, we performed these tests in separate statements or in nested if or if...else statements.

JavaScript provides **logical operators** that can be used to form more complex conditions by combining simple conditions. The logical operators are `&&` (**logical AND**), `||` (**logical OR**) and `!` (**logical NOT**, also called **logical negation**). We consider examples of each of these operators.

Suppose that, at some point in a program, we wish to ensure that two conditions are *both* true before we choose a certain path of execution. In this case, we can use the logical `&&` operator, as follows:

```
if ( gender == 1 && age >= 65 )
    ++seniorFemales;
```

This if statement contains two simple conditions. The condition `gender == 1` might be evaluated to determine, for example, whether a person is a female. The condition `age >= 65` is evaluated to determine whether a person is a senior citizen. The if statement then considers the combined condition

```
gender == 1 && age >= 65
```

This condition is true *if and only if* both of the simple conditions are true. Finally, if this combined condition is indeed true, the count of `seniorFemales` is incremented by 1. If either or both of the simple conditions are false, the program skips the incrementing and proceeds to the statement following the if statement. The preceding combined condition can be made more readable by adding redundant parentheses:

```
( gender == 1 ) && ( age >= 65 )
```

The table in Fig. 8.15 summarizes the `&&` operator. The table shows all four possible combinations of `false` and `true` values for `expression1` and `expression2`. Such tables are often called **truth tables**. JavaScript evaluates to `false` or `true` all expressions that include relational operators, equality operators and/or logical operators.

Now let us consider the `||` (logical OR) operator. Suppose we wish to ensure that either *or* both of two conditions are `true` before we choose a certain path of execution. In this case, we use the `||` operator, as in the following program segment:

```
if ( semesterAverage >= 90 || finalExam >= 90 )
    document.writeln( "Student grade is A" );
```

This statement also contains two simple conditions. The condition `semesterAverage >= 90` is evaluated to determine whether the student deserves an “A” in the course because of a solid performance throughout the semester. The condition `finalExam >= 90` is evaluated to determine whether the student deserves an “A” in the course because of an outstanding performance on the final exam. The `if` statement then considers the combined condition

```
semesterAverage >= 90 || finalExam >= 90
```

and awards the student an “A” if either or both of the simple conditions are `true`. Note that the message “Student grade is A” is *not* printed only when both of the simple conditions are `false`. Figure 8.16 is a truth table for the logical OR operator (`||`).

The `&&` operator has a higher precedence than the `||` operator. Both operators associate from left to right. An expression containing `&&` or `||` operators is evaluated only until truth or falsity is known. Thus, evaluation of the expression

```
gender == 1 && age >= 65
```

expression1	expression2	expression1 && expression2
false	false	false
false	true	false
true	false	false
true	true	true

Fig. 8.15 | Truth table for the `&&` (logical AND) operator.

expression1	expression2	expression1 expression2
false	false	false
false	true	true
true	false	true
true	true	true

Fig. 8.16 | Truth table for the `||` (logical OR) operator.

stops immediately if gender is not equal to 1 (i.e., the entire expression is `false`) and continues if gender is equal to 1 (i.e., the entire expression could still be `true` if the condition `age >= 65` is true). Similarly, the `||` operator immediately returns `true` if the first operand is `true`. This performance feature for evaluation of logical AND and logical OR expressions is called **short-circuit evaluation**.

JavaScript provides the `!` (logical negation) operator to enable a programmer to “reverse” the meaning of a condition (i.e., a `true` value becomes `false`, and a `false` value becomes `true`). Unlike the logical operators `&&` and `||`, which combine two conditions (i.e., they are binary operators), the logical negation operator has only a single condition as an operand (i.e., it is a unary operator). The logical negation operator is placed before a condition to choose a path of execution if the original condition (without the logical negation operator) is `false`, as in the following program segment:

```
if ( ! ( grade == sentinelValue ) )
    document.writeln( "The next grade is " + grade );
```

The parentheses around the condition `grade == sentinelValue` are needed, because the logical negation operator has a higher precedence than the equality operator. Figure 8.17 is a truth table for the logical negation operator.

In most cases, the programmer can avoid using logical negation by expressing the condition differently with an appropriate relational or equality operator. For example, the preceding statement may also be written as follows:

```
if ( grade != sentinelValue )
    document.writeln( "The next grade is " + grade );
```

The script in Fig. 8.18 demonstrates all the logical operators by producing their truth tables. The script produces an XHTML table containing the results.

expression	<code>!expression</code>
<code>false</code>	<code>true</code>
<code>true</code>	<code>false</code>

Fig. 8.17 | Truth table for operator `!` (logical negation).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 8.18: LogicalOperators.html -->
6  <!-- Demonstrating logical operators. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9          <title>Demonstrating the Logical Operators</title>
```

Fig. 8.18 | Demonstrating logical operators. (Part I of 2.)

```

10 <style type = "text/css">
11   table { width: 100% }
12   td.left { width: 25% }
13 </style>
14 <script type = "text/javascript">
15   <!--
16   document.writeln(
17     "<table border = \"1\">";
18   document.writeln(
19     "<caption>Demonstrating Logical " +
20     "Operators</caption>";
21   document.writeln(
22     "<tr><td class = \"left\">Logical AND (&&)</td>" +
23     "<td>false && false: " + ( false && false ) +
24     "<br />false && true: " + ( false && true ) +
25     "<br />true && false: " + ( true && false ) +
26     "<br />true && true: " + ( true && true ) +
27     "</td></tr>" );
28   document.writeln(
29     "<tr><td class = \"left\">Logical OR (||)</td>" +
30     "<td>false || false: " + ( false || false ) +
31     "<br />false || true: " + ( false || true ) +
32     "<br />true || false: " + ( true || false ) +
33     "<br />true || true: " + ( true || true ) +
34     "</td></tr>" );
35   document.writeln(
36     "<tr><td class = \"left\">Logical NOT (!)</td>" +
37     "<td>!false: " + ( !false ) +
38     "<br />!true: " + ( !true ) + "</td></tr>" );
39   document.writeln( "</table>" );
40   // -->
41 </script>
42 </head><body></body>
43 </html>

```

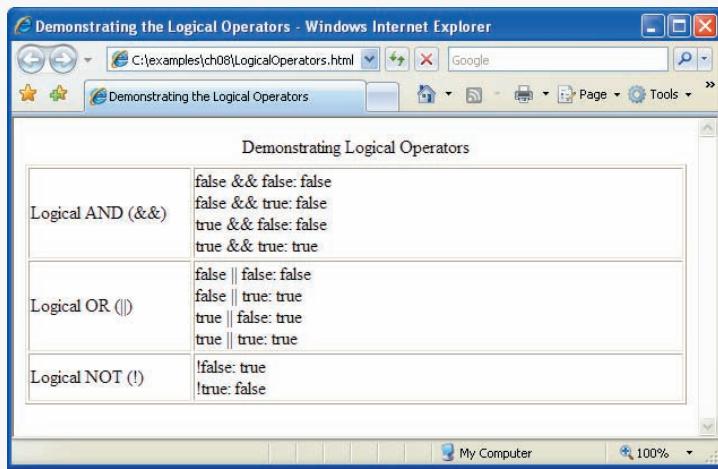


Fig. 8.18 | Demonstrating logical operators. (Part 2 of 2.)

In the output of Fig. 8.18, the strings "false" and "true" indicate false and true for the operands in each condition. The result of the condition is shown as true or false. Note that when you use the concatenation operator with a boolean value and a string, JavaScript automatically converts the boolean value to string "false" or "true". Lines 16–39 build an XHTML table containing the results.

An interesting feature of JavaScript is that most nonboolean values can be converted to a boolean true or false value. Nonzero numeric values are considered to be true. The numeric value zero is considered to be false. Any string that contains characters is considered to be true. The empty string (i.e., the string containing no characters) is considered to be false. The value null and variables that have been declared but not initialized are considered to be false. All objects (e.g., the browser's document and window objects and JavaScript's Math object) are considered to be true.

Figure 8.19 shows the precedence and associativity of the JavaScript operators introduced up to this point. The operators are shown top to bottom in decreasing order of precedence.

Operator	Associativity	Type
++ -- !	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical AND
	left to right	logical OR
? :	right to left	conditional
= += -= *= /= %=	right to left	assignment

Fig. 8.19 | Precedence and associativity of the operators discussed so far.

8.10 Summary of Structured Programming

Just as architects design buildings by employing the collective wisdom of their profession, so should programmers design programs. Our field is younger than architecture, and our collective wisdom is considerably sparser. We have learned that structured programming produces programs that are easier to understand than unstructured programs, and thus are easier to test, debug, and modify.

Flowcharts reveal the structured nature of programs or the lack thereof. Connecting individual flowchart symbols arbitrarily can lead to unstructured programs. Therefore, the programming profession has chosen to combine flowchart symbols to form a limited set of control structures and to build structured programs by properly combining control structures in two simple ways.

For simplicity, only single-entry/single-exit control structures are used—that is, there is only one way to enter and one way to exit each control structure. Connecting control structures in sequence to form structured programs is simple: The exit point of one control structure is connected to the entry point of the next control structure (i.e., the control structures are simply placed one after another in a program). We have called this process control-structure stacking. The rules for forming structured programs also allow for control structures to be nested. Figure 8.20 summarizes JavaScript's control structures. Small circles are used in the figure to indicate the single entry point and the single exit point of each structure.

Figure 8.21 shows the rules for forming properly structured programs. The rules assume that the rectangle flowchart symbol may be used to indicate any action, including input/output. [Note: An oval flowchart symbol indicates the beginning and end of a process.]

Applying the rules in Fig. 8.21 always results in a structured flowchart with a neat, building-block-like appearance. For example, repeatedly applying Rule 2 to the simplest

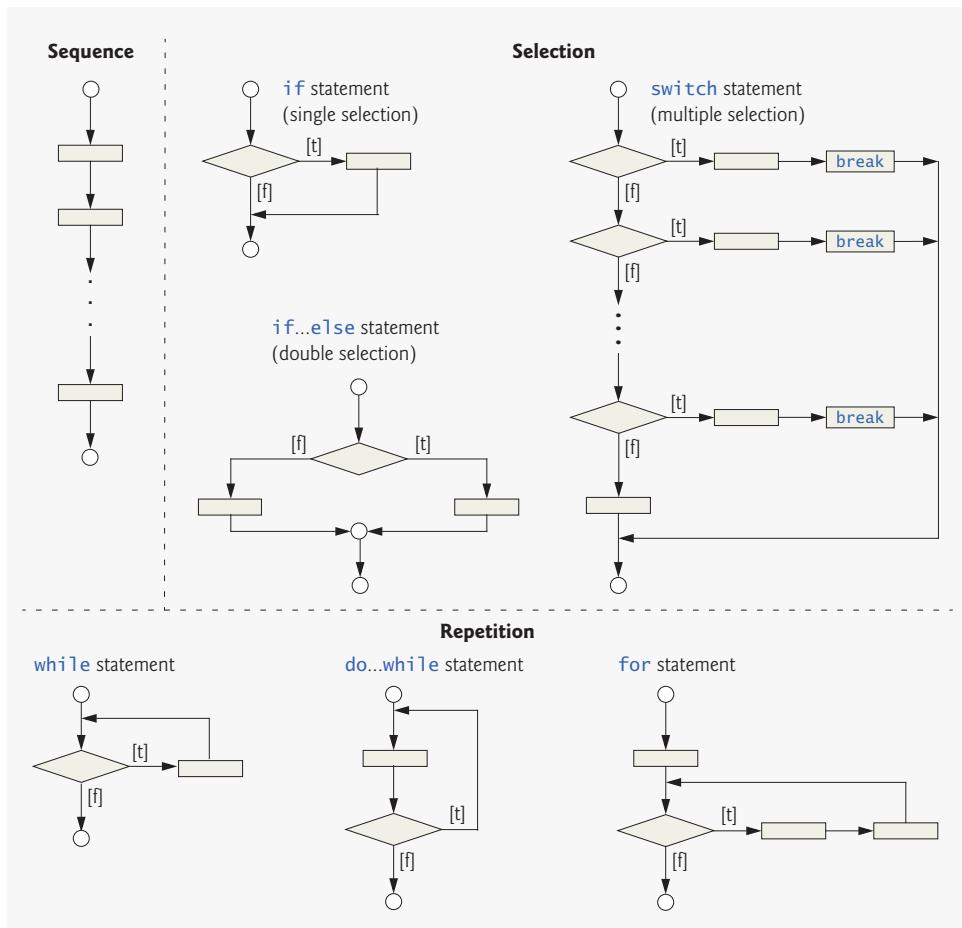


Fig. 8.20 | Single-entry/single-exit sequence, selection and repetition structures.

Rules for forming structured programs

1. Begin with the “simplest flowchart” (Fig. 8.22).
2. Any rectangle (action) can be replaced by two rectangles (actions) in sequence.
3. Any rectangle (action) can be replaced by any control structure (sequence, if, if...else, switch, while, do...while or for).
4. Rules 2 and 3 may be applied as often as necessary and in any order.

Fig. 8.21 | Forming rules for structured programs.

flowchart (Fig. 8.22) results in a structured flowchart containing many rectangles in sequence (Fig. 8.23). Note that Rule 2 generates a stack of control structures; so let us call Rule 2 the **stacking rule**.

Rule 3 is called the **nesting rule**. Repeatedly applying Rule 3 to the simplest flowchart results in a flowchart with neatly nested control structures. For example, in Fig. 8.24, the rectangle in the simplest flowchart is first replaced with a double-selection (if...else) structure. Then Rule 3 is applied again to both of the rectangles in the double-selection structure by replacing each of these rectangles with double-selection structures. The

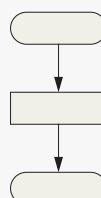


Fig. 8.22 | Simplest flowchart.

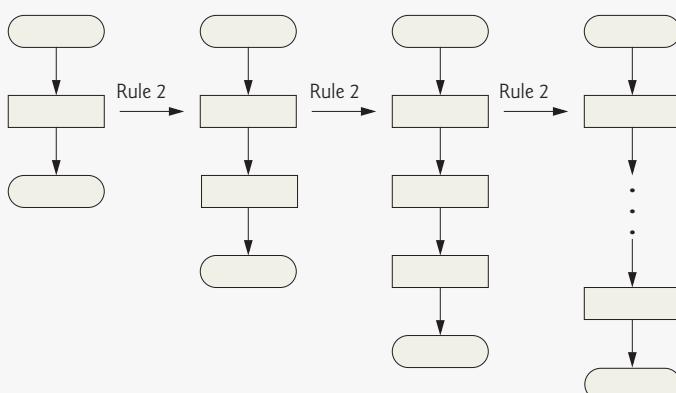


Fig. 8.23 | Repeatedly applying Rule 2 of Fig. 8.21 to the simplest flowchart.

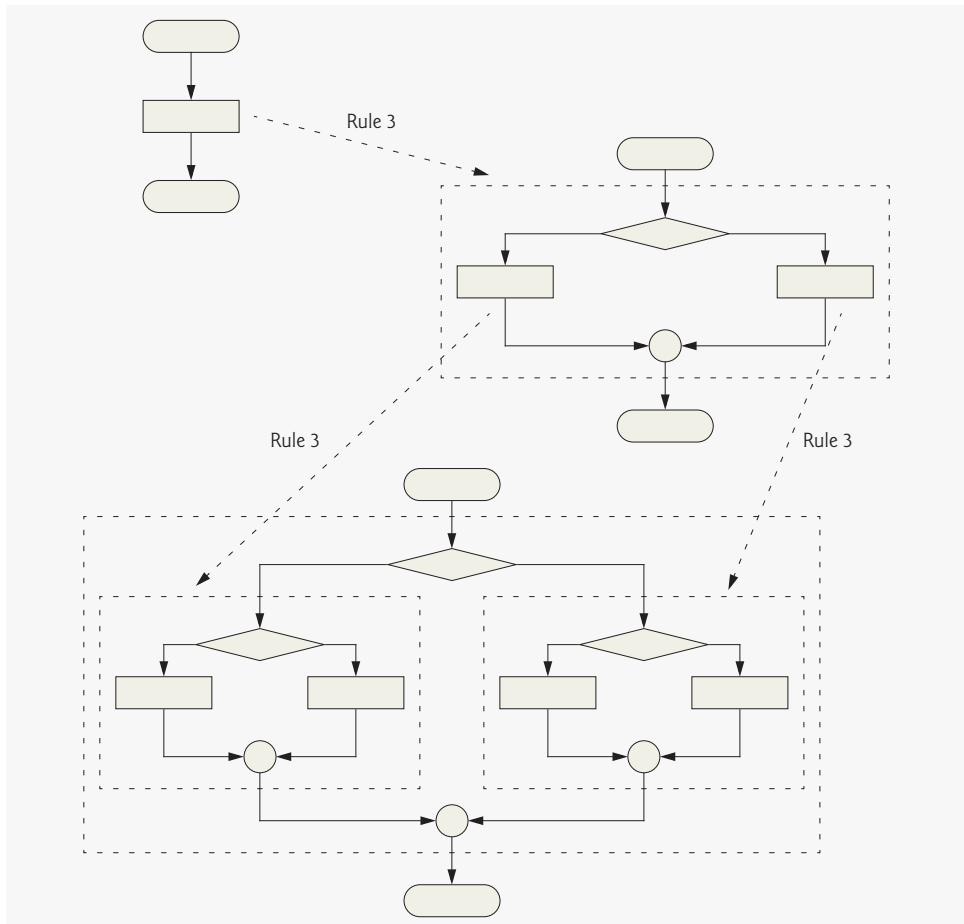


Fig. 8.24 | Applying Rule 3 of Fig. 8.21 to the simplest flowchart.

dashed box around each of the double-selection structures represents the rectangle in the original simplest flowchart that was replaced.

Rule 4 generates larger, more involved and more deeply nested structures. The flowcharts that emerge from applying the rules in Fig. 8.21 constitute the set of all possible structured flowcharts and thus the set of all possible structured programs.

The beauty of the structured approach is that we use only seven simple single-entry/single-exit pieces and assemble them in only two simple ways. Figure 8.25 shows the kinds of stacked building blocks that emerge from applying Rule 2 and the kinds of nested building blocks that emerge from applying Rule 3. The figure also shows the kind of overlapped building blocks that cannot appear in structured flowcharts (because of the elimination of the goto statement).

If the rules in Fig. 8.21 are followed, an unstructured flowchart (like the one in Fig. 8.26) cannot be created. If you are uncertain about whether a particular flowchart is structured, apply the rules of Fig. 8.21 in reverse to try to reduce the flowchart to the sim-

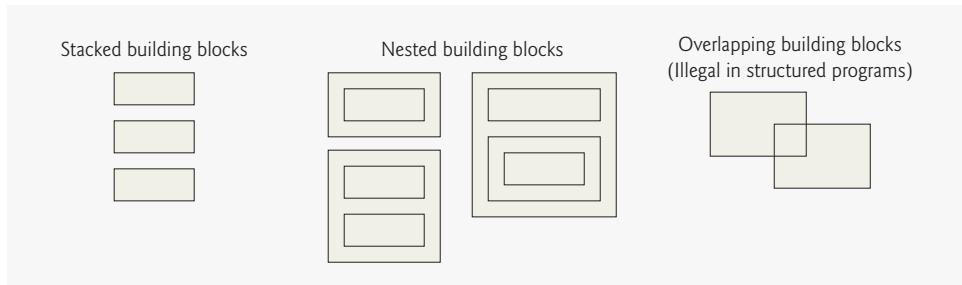


Fig. 8.25 | Stacked, nested and overlapped building blocks.

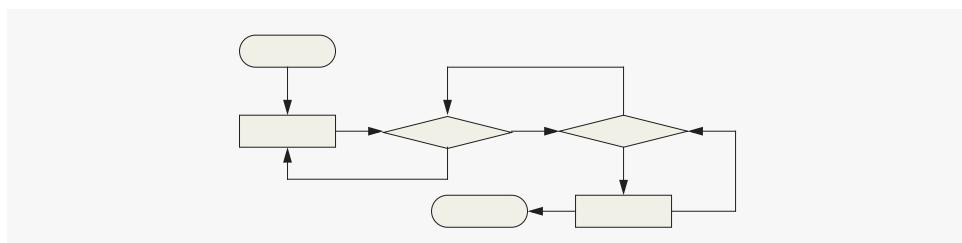


Fig. 8.26 | Unstructured flowchart.

pletest flowchart. If the flowchart is reducible to the simplest flowchart, the original flowchart is structured; otherwise, it is not.

Structured programming promotes simplicity. Bohm and Jacopini have given us the result that only three forms of control are needed:

- sequence
- selection
- repetition

Sequence is trivial. Selection is implemented in one of three ways:

- `if` statement (single selection)
- `if...else` statement (double selection)
- `switch` statement (multiple selection)

In fact, it is straightforward to prove that the `if` statement is sufficient to provide any form of selection; everything that can be done with the `if...else` statement and the `switch` statement can be implemented by combining `if` statements (although perhaps not as smoothly).

Repetition is implemented in one of four ways:

- `while` statement
- `do...while` statement
- `for` statement
- `for...in` statement (discussed in Chapter 10)

It is straightforward to prove that the `while` statement is sufficient to provide any form of repetition. Everything that can be done with the `do...while` statement and the `for` statement can be done with the `while` statement (although perhaps not as elegantly).

Combining these results illustrates that any form of control ever needed in a JavaScript program can be expressed in terms of:

- sequence
- `if` statement (selection)
- `while` statement (repetition)

These control structures can be combined in only two ways—stacking and nesting. Indeed, structured programming promotes simplicity.

8.11 Wrap-Up

In this chapter, we discussed the composition of programs from control structures containing actions and decisions. We expanded the previous chapter's discussion on control structures, discussing the `for` statement, the `do...while` statement and the `switch` statement. We also saw how `break` and `continue` can be used to alter the flow of the program.

In Chapter 9, we introduce another program-structuring unit, called the function. You'll learn to compose large programs by combining functions that are composed of control structures. We will also discuss how functions promote software reusability.

8.12 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 and more. Also check out the tutorials for all skill levels, from introductory to advanced. Be sure to visit the related Resource Centers on XHTML (www.deitel.com/xhtml/) and CSS 2.1 (www.deitel.com/css21/).

Summary

Section 8.2 Essentials of Counter-Controlled Repetition

- Counter-controlled repetition requires: the *name* of a control variable, the *initial value* of the control variable, the *increment* (or *decrement*) by which the control variable is modified each time through the loop, and the condition that tests for the *final value* of the control variable to determine whether looping should continue.
- Because the double-quote character delimits the beginning and end of a string literal in JavaScript, it cannot be used in the contents of the string unless it is preceded by a \ to create the escape sequence \" . XHTML allows either single quotes (') or double quotes (") to be placed around the value specified for an attribute. JavaScript allows single quotes to be placed in a string literal.

Section 8.3 for Repetition Statement

- The `for` statement “does it all”—it specifies each of the items needed for counter-controlled repetition with a control variable.

- You can use a block to put multiple statements into the body of a `for` construct.
- The `for` statement takes three expressions: an initialization, a condition and an expression.
- The increment expression in the `for` statement acts like a stand-alone statement at the end of the body of the `for` statement.
- Place only expressions involving the control variable in the initialization and increment sections of a `for` statement.
- The three expressions in the `for` statement are optional. The two semicolons in the `for` statement are required.
- The initialization, loop-continuation condition and increment portions of a `for` statement can contain arithmetic expressions.
- The part of a script in which a variable name can be used is known as the variable's scope.
- The "increment" of a `for` statement may be negative, in which case it is called a decrement and the loop actually counts downward.
- If the loop-continuation condition initially is `false`, the body of the `for` statement is not performed. Instead, execution proceeds with the statement following the `for` statement.

Section 8.4 Examples Using the `for` Statement

- JavaScript does not include an exponentiation operator. Instead, we use the `Math` object's `pow` method for this purpose. `Math.pow(x, y)` calculates the value of `x` raised to the `y`th power.
- Floating-point numbers can cause trouble as a result of rounding errors.

Section 8.5 `switch` Multiple-Selection Statement

- JavaScript provides the `switch` multiple-selection statement in which a variable or expression is tested separately for each of the values it may assume. Different actions are taken for each value.
- The CSS property `list-style-type` allows you to set the numbering system for the list. Possible values include `decimal` (numbers—the default), `lower-roman` (lowercase roman numerals), `upper-roman` (uppercase roman numerals), `lower-alpha` (lowercase letters), `upper-alpha` (uppercase letters), and several others.
- The `switch` statement consists of a series of case labels and an optional default case. When the flow of control reaches the `switch` statement, the script evaluates the controlling expression in the parentheses following keyword `switch`. The value of this expression is compared with the value in each of the case labels, starting with the first case label. If the comparison evaluates to `true`, the statements after the case label are executed in order until a `break` statement is reached.
- The `break` statement is used as the last statement in each case to exit the `switch` statement immediately.
- The `default` case allows you to specify a set of statements to execute if no other case is satisfied. This case is usually the last case in the `switch` statement.
- Each case can have multiple actions (statements). The `switch` statement is different from other statements in that braces are not required around multiple actions in a case of a `switch`.
- The `break` statement is not required for the last case in the `switch` statement because program control automatically continues with the next statement after the `switch` statement.
- Having several case labels listed together (e.g., `case 1: case 2:` with no statements between the cases) simply means that the same set of actions is to occur for each case.

Section 8.6 `do...while` Repetition Statement

- The `do...while` statement tests the loop-continuation condition *after* the loop body executes—therefore, *the loop body always executes at least once*.

Section 8.7 ***break*** and ***continue*** Statements

- The `break` statement, when executed in a `while`, `for`, `do...while` or `switch` statement, causes immediate exit from the statement. Execution continues with the first statement after the structure.
- The `break` statement is commonly used to escape early from a loop or to skip the remainder of a `switch` statement.
- The `continue` statement, when executed in a `while`, `for` or `do...while` statement, skips the remaining statements in the body of the statement and proceeds with the next iteration of the loop. In `while` and `do...while` statements, the loop-continuation test evaluates immediately after the `continue` statement executes. In `for` statements, the increment expression executes, then the loop-continuation test evaluates.

Section 8.8 Labeled ***break*** and ***continue*** Statements

- To break out of a nested control statement, you can use the labeled `break` statement. This statement, when executed in a `while`, `for`, `do...while` or `switch` statement, causes immediate exit from that statement and any number of enclosing repetition statements; program execution resumes with the first statement after the specified labeled statement (a statement preceded by a label).
- A labeled statement can be a block (a set of statements enclosed in curly braces, `{}`).
- Labeled `break` statements commonly are used to terminate nested looping structures containing `while`, `for`, `do...while` or `switch` statements.
- The labeled `continue` statement, when executed in a repetition statement (`while`, `for` or `do...while`), skips the remaining statements in the structure's body and any number of enclosing repetition statements, then proceeds with the next iteration of the specified labeled repetition statement (a repetition statement preceded by a label).
- In labeled `while` and `do...while` statements, the loop-continuation test evaluates immediately after the `continue` statement executes. In a labeled `for` statement, the increment expression executes, then the loop-continuation test evaluates.

Section 8.9 Logical Operators

- JavaScript provides logical operators that can be used to form more complex conditions by combining simple conditions. The logical operators are `&&` (logical AND), `||` (logical OR) and `!` (logical NOT, also called logical negation).
- The `&&` operator is used to ensure that two conditions are *both* `true` before choosing a certain path of execution.
- JavaScript evaluates to `false` or `true` all expressions that include relational operators, equality operators and/or logical operators.
- The `||` (logical OR) operator is used to ensure that either *or* both of two conditions are `true` before choosing choose a certain path of execution.
- The `&&` operator has a higher precedence than the `||` operator. Both operators associate from left to right.
- An expression containing `&&` or `||` operators is evaluated only until truth or falsity is known. This is called short-circuit evaluation.
- JavaScript provides the `!` (logical negation) operator to enable a programmer to “reverse” the meaning of a condition (i.e., a `true` value becomes `false`, and a `false` value becomes `true`).
- The logical negation operator has only a single condition as an operand (i.e., it is a unary operator). The logical negation operator is placed before a condition to evaluate to `true` if the original condition (without the logical negation operator) is `false`.
- The logical negation operator has a higher precedence than the equality operator.

- Most nonboolean values can be converted to a boolean `true` or `false` value. Nonzero numeric values are considered to be `true`. The numeric value zero is considered to be `false`. Any string that contains characters is considered to be `true`. The empty string (i.e., the string containing no characters) is considered to be `false`. The value `null` and variables that have been declared but not initialized are considered to be `false`. All objects (e.g., the browser's `document` and `window` objects and JavaScript's `Math` object) are considered to be `true`.

Terminology

<code>!</code> operator	loop-continuation condition
<code>&&</code> operator	<code>Math</code> object
<code> </code> operator	multiple selection
<code>break</code>	nested control structure
<code>case</code> label	nesting rule
<code>continue</code>	<code>Number</code> object
counter-controlled repetition	off-by-one error
<code>default</code> case in <code>switch</code>	<code>pow</code> method of the <code>Math</code> object
<code>do...while</code> repetition statement	repetition structure
<code>for</code> repetition statement	scope
<code>for</code> statement header	short-circuit evaluation
infinite loop	simple condition
labeled <code>break</code> statement	single-entry/single-exit control structure
labeled block	stacked control structure
labeled <code>continue</code> statement	stacking rule
labeled repetition statement	<code>switch</code> selection statement
logical AND (<code>&&</code>)	<code>toFixed</code> method
logical negation (<code>!</code>)	truth tables
logical operator	<code>while</code> repetition statement
logical OR (<code> </code>)	zero-based counting

Self-Review Exercises

- 8.1** State whether each of the following is *true* or *false*. If *false*, explain why.
- The `default` case is required in the `switch` selection statement.
 - The `break` statement is required in the last case of a `switch` selection statement.
 - The expression `(x > y && a < b)` is true if either `x > y` is true or `a < b` is true.
 - An expression containing the `||` operator is true if either or both of its operands is true.
- 8.2** Write a JavaScript statement or a set of statements to accomplish each of the following tasks:
- Sum the odd integers between 1 and 99. Use a `for` statement. Assume that the variables `sum` and `count` have been declared.
 - Calculate the value of 2.5 raised to the power of 3. Use the `pow` method.
 - Print the integers from 1 to 20 by using a `while` loop and the counter variable `x`. Assume that the variable `x` has been declared, but not initialized. Print only five integers per line.
[Hint: Use the calculation `x % 5`. When the value of this expression is 0, use `document.write("
 ")` to output a line break in the XHTML document.]
 - Repeat Exercise 8.2 (c), but using a `for` statement.
- 8.3** Find the error in each of the following code segments, and explain how to correct it:
- ```
x = 1;
while (x <= 10);
 ++
}
```

- b) `for ( y = .1; y != 1.0; y += .1 )  
 document.write( y + " " );`
- c) `switch ( n )  
{  
 case 1:  
 document.writeln( "The number is 1" );  
 case 2:  
 document.writeln( "The number is 2" );  
 break;  
 default:  
 document.writeln( "The number is not 1 or 2" );  
 break;  
}`
- d) The following code should print the values from 1 to 10:  
`n = 1;  
while ( n < 10 )  
 document.writeln( n++ );`

## Answers to Self-Review Exercises

- 8.1** a) False. The `default` case is optional. If no default action is needed, then there is no need for a `default` case. b) False. The `break` statement is used to exit the `switch` statement. The `break` statement is not required for the last case in a `switch` statement. c) False. Both of the relational expressions must be true in order for the entire expression to be true when using the `&&` operator. d) True.

- 8.2** a) `sum = 0;  
for ( count = 1; count <= 99; count += 2 )  
 sum += count;`
- b) `Math.pow( 2.5, 3 )`
- c) `x = 1;  
while ( x <= 20 ) {  
 document.write( x + " " );  
 if ( x % 5 == 0 )  
 document.write( "<br />" );  
 ++x;  
}`
- d) `for ( x = 1; x <= 20; x++ ) {  
 document.write( x + " " );  
  
 if ( x % 5 == 0 )  
 document.write( "<br />" );  
}  
or  
for ( x = 1; x <= 20; x++ )  
  
if ( x % 5 == 0 )  
 document.write( x + "<br />" );  
else  
 document.write( x + " " );`

- 8.3**
- a) Error: The semicolon after the `while` header causes an infinite loop, and there is a missing left brace.  
Correction: Replace the semicolon by a `{`, or remove both the `;` and the `}`.
  - b) Error: Using a floating-point number to control a `for` repetition statement may not work, because floating-point numbers are represented approximately by most computers.  
Correction: Use an integer, and perform the proper calculation to get the values you desire:
- ```
for ( y = 1; y != 10; y++ )
    document.writeln( ( y / 10 ) + " " );
```
- c) Error: Missing `break` statement in the statements for the first `case`.
Correction: Add a `break` statement at the end of the statements for the first `case`. Note that this missing statement is not necessarily an error if the programmer wants the statement of `case 2`: to execute every time the `case 1:` statement executes.
 - d) Error: Improper relational operator used in the `while` continuation condition.
Correction: Use `<=` rather than `<`, or change 10 to 11.

Exercises

- 8.4** Find the error in each of the following segments of code. [Note: There may be more than one error.]

- a) `For (x = 100, x >= 1, x++)`
 `document.writeln(x);`
- b) The following code should print whether integer value is odd or even:

```
switch ( value % 2 ) {
    case 0:
        document.writeln( "Even integer" );
    case 1:
        document.writeln( "Odd integer" );
}
```
- c) The following code should output the odd integers from 19 to 1:

```
for ( x = 19; x >= 1; x += 2 )
    document.writeln( x );
```
- d) The following code should output the even integers from 2 to 100:

```
counter = 2;
do {
    document.writeln( counter );
    counter += 2;
} while ( counter < 100 );
```

- 8.5** What does the following script do?

```

1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Exercise 8.5: ex08_05.html -->
6  <html xmlns = "http://www.w3.org/1999/xhtml">
7      <head><title>Mystery</title>
8      <script type = "text/javascript">
9          <!--
10             document.writeln( "<table>" );
11

```