

LEMAY

SAMS TEACH
YOURSELF
WEB
PUBLISHING
WITH HTML 4
IN 21 DAYS

T MEADE
GenColl

QA 76

.76

.H94 L44

2000

Copy 1

SECOND EDITION
Includes coverage of XHTML

Laura Lemay

“If you’re going to read
only one book on HTML,
make it this one.”

—Pittsburgh Post-Gazette

SAMS
Teach Yourself
Web Publishing
with HTML 4

SAMS

in 21 Days

Sams Teach Yourself Web Publishing with HTML 4 in 21 Days

PART 1	PART 2	PART 3	PART 4	PART 5	PART 6	PART 7
GETTING STARTED	CREATING SIMPLE WEB PAGES	WEB GRAPHICS	DOING MORE WITH HTML	MULTIMEDIA, FORMS, AND DYNAMIC HTML	DESIGNING EFFECTIVE WEB PAGES	GOING LIVE ON THE WEB
DAY 1 The World of the World Wide Web	DAY 4 Begin with the Basics	DAY 7 Using Images, Color, and Backgrounds	DAY 10 Style Sheets	DAY 13 Multimedia: Adding Sound, Video, and More	DAY 16 Writing and Designing Web Pages: Dos and Don'ts	DAY 19 Putting Your Site Online
DAY 2 Get Organized	DAY 5 All About Links	DAY 8 Creating Animated Graphics	DAY 11 Tables	DAY 14 Designing Forms	DAY 17 Examples of Good and Bad Web Design	DAY 20 Letting People Know It's There
DAY 3 An Introduction to HTML	DAY 6 More Text Formatting with HTML	DAY 9 Creating and Using Imagemaps	DAY 12 Frames and Linked Windows	DAY 15 Using Dynamic HTML	DAY 18 Designing for the Real World	DAY 21 Testing, Revising, and Maintaining Your Site

HTML Quick Reference

Structure Tags

<!--...-->	Creates a comment
<html>...</html>	Encloses the entire HTML document
<head>...</head>	Encloses the head of the HTML document
<isindex>	Indicates the document is a gateway script that allows searches
<meta>	Provides general information about the document
<style>...</style>	Style information
<script>...</script>	Scripting language
<noscript>...</noscript>	Alternative content when scripting not supported
<title>...</title>	The title of the document
<body>...</body>	Encloses the body (text and tags) of the HTML document

Headings

...	Headings 1 through 6
...	
...	
...	
...	
...	

Paragraphs

...	A plain paragraph; </p> is optional
------------	-------------------------------------

Links

<a>...	Creates a link or anchor Includes common attributes
href="..."	The URL of the document to be linked to this one
name="..."	The name of the anchor
target="..."	Identifies the window or location to open the link in
rel="..."	Defines forward link types
rev="..."	Defines reverse link types
accesskey="..."	Determines the accessibility character
shape="..."	Is for use with object shapes

Links

coords="..."	Is for use with object shapes
tabindex="..."	Determines the tabbing order
onClick	Is an intrinsic event
onmouseover	Is an intrinsic event
onmouseout	Is an intrinsic event

Lists

...	An ordered (numbered) list
...	An unordered (bulleted) list
<menu>...</menu>	A menu list of items
<dir>...</dir>	A directory listing
	A list item
<dl>...</dl>	A definition or glossary list
<dt>	A definition term
<dd>	The corresponding definition to a definition term

Character Formatting

...	Emphasis (usually italic)
...	Stronger emphasis (usually bold)
<code>...</code>	Code sample
<kbd>...</kbd>	Text to be typed
<var>...</var>	A variable or placeholder for some other value
<samp>...</samp>	Sample text
<dfn>...</dfn>	A definition of a term
<cite>...</cite>	A citation
...	Boldface text
<i>...</i>	Italic text
<tt>...</tt>	Typewriter font
<u>...</u>	Underlined text
<pre>...</pre>	Preformatted text

Other Elements

<hr />	A horizontal rule line
 	A line break
<blockquote>...</blockquote>	Used for long quotes or citations
<address>...</address>	Signatures or general information about a document's author

Other Elements

<code>...</code>	Change the size and color of the font
<code>size="..."</code>	The size of the font, from 1 to 7
<code>color="..."</code>	The font color
<code>face="..."</code>	The font type
<code><basefont></code>	Sets the default size of the font for the current page
<code>size="..."</code>	The default size of the font, from 1 to 7

Images

<code></code>	Inserts an inline image into the document: includes common attributes
<code>ismap</code>	This image is a server-side imagemap
<code>usemap</code>	This image is a client-side imagemap
<code>src="..."</code>	The URL of the image
<code>alt="..."</code>	A text string that will be displayed in browsers that cannot support images
<code>align="..."</code>	Determines the alignment of the given image
<code>height="..."</code>	Is the suggested height in pixels
<code>width="..."</code>	Is the suggested width in pixels
<code>vspace="..."</code>	The space between the image and the text above or below it
<code>hspace="..."</code>	The space between the image and the text to its left or right

Frames

<code><frameset>...</frameset></code>	Defines a frameset
<code>rows="..."</code>	Number of rows in frame
<code>cols="..."</code>	Number of columns in frame
<code>onload</code>	Is an intrinsic event
<code>onunload</code>	Is an intrinsic event
<code><frame></code>	Creates a frame
<code>name="..."</code>	Is the name of target frame
<code>src="..."</code>	Calls the frame content source
<code>frameborder="..."</code>	Determines the frame border
<code>marginwidth="..."</code>	Defines margin widths
<code>marginheight="..."</code>	Defines margin heights
<code>noresize="..."</code>	Determines ability to resize frames
<code>scrolling="..."</code>	Determines ability to scroll within frames

Frames

<code><iframe>...</iframe></code>	Defines an inline frame
<code><noframes>...</noframes></code>	Alternate content when frames not supported

Tables

<code><table>...</table></code>	Creates a table
<code>border="..."</code>	Width of the border in pixels
<code>cols="..."</code>	Number of columns
<code>cellspacing="..."</code>	Spacing between cells
<code>cellpadding="..."</code>	Spacing in cells
<code>width="..."</code>	Table width
<code><caption>...</caption></code>	The caption for the table
<code><tr>...</tr></code>	A table row
<code>align="..."</code>	The horizontal alignment of the contents of the cells within this row: possible values are LEFT, RIGHT, CENTER, and JUSTIFY
<code>valign="..."</code>	The vertical alignment of the contents of the cells within this row: possible values are TOP, MIDDLE, BOTTOM, and BASELINE
<code><th>...</th></code>	A table heading cell
<code>align="..."</code>	The horizontal alignment of the contents of the cell
<code>valign="..."</code>	The vertical alignment of the contents of the cell
<code>rowspan="..."</code>	The number of rows this cell will span
<code>colspan="..."</code>	The number of columns this cell will span
<code>nowrap</code>	Do not automatically wrap the contents of this cell
<code><td>...</td></code>	Defines a table data cell
<code>align="..."</code>	The horizontal alignment of the contents of the cell
<code>valign="..."</code>	The vertical alignment of the contents of the cell
<code>rowspan="..."</code>	The number of rows this cell will span
<code>colspan="..."</code>	The number of columns this cell will span
<code>nowrap</code>	Do not automatically wrap the contents of this cell

Acclaim for Laura Lemay's *Sams Teach Yourself Web Publishing with HTML*

"There are some good HTML primers on the Web itself, but if you're like me, you'll find it easier to learn by cracking a book. The best I've found is Laura Lemay's *[Sams] Teach Yourself Web Publishing with HTML*."

—Marc Frons, *Business Week*

"Laura Lemay delivers on her title's promise. By following her clear sequence of explanations, examples, and exercises, even the absolute Web novice can create serviceable documents within a few days. Better, she moves quickly beyond mechanics to techniques and tools for designing maximally effective and attractive presentations in spite of the medium's limitations."

—Michael K. Stone, *Whole Earth Review*

"If you're going to read only one book on HTML, and want it to be as comprehensive as can be, then make it this one."

—Mitch Gitman, *Pittsburgh Post-Gazette*

"Of all the HTML books out right now, I think Lemay's is the best, and I recommend it."

—Nancy McGough, *Infinite Ink*

"If you are looking for an easy-to-read introduction to HTML, this book is for you. Lemay has a clear understanding of what works and what doesn't, and she conveys her thoughts in a concise, orderly fashion."

—Robert Stewart, *The Virtual Mirror*

"If you want to create a Web page, or even if you already have created one, and you want a *great* book to help you understand it all, check out Laura Lemay's *[Sams] Teach Yourself Web Publishing with HTML*. I've used mine so much I practically know it by heart!"

—Camille Tillman, *Book Stacks Unlimited*

"All in all, this is a quality 'do-it-yourself' book for beginners of HTML publishing. The ABCs of HTML are explained clearly, and the exercises are instructive and easy to follow."

—Jim Duber, *Chorus*

"This is a very thorough book on HTML, and quite accurate. Laura Lemay is a good technical writer who explains things well. This is the book I wish I'd had when I started to learn HTML."

—Bob Cunningham, University of Hawaii

"My best recommendation goes to this book by Laura Lemay, entitled [*Sams*] *Teach Yourself Web Publishing with HTML*. There is simply no better book available, and many that are much worse than this one. If you study it, you will know more than enough to create stunning Web pages of your own."

—Bob Bickford

"If you want a good book that will help you understand how everything is really working, take a look at Sams Publishing's *Teach Yourself Web Publishing with HTML*. It is a very well-written book, and it includes all the information in an easy to understand format."

—Ron Loewy, HyperAct, Inc.

"There's a superb new book about HTML called [*Sams*] *Teach Yourself Web Publishing with HTML*.... It is very thorough and well-laid out."

—Michael MacDonald

"I think Lemay's first book can take some degree of credit for the growth of the Web itself. I wonder how many of the tens of thousands of home pages created in the past six months have been done by folks with dog-eared copies of [*Sams*] *Teach Yourself Web Publishing* within arm's reach."

—Dave Elliott, The Web Academy

Laura Lemay
With revisions by Denise Tyler



SAMS
Teach Yourself

Web Publishing with HTML 4

in 21 Days

SECOND EDITION

SAMS
A Division of Macmillan USA
201 West 103rd St., Indianapolis, Indiana, 46290 USA

Sams Teach Yourself Web Publishing with HTML 4 in 21 Days, Second Edition

Copyright © 2000 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-31725-7

Library of Congress Catalog Card Number: 99-63541

Printed in the United States of America

First Printing: December 1999

01 00 99 4 3 2 1

Trademarks

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

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability or responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

ACQUISITIONS EDITOR
Jeff Schultz

DEVELOPMENT EDITOR
Damon Jordon

MANAGING EDITOR
Charlotte Clapp

PROJECT EDITOR
George E. Nedeff

COPY EDITOR
Jill Bond

INDEXER
Christine Nelsen

PROOFREADER
Maryann Steinhart

TECHNICAL EDITOR
Will Kelly

INTERIOR DESIGNER
Gary Adair

COVER DESIGNER
Aren Howell

TEAM COORDINATOR
Amy Patton

COPY WRITER
Eric Borgert

PRODUCTION
Dan Harris
Mark Walchle

LIBRARY OF CONGRESS
FEB 25 1999

Sams Teach Yourself Web Publishing with HTML 4 in 21 Days, Second Edition

Copyright © 2000 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-31725-7

Library of Congress Catalog Card Number: 99-63541

Printed in the United States of America

First Printing: December 1999

01 00 99 4 3 2 1

Trademarks

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

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability or responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

ACQUISITIONS EDITOR

Jeff Schultz

DEVELOPMENT EDITOR

Damon Jordon

MANAGING EDITOR

Charlotte Clapp

PROJECT EDITOR

George E. Nedeff

COPY EDITOR

Jill Bond

INDEXER

Christine Nelsen

PROOFREADER

Maryann Steinhart

TECHNICAL EDITOR

Will Kelly

INTERIOR DESIGNER

Gary Adair

COVER DESIGNER

Aren Howell

TEAM COORDINATOR

Amy Patton

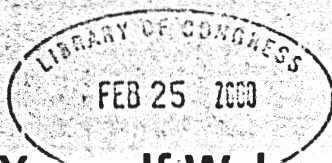
COPY WRITER

Eric Borgert

PRODUCTION

Dan Harris

Mark Walchle



Sams Teach Yourself Web Publishing with HTML 4 in 21 Days, Second Edition

Copyright © 2000 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-31725-7

Library of Congress Catalog Card Number: 99-63541

Printed in the United States of America

First Printing: December 1999

01 00 99 4 3 2 1

Trademarks

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

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability or responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

ACQUISITIONS EDITOR

Jeff Schultz

DEVELOPMENT EDITOR

Damon Jordan

MANAGING EDITOR

Charlotte Clapp

PROJECT EDITOR

George E. Nedeff

COPY EDITOR

Jill Bond

INDEXER

Christine Nelsen

PROOFREADER

Maryann Steinhart

TECHNICAL EDITOR

Will Kelly

INTERIOR DESIGNER

Gary Adair

COVER DESIGNER

Aren Howell

TEAM COORDINATOR

Amy Patton

COPY WRITER

Eric Borgert

PRODUCTION

Dan Harris
Mark Walchle

Overview

Introduction	1
PART 1 GETTING STARTED	7
Day 1 The World of the World Wide Web	9
2 Get Organized	29
3 An Introduction to HTML	51
PART 2 CREATING SIMPLE WEB PAGES	67
Day 4 Begin with the Basics	69
5 All About Links	93
6 More Text Formatting with HTML	127
PART 3 WEB GRAPHICS	169
Day 7 Using Images, Color, and Backgrounds	171
8 Creating Animated Graphics	207
9 Creating and Using Imagemaps	229
PART 4 DOING MORE WITH HTML	253
Day 10 Style Sheets	255
11 Tables	301
12 Frames and Linked Windows	351
PART 5 MULTIMEDIA, FORMS, AND DYNAMIC HTML	389
Day 13 Multimedia: Adding Sounds, Video, and More	391
14 Designing Forms	429
15 Using Dynamic HTML	471
PART 6 DESIGNING EFFECTIVE WEB PAGES	537
Day 16 Writing and Designing Web Pages: Dos and Don'ts	539
17 Examples of Good and Bad Web Design	575
18 Designing for the Real World	607

PART 7 GOING LIVE ON THE WEB	629
Day 19 Putting Your Site Online	631
20 Letting People Know It's There	649
21 Testing, Revising, and Maintaining Your Site	671
APPENDIXES	693
Appendix A Sources for Further Information	695
B HTML 4.0 Quick Reference	705
C Cascading Style Sheet (CSS) Quick Reference	745
D Colors By Name and Hexadecimal Value	781
E MIME Types and File Extensions	785
Index	789

Contents

INTRODUCTION	1
Who Should Read This Book	1
What This Book Contains.....	2
What You Need Before You Start	3
Conventions Used in This Book	3
Special Elements	4
HTML Input and Output Examples	4
Special Fonts	4
Teach Yourself Web Publishing with HTML: The Web Site	5
 PART 1 GETTING STARTED	 7
 DAY 1 THE WORLD OF THE WORLD WIDE WEB	 9
What Is the World Wide Web?	10
The Web Is a Hypertext Information System	10
The Web Is Graphical and Easy to Navigate	12
The Web Is Cross-Platform	13
The Web Is Distributed	14
The Web Is Dynamic	14
Web Browsers Can Access Many Forms of Internet Information	16
The Web Is Interactive.....	18
Web Browsers	19
What the Browser Does.....	20
An Overview of Two Popular Browsers	21
Web Servers	23
Uniform Resource Locators (URLs)	24
Summary	25
Workshop	25
Q&A	25
Quiz	26
Answers	26
Exercises	27
 DAY 2 GET ORGANIZED	 29
Anatomy of a Web Site.....	30
What Do You Want to Do on the Web?	32
Set Your Goals	34
Break Up Your Content into Main Topics	34

Ideas for Organization and Navigation.....	36
Hierarchies.....	36
Linear.....	39
Linear with Alternatives	40
Combination of Linear and Hierarchical.....	41
Web	43
Storyboarding Your Web Site	45
What Is Storyboarding and Why Do I Need It?	45
Hints for Storyboarding.....	46
Summary.....	47
Workshop	48
Q&A	48
Quiz	49
Answers	49
Exercises	49
DAY 3 AN INTRODUCTION TO HTML	51
What HTML Is—and What It Isn't	51
HTML Describes the Structure of a Page	52
HTML Does Not Describe Page Layout.....	53
Why It Works This Way	54
HTML Is a Markup Language	55
A Brief History of HTML Tags.....	55
Bridging the Future with XHTML 1.0.....	57
What HTML Files Look Like.....	57
A Note About Formatting	62
Programs to Help You Write HTML	62
Summary.....	64
Workshop	64
Q&A	64
Quiz	64
Answers	65
Exercises	65
PART 2 CREATING SIMPLE WEB PAGES	67
DAY 4 BEGIN WITH THE BASICS	69
Structuring Your HTML	69
The <html> Tag	70
The <head> Tag	71
The <body> Tag	71
The Title	72
Headings	73

Paragraphs	75
Lists, Lists, and More Lists	76
List Tags	77
Numbered Lists	78
Customizing Ordered Lists with HTML 3.2	79
Unordered Lists	82
Customizing Unordered Lists in HTML 3.2	83
Glossary Lists	84
¹ Nesting Lists	85
Comments	87
Summary	89
Workshop	90
Q&A	91
Quiz	92
Answers	92
Exercises	92
Day 5 All About Links	93
Creating Links	93
The Link Tag—<a>	94
Linking Local Pages Using Relative and Absolute Pathnames	99
Absolute Pathnames	100
Should You Use Relative or Absolute Pathnames?	101
Links to Other Documents on the Web	102
Linking to Specific Places Within Documents	107
Creating Links and Anchors	107
Linking to Anchors in the Same Document	113
Anatomy of a URL	114
Parts of URLs	114
Special Characters in URLs	115
Kinds of URLs	116
HTTP	116
Anonymous FTP	117
Non-Anonymous FTP	118
Mailto	118
Gopher	119
Usenet Newsgroups	119
File	120
Summary	121
Workshop	121
Q&A	122
Quiz	124
Answers	124
Exercises	125

DAY 6 MORE TEXT FORMATTING WITH HTML	127
Character Styles	128
Logical Styles	128
Physical Styles.....	131
Preformatted Text	133
Horizontal Rules	136
Attributes of the <hr> Tag	137
Line Break.....	140
Addresses	141
Quotations.....	142
Special Characters.....	142
Character Entities for Special Characters	143
Character Entities for Reserved Characters	145
Text Alignment	146
Aligning Individual Elements	146
Aligning Blocks of Elements	147
Fonts and Font Sizes	148
Changing the Font Size	148
Changing the Font Face.....	149
The Dreaded <blink>	149
<nobr> and 	150
Summary	162
Workshop	165
Q&A	165
Quiz	166
Answers	166
Exercises	167
 PART 3 WEB GRAPHICS	 169
DAY 7 USING IMAGES, COLOR, AND BACKGROUNDS	171
Images on the Web	172
Inline Images in HTML: The Tag	173
Adding an Image Alternative	174
Images and Text	177
Text and Image Alignment	179
Wrapping Text Next to Images	181
Adjusting the Space Around Images.....	184
Images and Links	185
Using External Images	189

Other Neat Tricks with Images	192
Image Dimensions and Scaling	192
More About Image Borders	193
Image Previews	194
Using Color	194
Naming Colors	195
Changing the Background Color	196
Changing Text Colors	196
Spot Color	197
Image Backgrounds	198
Hints for Better Use of Images	200
Do You Really Need This Image?	200
Keep Your Images Small	200
Reuse Images as Often as Possible	201
Provide Alternatives to Images	201
Summary	202
Workshop	203
Q&A	203
Quiz	204
Answers to Quiz	205
Exercises	205
DAY 8 CREATING ANIMATED GRAPHICS	207
What Is an Animation?	208
When and Where to Use Animation	209
Creating Transparent GIF Files	210
Choosing a Transparent Color	211
Programs to Help You Compile Animated GIFs	212
GIF Construction Set (Windows Platform)	212
GifBuilder (Macintosh Platform)	213
Economizing Animation Size	215
Tools to Help You Create Your Pictures	218
Useful Software Features	220
Summary	225
Workshop	225
Q&A	225
Quiz	226
Answers	226
Exercises	227

DAY 9 CREATING AND USING IMAGEMAPS	229
What Is an Imagemap?	229
Server-Side Imagemaps	231
Client-Side Imagemaps	231
Imagemaps and Text-Only Browsers	232
Creating Server-Side Imagemaps	232
Getting an Image	233
Determining Your Coordinates	234
Creating and Saving Your Map File	237
Installing the Server-Side Imagemap Program	238
Linking It All Together	239
Creating Client-Side Imagemaps	244
The <map> and <area> Tags	245
The usemap Attribute	246
Building Web Pages That Support Both Types of Imagemaps	249
Summary	249
Workshop	250
Q&A	250
Quiz	251
Answers	251
Exercises	252
 PART 4 DOING MORE WITH HTML	 253
DAY 10 XHTML AND STYLE SHEETS	255
What is XHTML and Why Use It?	256
The <DOCTYPE> Identifier	256
XHTML Syntax	257
The Concept of Style Sheets	258
Some Background on Style Sheets	260
The Bookworm Bookshop Revisited	261
Approaches to Style Sheets	264
External Style Sheets	265
Embedded Style Sheets	268
About Cascading	272
Commonly Used Style Sheet Properties and Values	273
Controlling Page Layout CSS Properties	273
Backgrounds, Colors, and Images	276
Setting Border Appearance	280
Font Appearance and Style	282
Text Alignment Properties in CSS	285
Inline Styles	288
Sources of Information About Cascading Style Sheets	297

Workshop	297
Q&A	298
Quiz	298
Answers	298
Exercises	299
DAY 11 TABLES	301
A Note About the Table Definition	302
Creating Tables	302
Table Parts	303
The <table> Element	303
Rows and Cells	304
Empty Cells	307
Captions	307
Sizing Tables, Borders, and Cells	311
Setting Table Widths	311
Changing Table Borders	312
Cell Padding	313
Cell Spacing	314
Column Widths	315
Setting Breaks in Text	316
Table and Cell Color and Alignment	318
Changing Table and Cell Background Colors	318
Changing Border Colors	320
Aligning Your Table Content	322
Table Alignment	322
Cell Alignment	323
Caption Alignment	325
Spanning Multiple Rows or Columns	326
More Advanced Table Enhancements	335
Grouping and Aligning Columns	335
Grouping and Aligning Rows	338
The frame and rules Attributes	341
Other Table Elements and Attributes	343
Summary	343
Workshop	348
Q&A	348
Quiz	349
Answers	349
Exercises	350

DAY 12 FRAMES AND LINKED WINDOWS	351
What Are Frames and Who Supports Them?	352
Working with Linked Windows	353
The <base> Tag	359
Working with Frames	360
The <frameset> Tag	361
The <frame> Tag	364
The <noframes> Tag	365
Changing Frame Borders	366
Creating Complex Framesets	368
Magic target Names	380
Floating Frames	381
Summary	384
Workshop	386
Q&A	386
Quiz	387
Answers	387
Exercises	388
 PART 5 MULTIMEDIA, FORMS, AND DYNAMIC HTML	 389
DAY 13 MULTIMEDIA: ADDING SOUNDS, VIDEOS, AND MORE	391
Understanding How to Present Sound and Video	392
The Old Standby: Linking	393
Embedding Sound and Video	402
Using the embed Element	403
Using the object Element	406
Combining embed and object	407
Embedding Flash Animations	407
Embedding Shockwave Animations	408
Embedding RealAudio and RealVideo	410
Multimedia Techniques Using Microsoft Internet Explorer	412
Sound and Video File Types	418
Of Plug-Ins and Players	420
Windows Media Player	420
Macromedia Flash	421
Macromedia Shockwave	421
QuickTime 4 by Apple	422
RealNetworks Grab Bag	422
Other Plug-Ins	423
Summary	424

Workshop	425
Q&A	425
Quiz	426
Answers	426
Exercises	427
DAY 14 DESIGNING FORMS	429
Understanding Form and Function	430
Essential Elements of a Form	436
Using the form Element to Create Forms	437
Creating Form Controls with the input Element	439
Using Other Control Elements	453
Creating Menus with select and option	455
Adding Extras	458
Displaying Control labels	458
Grouping Controls with fieldset and legend	459
Changing the Default Tabbed Navigation	460
Using Access Keys	460
Creating disabled and readonly Controls	461
Planning Your Forms	466
Summary	467
Workshop	468
Q&A	469
Quiz	470
Answers	470
Exercises	470
DAY 15 USING DYNAMIC HTML	471
What Exactly Is Dynamic HTML?	472
Learning JavaScript	475
Using Document Object Models	484
Handling Events	488
Coping with Reality: Cross-Browser DHTML Techniques	489
Sniffing for Browsers	490
Developing a Cross-Browser DOM Reference	492
Continuing Your DHTML Education	533
Summary	534
Workshop	534
Q&A	534
Quiz	535
Answers	536
Exercises	536

PART 6 DESIGNING EFFECTIVE WEB PAGES 537

DAY 16 WRITING AND DESIGNING WEB PAGES: DOS AND DON'TS 539

Using the HTML Extensions	540
Writing for Online Publication	543
Write Clearly and Be Brief	544
Organize Your Pages for Quick Scanning	544
Make Each Page Stand on Its Own	546
Be Careful with Emphasis	547
Don't Use Browser-Specific Terminology	548
Spell Check and Proofread Your Pages	549
Design and Page Layout	549
Use Headings as Headings	549
Group Related Information Visually	550
Use a Consistent Layout	551
Using Links	552
Use Link Menus with Descriptive Text	552
Use Links in Text	554
Avoid the Here Syndrome	555
To Link or Not to Link	557
Using Images	559
Don't Overuse Images	559
Use Alternatives to Images	561
Keep Images Small	561
Watch Out for Display Assumptions	563
Be Careful with Backgrounds and Link Colors	564
Other Good Habits and Hints	565
Link Back to Home	565
Don't Split Topics Across Pages	565
Don't Create Too Many or Too Few Pages	565
Sign Your Pages	567
Provide Non-Hypertext Versions of Hypertext Pages	569
Summary	569
Workshop	571
Q&A	571
Quiz	572
Answers	572
Exercises	573

DAY 17	EXAMPLES OF GOOD AND BAD WEB DESIGN	575
Example One: A Company Profile		576
What's New at Foozle?		577
Technical Information		579
The Company Overview		579
The Foozle Products Page		579
The Product Support Page		581
The Open Opportunities Page		582
Features of This Web Site and Issues for Development		583
Example Two: A Multimedia Encyclopedia		585
The Overview Page		585
The Norton Page		586
The Index of Marques		589
Features of This Web Site and Issues for Development		590
Example Three: A Shopping Catalog		591
Browsing the Catalog		593
Ordering		597
Features of This Web Site and Issues for Development		601
Summary		604
Workshop		604
Q&A		604
Quiz		605
Answers		605
Exercises		606
DAY 18	DESIGNING FOR THE REAL WORLD	607
What Is the Real World Anyway?		608
Considering User Experience Level		609
List Pages That Provide Descriptions of Your Site		610
Describe Your Navigation System		610
Add Browser Recommendations		614
List Other Necessary Files		614
Determining User Preference		615
Deciding on an HTML 4.0 or XHTML 1.0 Approach		616
HTML 4.0 and XHTML 1.0 Transitional		617
HTML 4.0 and XHTML 1.0 Framesets		620
HTML 4.0 and XHTML 1.0 Strict		623
Summary		626
Workshop		627
Q&A		627
Quiz		628
Answers		628
Exercises		628

PART 7 GOING LIVE ON THE WEB	629
DAY 19 PUTTING YOUR SITE ONLINE	631
What Does a Web Server Do?	632
Other Things Web Servers Do	632
Locating a Web Server	634
Using a Web Server Provided by Your School or Work	634
Using a Commercial Internet or Web Service	634
Setting Up Your Own Server	635
Organizing and Installing Your HTML Files	636
Questions to Ask Your Webmaster	636
Keeping Your Files Organized Using Directories	637
Having a Default Index File and Correct Filenames	638
Installing Your Files	639
Moving Files Between Systems	639
What's My URL?	641
Test, Test, and Test Again	642
Troubleshooting	643
Can't Access the Server	643
Can't Access Files	643
Can't Access Images	643
Links Don't Work	644
Files Are Displaying Incorrectly	644
Summary	645
Workshop	645
Q&A	645
Quiz	646
Answers	647
Exercises	647
DAY 20 LETTING PEOPLE KNOW IT'S THERE	649
Registering and Advertising Your Web Pages	650
World Wide Web Site Listings	650
Yahoo!	650
The World Wide Web Virtual Library	653
Yellow Pages Listings	655
Private Directories	655
Site Indexes and Search Engines	656
AltaVista	657
Excite	657
HotBot	657
Lycos	659

Infoseek	660
Submission Tools	661
Announce Your Site Using Usenet	663
Web Rings	664
Business Cards, Letterheads, Brochures, and Advertisements	666
Finding Out Who's Viewing Your Web Pages	666
Log Files	666
Access Counters	667
Summary	668
Workshop	668
Q&A	668
Quiz	669
Answers	669
Exercises	670
DAY 21 TESTING, REVISING, AND MAINTAINING YOUR WEB SITE	671
Integrity Testing	672
Validating Your HTML	672
Browser Testing	684
Verifying Your Links	685
Usability Testing	685
Examining Your Logs	686
Updating and Adding Pages to Your Site	687
Adding New Content	687
Revising Your Structure	688
Summary	689
Workshop	690
Q&A	690
Quiz	691
Answers	691
Exercises	692
PART 8 APPENDICES	693
APPENDIX A SOURCES FOR FURTHER INFORMATION	695
Access Counters	696
Browsers	697
Collections of HTML and Web Development Information	697
Forms and Imagemaps	698
HTML Editors and Converters	698
HTML Validators, Link Checkers, and Simple Spiders	699
Java, JavaScript, and Embedded Objects	699

Log File Parsers	699
Other	700
Servers and Server Administration	700
Sound and Video	700
Specifications for HTML, HTTP, and URLs	701
The Common Gateway Interface (CGI) and CGI Scripting.....	702
The Future of HTML and the Web.....	702
Tools and Information for Images	702
Web Providers	703
Web Indexes and Search Engines.....	703
APPENDIX B HTML 4.0 QUICK REFERENCES	705
Common Attributes and Events	706
%coreattrs	707
%i18n	707
%events	707
Structure	707
<bdo>...</bdo>.....	708
<body>...</body>.....	708
Comments <!-- ... -->	709
...	709
<!DOCTYPE...>.....	709
... through 	709
<head>...</head>	710
<hr>	710
<html>...</html>.....	711
<meta>	711
...	711
<title>...</title>	712
Text Phrases and Paragraphs.....	712
<acronym>...</acronym>	712
<address>...</address>	712
<blockquote>...</blockquote>.....	712
 	713
<cite>...</cite>.....	713
<code>...</code>.....	713
~~...~~.....	713
<dfn>...</dfn>.....	714
........	714
<ins>...</ins>.....	714
<kbd>...</kbd>.....	714
... 	714

<pre>...</pre>	715
<q>...</q>	715
<samp>...</samp>	715
...	715
_{...}	716
^{...}	716
<var>...</var>	716
Text Formatting Elements	716
...	716
<basefont>	716
<big>...</big>	717
...	717
<i>...</i>	717
<s>...</s>	717
...	718
<strike>...</strike>	718
<tt>...</tt>	718
<u>...</u>	718
Lists	718
<dd>...</dd>	718
<dir>...</dir>	719
<dl>...</dl>	719
<dt>...</dt>	719
... 	719
<menu>...</menu>	720
... 	720
... 	720
Links	721
<a>...	721
<base>	722
<link>	722
Tables	723
<caption>...</caption>	723
<col>	723
<colgroup>...</colgroup>	724
<table>...</table>	724
<tbody>...</tbody>	725
<td>...</td>	725
<tfoot>...</tfoot>	726
<th>...</th>	726
<thead>...</thead>	727
<tr>...</tr>	727

Frames	728
<frame>	728
<frameset>...</frameset>	729
<iframe>...</iframe>	729
<noframes>...</noframes>	729
Embedded Content	730
<applet>...</applet>	730
<area>	730
	731
<map>...</map>	732
<object>...</object>	732
<param>	733
Style	733
<style>...</style>	733
Forms	734
<button>...</button>	734
<fieldset>...</fieldset>	734
<form>...</form>	735
<input>	735
<isindex>	736
<label>...</label>	737
<legend>...</legend>	737
<option>...</option>	737
<select>...</select>	738
<textarea>...</textarea>	738
Scripts	739
<script>...</script>	739
<noscript>...</noscript>	739
Character Entities	740
APPENDIX C CASCADING STYLE SHEET (CSS) QUICK REFERENCE	745
How to Use This Appendix.....	746
Block-Level Properties	747
bottom, left, right, top	748
direction	748
display	748
float	749
position	749
unicode-bidi.....	749
z-index	749

Background and Color Properties.....	750
background	750
background-attachment	750
background-color	750
background-image	751
background-position	751
background-repeat	751
color	751
Box Model Properties.....	752
border	752
border-bottom, border-left, border-right, border-top.....	752
border-color.....	753
border-bottom-color, border-left-color,	
border-right-color, border-top-color	753
border-style.....	753
border-bottom-style, border-left-style,	
border-right-style, border-top-style	754
border-width.....	754
border-bottom-width, border-left-width,	
border-right-width, border-top-width.....	754
clear	755
height, width	755
margin	755
margin-bottom, margin-left, margin-right, margin-top.....	755
max-height, max-width	756
min-height, min-width	756
padding	756
padding-top, padding-right, padding-bottom, padding-left	757
Font Properties	757
font	757
font-family.....	758
font-size	758
font-size-adjust.....	758
font-stretch.....	759
font-style	759
font-variant.....	759
font-weight.....	759
List Properties.....	760
list-style	760
list-style-image.....	760
list-style-position	761
list-style-type	761

Text Properties	761
letter-spacing	761
line-height	762
text-align	762
text-decoration	762
text-indent	762
text-shadow	763
text-transform	763
vertical-align	763
white-space	763
word-spacing	764
Visual Effects Properties	764
clip	764
overflow	764
visibility	764
Aural Style Sheet Properties	765
azimuth	765
cue	765
cue-after, cue-before	765
elevation	766
pause	766
pause-after, pause-before	766
pitch	766
pitch-range	767
play-during	767
richness	767
speak	767
speak-header	768
speak-numeral	768
speak-punctuation	768
speech-rate	768
stress	769
voice-family	769
volume	769
Generated Content/Automatic Numbering Properties	770
content	770
counter-increment	770
counter-reset	770
marker-offset	771
quotes	771
Paged Media Properties	771
marks	771
orphans	771

page	772
page-break-after, page-break-before	772
page-break-inside	772
size	772
widows	773
Table Properties	773
border-collapse	773
border-spacing	773
caption-side	774
column-span, row-span	774
empty-cells	774
table-layout	774
User Interface Properties	775
cursor	775
outline	775
outline-color	775
outline-style	776
outline-width	776
Cascading Style Sheet Units	776
<absolute-size>	776
<angle>	776
<border-style>	777
<border-width>	777
<color>	777
<family-name>	778
<frequency>	778
<generic-family>	778
<generic-voice>	778
<integer>	779
<length>	779
<number>	779
<percentage>	779
<relative-size>	779
<shape>	780
<specific-voice>	780
<time>	780
<uri>	780
APPENDIX D COLORS BY NAME AND HEXADECIMAL VALUE	781
APPENDIX E MIME TYPES AND FILE EXTENSIONS	785
INDEX	789

About the Authors

Laura Lemay is a technical writer, author, Web addict, and motorcycle enthusiast. One of the world's most popular authors on Web development topics, she is the author of *Sams Teach Yourself Web Publishing with HTML*, *Sams Teach Yourself Java in 21 Days*, and *Sams Teach Yourself Perl in 21 Days*. You can visit her home page at <http://www.lne.com/lemay/>.

Denise Tyler is a freelance author, graphics artist, animator, and Web designer who resides in Madison, Wisconsin. She is the author of several FrontPage books in the *Laura Lemay's Web Workshop* series, the most recent being the best-selling *Laura Lemay's Web Workshop: Microsoft FrontPage 98*. She was also a contributing author for *Tricks of the Game Programming Gurus*, and author of *Fractal Design Painter 3.1 Unleashed*.

Deidre Hayes is an information architect with a Web services group and author of *Sams Teach Yourself HTML 4 in 10 Minutes*.

Bob Conell is a freelance author specializing in HTML and Web development.

Dedication

For Ed, a constant source of inspiration and support in my attempts to follow my dreams.
—Denise

Acknowledgments

To Sams Publishing for letting me write the kind of HTML book I wanted to see.

To the Coca-Cola Company, for creating Diet Coke and selling so much of it to me.

To all the folks on the `comp.infosystems.www` newsgroups, the `www-talk` mailing list, and the Web conference on the WELL, for answering questions and putting up with my late-night rants.

To innumerable people who helped me with the writing of this book, including Lance Norskog, Ken Tidwell, Steve Krause, Tony Barreca, CJ Silverio, Peter Harrison, Bill Whedon, Jim Graham, Jim Race, Mark Meadows, and many others I'm sure I've forgotten.

And finally, to Eric Murray, the other half of `lne.com`, for moral support when I was convinced I couldn't possibly finish writing any of this book on time, for setting up all my UNIX and networking equipment and keeping it running, and for writing a whole lot of Perl code on very short notice.

—Laura Lemay

The more I write, the more people I have to thank. With this book comes a whole new wave of talented folks at Macmillan with which I've had the pleasure to work. These people consistently demonstrate endless amounts of talent, dedication, and knowledge that never fails to impress me. Thanks, especially, to Mark Taber for his expert guidance in my efforts to update Laura's ever-popular and successful book. In spite of hard drive crashes, doctor visits, and a faulty call-waiting connection that knocked me offline as I downloaded files, he continued to provide an endless amount of positive feedback and *atta-girls*.

Thanks also to my family and friends for understanding that I have to disappear every now and then to focus on my work; and to caffeine for keeping my eyes open late at night while I scrounged through code examples and prose.

I also thank Laura Lemay for writing this great book in the first place. As always, her inspiration and talented writing style helps bring everything in focus and makes learning fun.

—Denise Tyler

Tell Us What You Think!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can fax, email, or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and authors as well as your name and phone or fax number. I will carefully review your comments and share them with the authors and editors who worked on the book.

Fax: 317-581-4770

Email: webdev_sams@mcp.com

Mail: Mark Taber
Associate Publisher
Sams Publishing
201 West 103rd Street
Indianapolis, IN 46290 USA

PART 5

DAY 14

Designing Forms

Up to this point, you have learned almost everything you need to know to create functional, attractive, and somewhat interactive Web pages. If you think about it, however, the pages you've created thus far have an information flow that is one-way. Your HTML documents, images, sounds, and video have been traveling, with no return ticket, to the people viewing them in their Web browsers.

Today's lesson is about creating HTML forms in order to collect information from people visiting your Web site. Forms enable you to gather just about any kind of information for immediate processing by a server-side script or for later analysis using other applications. If you've spent much time browsing the Web, you undoubtedly have run across forms of various flavors. Many forms exist: simple forms that perform searches, forms that log you in to Web sites, forms that enable you to order products online, and so on. They all share one thing in common: accepting input from a Web page visitor.

If you're one to worry about compatibility, you can set your mind at ease. HTML forms have been around virtually since the beginning of the HTML language, and are widely supported by all leading Web browsers. I'll make sure to point out any possible compatibility problems along the way.

Don't be intimidated by forms! While they may look complex, they actually are very easy to code. The hardest part is formatting them. In today's lesson, the following topics are covered, which will enable you to create any type of form possible with HTML:

- How HTML forms interact with server-side scripts to provide interactivity
- Creating simple forms to get the hang of it.
- Learning all the types of form controls you can use to create radio buttons, check boxes, and more
- Using more advanced form controls to amaze your friends and coworkers
- Form planning so that your data matches any server-side scripts you use

Understanding Form and Function

Right off the bat, you need to understand a few things about forms. First, a form is part of a Web page that you create using HTML elements. Each form contains a **form element** that has special controls, such as buttons, text fields, check boxes, submit buttons, and menus. These controls make up the user interface for the form (that is, the pieces of the form users see on the Web page). When people fill out forms, they are interacting with the controls of the forms. In addition, you can use many other HTML elements within forms to create legends, provide additional information, add structure, and so on. These elements are not part of the form itself, but can enhance your form's look or improve its usability.



An *HTML form* allows you to gather information from a person visiting your Web page. You can take the information you collect and send it to a script residing on your Web server to manipulate or store it.

When a person fills out an HTML form, he or she enters information or makes choices using the form controls. The final step is submitting it, at which time several things happen. First, the form identifies the controls within the form that contain data and builds a form data set to contain it. Next, the data set is encoded and sent to the Web server to be processed.

It is very important that you understand the implications of this final step. The data, after all, is what you are after! This is the reason you've chosen to create a form in the first place. After a user clicks the Submit button, the process ceases to be one of pure HTML and becomes reliant upon scripts (called *Common Gateway Interface*, or *CGI* scripts, most often written in a scripting language called Perl) that are resident on the Web server. In other words, for your form to be successful, you must already have a script on the server that will take the data and store it or manipulate it in some manner.

There are a few important exceptions to this rule. First, a form may redirect a person to another Web page, based on his or her input, or the form sends an email containing the form data to an email address. The second exception is useful and is an easy way for you to test forms. Rather than using a script, you can simply instruct the form to email the contents to you. One final exception is that forms sometimes are used in Dynamic HTML because they trap user events, such as clicking the mouse. A form in a Dynamic Web page may not be used to collect data, but could be used to create buttons from which the user can choose to perform some action.

Exercise 14.1: Creating a Simple Form that Accepts a Name and Password

▼ To Do

Okay, let's get right to it and create a simple form that illustrates the concepts just presented.

I've created a Web page that prompts the user to enter a name and a password to continue. Take a peek down at Figure 14.1 to see the Web page.

Start by opening up your favorite HTML editor (mine is Notepad) and creating a Web page template. Begin by entering the standard HTML header information, include the body element, and then close the body and html elements to form a template from which to work. If you already have a template similar to this, just load it into your HTML editor.

Note

I tend to use Transitional HTML and note that in the `<!doctype>` declaration. This gives me the flexibility of adding deprecated HTML elements if I choose, without having to worry about validation errors.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>myTitle</title>
</head>
<body>

</body>
</html>
```

Next, add your title so that people will understand the purpose of the Web page.

▼ `<title>Enter the Wabbit Hole</title>`

- ▼ Within the body of the Web page, add a `form` element. I have added both the opening and closing tags, with an empty line in between so that I don't forget to close the `form` when I'm finished.

```
<form action="http://www.fakeurl.com/cgi-bin/entrance.cgi" method="post">

</form>
```

Before continuing, you need to know more about the `form` element and the attributes you see within the opening tag. Obviously, `<form>` begins the element and indicates that you are creating an HTML form. The `action` attribute, which is required, specifies the URL to the server-side script (including the file name) that will process the form when it is submitted. It is very important that the script with the name you have entered is present on your Web server at the location the URL specifies.

Note

Prior to "going live" with forms, you should contact your ISP and ask them whether you can use the scripts they have or add your own. You also need to determine the URL that points to the directory in which the scripts reside on the server. Some ISPs rigidly control scripts for security purposes and will not allow you to create or add scripts to the server. If that is the case, and you really need to implement forms on your Web pages, you should consider searching for a new ISP.

The next attribute is `method`, which can accept one of two possible values: `post` or `get`. These values define how form data is submitted to your Web server. The `post` method includes the form data in the body of the form and sends it to the Web server. The `get` method appends the data to the URL specified in the `action` attribute and most often is used in searches.

Now add some form controls and information to make it easy for a visitor to understand how to fill out the form. Within the `form` element, begin by adding a helpful description of the data to be entered by the user, and then add a text form control. This prompts them to enter their name in a text-entry field. Don't worry about positioning just yet because later you will put all the form controls into a table.

```
<form action="http://www.fakeurl.com/cgi-bin/entrance.cgi" method="post">
Your wascaw name:<input type="text" name="username" />
</form>
```

Next, add another bit of helpful text and a password control.

```
<form action="http://www.fakeurl.com/cgi-bin/entrance.cgi" method="post">
Your wascaw name:<input type="text" name="username" />
Your just as wascaw password:<input type="password" name="userpassword" />
</form>
```

- ▼ Notice that both of these form controls are part of the `input` element. The `type` attribute defines which type of control will be created. In this case, you have a text control and a password control. Each type of control looks differently when it appears in a Web browser (such as a text-entry field, button, or check box), accepts a different type of user input (such as entering text or the click of a button), and is suitable for different purposes. Each control also receives a name that distinguishes it and its data from the other form controls. You can use whatever name you like, as long as it is unique. (Normally control names must all be different; however, exceptions to this are explained later in this lesson).

Finally, add a submit button so that the user can send the information he or she entered into the form. Add a reset button that clears the form in case the user made a mistake or wants to start over.

```
<form action="http://www.fakeurl.com/cgi-bin/entrance.cgi" method="post">
Your wascaw name:<input type="text" name="username" />
Your just as wascaw password:<input type="password" name="userpassword" />
<input type="submit" value="Enter" />
<input type="reset" value="Oops!" />
</form>
```

The submit and reset buttons are another type of input element. Notice that I've included the `value` attribute with each. In this case, the `value` attribute modifies the text that is shown in the button and displayed in the Web browser. You can choose not to use the `value` attribute with these buttons. If so, the Web browser will display default text for these two types of buttons.



When naming form controls and labeling buttons, you should strive for clarity and meaning. If a form is frustrating or hard to figure out, the person you want to provide the information most likely will leave your site for greener pastures!

At this point, you have created the form and it is ready to rumble; however, if you load it into your Web browser, you'll see that it doesn't look all that appealing. It's time to add a few graphical elements to make the page look better, and arrange the elements of the form into a nicely aligned table.

I've created three graphics to place at the top of the page and used a table to arrange them.

```
<table border="0">
  <tr>
    <td></td>
```

```

▼      <td align="center"><br>
      </td>
      </tr>
    </table>

```

Next, create another table and that contains four rows and two columns. You're going to put your form in the table to position all the elements.

```

<table border="0">
<tr>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
</tr>
</table>

```

Now, add the form elements by inserting the opening form tag after the opening table tag. Insert the form controls into the rows and columns of the table, as I have shown, making sure not to forget the closing form tag.

Notice that I've left the third row of the table blank—that's because I'm cheating! After I completed the source code for this example, I wanted a bit more space between the password field and the buttons. After a bit of experimentation, I also aligned the text prompts to the right, set each column to take up 50 percent of the total width of the table, and centered the buttons in their respective columns.

```

<table border="0">
  <form action="http://www.fakeurl.com/cgi-bin/entrance.cgi" method="post">
    <tr>
      <td align="right" width="50%">Your wascawy name:</td>
      <td width="50%"><input type="text" name="username" /></td>
    </tr>
    <tr>
      <td align="right">Your just as wascawy password:</td>
      <td><input type="password" name="userpassword" /></td>
    </tr>
    <tr>
      <td></td>
      <td></td>
    </tr>
  </form>
</table>
▼

```

```

    <tr>
      <td align="center"><input type="submit" value="Enter" /></td>
      <td align="center"><input type="reset" value="Oops!" /></td>
    </tr>
  </form>
</table>

```

This looks like it stands a chance of working, so now put all the source code together. A few final notes are warranted. First, I wrapped everything up in a div element, and set the align attribute to center in order to center everything on the page. Note on the page where the opening and closing div tags are located. To add some extra space between the graphics and the form, I also included a line break using the br element.

INPUT

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Enter the Wabbit Hole</title>
</head>
<body>
<div align="center">
<table border="0">
  <tr>
    <td></td>
    <td align="center"><br>
    </td>
  </tr>
</table>
<br>
<table border="0">
  <form action="http://www.fakeurl.com/cgi-bin/entrance.cgi"
method="post">
    <tr>
      <td align="right" width="50%">Your wascaw name:</td>
      <td width="50%"><input type="text" name="username" /></td>
    </tr>
    <tr>
      <td align="right">Your just as wascaw password:</td>
      <td><input type="password" name="userpassword" /></td>
    </tr>
    <tr>
      <td></td> <!-- Comment: left blank to add space -->
      <td></td> <!-- Comment: left blank to add space -->
    </tr>
    <tr>
      <td align="center"><input type="submit" value="Enter" /></td>
      <td align="center"><input type="reset" value="Oops!" /></td>
    </tr>
  </form>

```

```

</table>
</div>
</body>
</html>

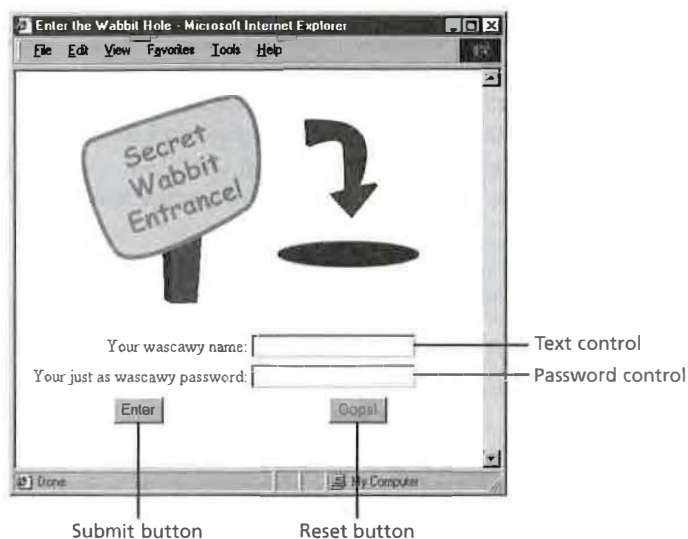
```

That took a little work, but I think the final product shown in Figure 14.1 looks good.

OUTPUT

FIGURE 14.1

A simple HTML form with four form controls provides access to the Secret Wabbit Hole.



To complete the exercise, I tested the form to see whether it returned the data I wanted. Following is what the data set looks like:

```
username=Elmer&userpassword=hunter
```

It is pretty measly, but you can see that each form element name contains data that I entered into the form and is nicely tied to that data. Using a script, you will find it easy to parse this output, manipulate it, and achieve your overall purpose.

Essential Elements of a Form

After completing the first exercise, you should have picked up a few things: how to use the form element, including simple form controls, and using the submit and reset buttons. This section fully explains and expands upon them. The discussion is divided into two major areas: the form element and the controls created by the input element. After this discussion, you'll walk through a more complicated exercise.

Using the `form` Element to Create Forms

Because the `form` element is the basis for all forms created in Web pages, it is important that you understand how to use it.

First, forms are considered *block-level elements*. To understand this, you need to know about the other basic HTML content model: *inline elements*. You apply inline elements, such as `<b>` and `<u>`, to text or individual characters to modify their appearance. As such, they contain the text they modify, such as `<b>I Love this game!</b>`, and can contain other nested inline elements. `<b>I <i>Love</i> this game!</b>` is perfectly valid HTML.

Block-level elements are larger organizational structures (such as the `form` and `table` elements). They can include contain Web page content, inline, as well as other block-level HTML elements. If you display the following HTML in a Web browser, the existence of the nested inline element does not start a new line.

```
<b>Nesting <i>inline</i> elements does not cause a line-break.</b>
```

However, block-level elements such as `form` start a new line in the Web browser. The upshot of all of this is that if you are concerned about the layout of your page, the opening tag of the `form` element will cause a new line to be created. Consider the following code fragment:



Inline elements such as `<b>...</b>` can contain text and other inline elements.

Block-level elements such as `<div>...</div>` can contain content, other block-level elements, and inline elements. Most block-level elements are displayed on a new line while inline elements do not start a new line.

```
<p>Oatmeal <form action="http://www.fakeurl.com/cgi-bin/register.cgi"
method="post">
lovers wishing to view our private stock of recipes need to enter your name
and password.</p>
```

Placing the `form` tag in its current location causes a new line to be created after `Oatmeal`, resulting in the inappropriate visual effect shown in Figure 14.2.

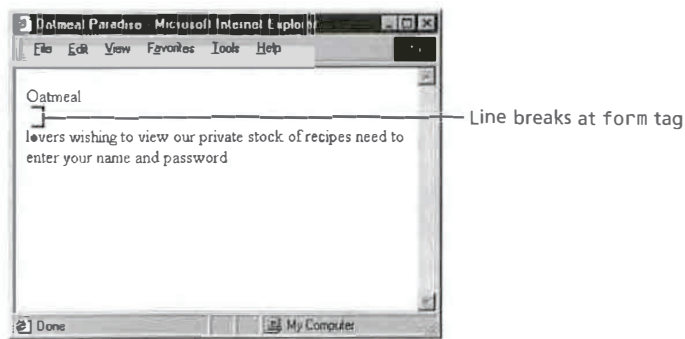
To sum up thus far, the `form` element causes a new line to be created and the `form` can contain content, inline, and other block-level elements. The one exception to this is that you cannot nest forms—only one form at a time, please!

The syntax for creating the `form` element is fairly straightforward. You most often will use only two attributes—`action` and `method`—of which `action` is mandatory, as in the following code snippet.

```
<form action="someaction" method="get or post">
content, form controls, and other HTML elements
</form>
```

FIGURE 14.2

Placing a form element within an inline element, such as the paragraph <p>, causes a line break.



action defines the action taken when the form is submitted, and generally contains a URL to a CGI script on the Web server. Here again, remember that the script must be in the exact location you specify and working properly for the form to successfully be submitted. Although most forms send their data to scripts, you also can make the action link to another Web page or a `mailto` link. The latter is formed as follows:

```
<form action="mailto:somebody@isp.com" method="post">
```

This attaches the form data set to an email, which then is sent to the email address listed in the action attribute.

Tip

To test your forms, I recommend using `mailto` to send yourself the results of your test form. Doing this enables you to debug your form and ensures that you are getting the correct data prior to linking up with a script.

The `method` attribute can be one of only two possible values: `get` or `post`. These values define how the form data provided by the action attribute will be sent to the script. Forms that use `get` append the form data to the action URL with a question mark, and then send the entire kit-and-caboodle to the modified URL. Sites with search engines normally use this method. If I had a text control named `searchstring` and had entered `Orangutans` in the field, the resulting URL might look the following:

```
http://www.fakeurl.com/cgi-bin/search?searchstring=Orangutans
```

Although the `method` attribute is not required, the default value of `get` will be used if you omit it.

The other possible value is `post`. Rather than appending the form data to a URL and sending the combined URL-data string to the server, `post` simply sends the form data to the location specified by the action attribute.

That about does it for the `form` element, but you've really only just begun. Did you notice that the `form` element, by itself, doesn't do much more than indicate which type of action to take when the form is submitted? To actually gather information, you're going to need items called form controls.

Creating Form Controls with the `input` Element

Now it's time to learn how to create the elements that gather information within a form. The `input` element enables you to create many different types of controls.



Form controls are special HTML elements used in a `form` that allow you to gather information from a person viewing your Web page. The information is collected into a data set that is sent to the location of the `action` attribute when the form is submitted.

The `input` element consists of an opening tag with attributes, no other content, and no closing tag.

```
<input attributes />
```

The key point here is choosing the right attributes that will create the type of form control you need. The most important of these is `type`, which defines the control. For all controls, except submit and reset buttons, the `name` attribute is required. The `name` attribute names the control so that data can be assigned to the form control name when sent to the server. The rest of this section describes the different types of controls you can create using the `input` element.

Creating Text Controls

Text controls enable you to gather information from a user in small quantities. This creates a single-line text input field in which users can type information, such as their name or a search term. Unlike other controls, such as radio buttons (that have only two states—on and off) or menu items (that are predefined), text controls give visitors complete freedom to enter any text they desire.

Text controls always are part of an `input` element, which is included in the `form` element. Begin by creating an `input` element and choose `text` as the value for the `type` attribute. Make sure to name your control so that the server script will be able to process the value. Figure 14.3 shows a text control that offers helpful text that describes what you want the user to enter in the control.

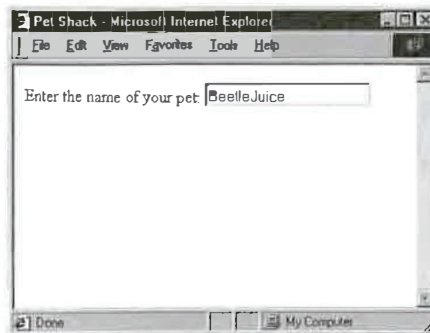
```
<p>Enter the name of your pet: <input type="text" name="petname" /></p>
```

You can modify the appearance of text controls by using the `size` attribute. Entering a number sets the width of the text control in characters, as in the following:

```
<input type="text" name="petname" size="15" />
```

FIGURE 14.3

Text controls create a single-line text entry field.



To limit the number of characters a user can enter, add the `maxLength` attribute to the text control. This sets the maximum number of characters the user can enter. If users attempt to enter more text, their Web browser will stop accepting input for that particular control.

```
<input type="text" name="petname" size="15" maxLength="15" />
```

To display text in the text control prior to the user entering information, use the `value` attribute. The `value` attribute can act as a prompt or reminder to the user to enter data, as in the following:

```
<input type="text" name="petname" size="15" maxLength="15" value="Enter Pet Name" />
```

In this case, `Enter Pet Name` appears in the control when the form is drawn in the Web browser. It remains there until the user modifies it.

Tip

When using the `value` attribute, using a value that is larger than the size of the text control can confuse the user because the text will appear to be cut off. Try to use only enough information to make your point. Ensure that any value is less than or equal to the number of characters you specified in `maxLength`.

To make text controls read-only, include the `readonly` attribute, as in the following:

```
<input type="text" name="year" value="1999" readonly />
```

This prevents users from entering text in the text control. It also is useful to establish a constant default for a text control when the form is submitted.

Creating Password Controls

The password control is very similar to the text control in that users enter only a small amount of text in the form. This, however, is a special case because the password control uses asterisks to mask the user's input.

Tip

You don't have to limit your use of the password control to just passwords. You can use it for any sensitive material that you feel needs to be hidden when the user enters the information into the form.

To create a password control, create an input element and assign password to the type attribute. To limit the size of the password control and the maximum number of characters a user can enter, you can use the length and maxlength attributes, as in the following code:

```
<p>Enter your password: <input type="password" name="userpassword"
size="8" maxlength="8" /></p>
```

Figure 14.4 shows a password control with some helpful text that describes to the user the purpose of the control.

FIGURE 14.4

Create masked text input fields with password controls.



Note that you do not need to include the value attribute because the value you enter is masked when drawn into the user's Web browser.

Caution

Any information sent by a password control is not masked or encrypted in any way; therefore, this is not a secure means of transmitting sensitive information. Although the users can't read what they are typing, the password control has no other inherent security measures.

Creating Submit Buttons

You use the submit button to transmit form data for processing by a server-side script or other program. Entering submit as the type of control creates a submit button that contains the default caption of Submit Query. To change the text shown in the button, use the value attribute and enter your own description, as in the following.

```
<input type="submit" value="Send Form Data" />
```

Note

You can have more than one submit button per form.

Creating Reset Buttons

You use reset buttons to clear the contents of the form and set all the form controls to their initial default values. In most cases, this is the value you specify using the value attribute when you create each control. As with the submit button, you can change the default caption of reset to one of your own choosing by using the value attribute, as in the following:

```
<input type="reset" value="Clear Form" />
```

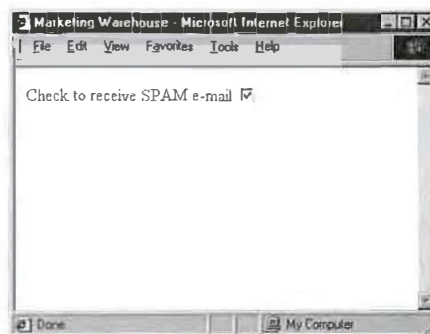
Creating Check Box Controls

Check boxes enable you to create a control that can be toggled between checked and unchecked, reflecting an on or off state (see Figure 14.5). To create a check box, use the input element and enter checkbox as the type of control. Be sure to include the name attribute so that the script will know to which control the data is assigned, as in the following:

```
<p>Check to receive SPAM e-mail <input type="checkbox" name="spam" /></p>
```

FIGURE 14.5

Check box controls create boxes users check.



To initially display the check box as checked, include the checked attribute, as in the following:

```
<input type="checkbox" name="year" checked />
```

You can group check boxes together and assign the same control name. This allows multiple values to be chosen and applied to one property.

```
<p>Check all symptoms that you are experiencing:<BR />
Nausea <input type="checkbox" name="symptoms" value="nausea" /><br />
Light-headedness <input type="checkbox" name="symptoms" value="lightheadedness" /><br />
Fever <input type="checkbox" name="symptoms" value="fever" /><br />
Headache <input type="checkbox" name="symptoms" value="headache" /><br />
</p>
```

When this form is submitted to a script for processing, each check box that is checked returns a value associated with the name of the check box.

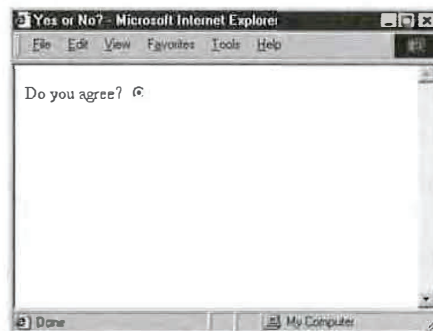
Creating Radio Buttons

Radio buttons are almost identical to check boxes, but appear differently in the Web browser. Rather than a box that can be checked or unchecked, a small circular (sometimes diamond shaped) button is drawn and filled when the user selects it (see Figure 14.6). You create radio buttons in an input element by entering radio as the type of control. The value attribute, which is mandatory, assigns data to the radio button. When the form is submitted, the value is sent to the server if the radio button is selected.

```
<p>Do you agree? <input type="radio" name="agree" value="Yes" /></p>
```

FIGURE 14.6

Radio buttons are useful for yes or no questions or choosing one item from a list.



When two or more radio buttons share the same control name, they form a mutually exclusive data set. In other words, only one radio button can be selected. In the following code snippet, both radio buttons have the name of sex, which means that the user can

select only one. If a user selected Male, realized it was a mistake, and then selected Female, the Male radio button automatically would then be unselected.

```
<input type="radio" name="sex" value="Male" />
<input type="radio" name="sex" value="Female" />
```

Creating Graphical Buttons

Using image as the type of input control creates a submit button decorated with the image you choose using the src attribute, as in the following:

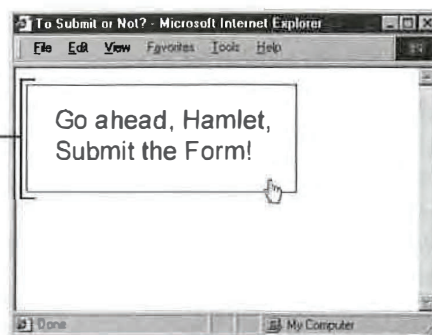
```
<input type="image" src="submit.gif" name="submitformbtn" />
```

Figure 14.7 shows a custom button created with an image.

FIGURE 14.7

Create custom buttons with the type="image".

Custom button



In addition to submitting the form data, the x and y coordinates of the point the user clicked are transmitted to the server. The data is submitted as `name.x = x coord` and `name.y = y coord`, where `name` is the name of the control. Using the preceding code, a result might look like the following:

```
submitformbtn.x=150&submitformbtn.y=200
```

You can omit the name, if you choose. In doing so, the coordinates returned would just be `x =` and `y =`.

Creating Push Buttons

In addition to creating submit, reset, and image buttons, you also can create push buttons that cause client-side scripts to execute when they are enabled. Figure 14.8 shows a button that has a client-side script as the action to take when it is pressed. Use the following code to create a push button:

```
<input type="button" name="verify" value="verify" onclick="verifydata()" />
```

FIGURE 14.8

Push buttons can be tied to client-side scripts.



In this example, a button that runs a script function called `verifydata` when it is clicked is created. You indicate the name that appears in the button with the `value` attribute of `Verify Data`.

Buttons created using this method are more limited than those created by using the `button` element. Don't confuse the two!

Hiding Controls with `hidden`

Hidden controls are special controls that are hidden from and cannot be modified by the user. When the user submits the form, the value assigned to the hidden control is sent along with the other data.

```
<input type="hidden" name="surveynumber" value="1402" />
```

I have seen this technique used most often with surveys where a way to identify the data is required. In this case, I have created a hidden form control named `surveynumber` with a value of `1402`. When the form is submitted for processing, I can identify the form based on this information. When I change the survey tomorrow, I can change the value to a different number.

Upload Files with the `file` Select Control

The file select control enables you to enable a person to upload a file along with the form. Figure 14.9 shows a file select control.

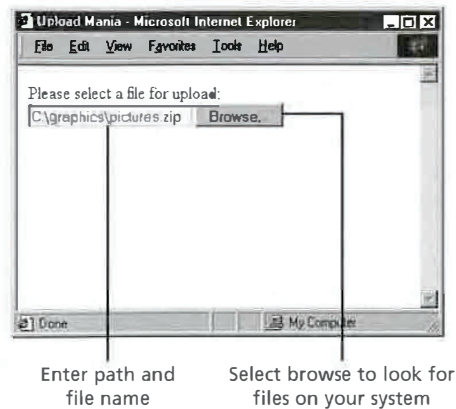
```
<p>Please select a file for upload: <input type="file" name="fileupload" /></p>
```

Users can enter a file name (with the path) or choose to browse the contents of their hard drive in order to identify a file to upload. If you use the `value` attribute, you theoretically can specify an initial file name to appear; however, this does not seem to be supported by Microsoft Internet Explorer 5 or Netscape Navigator 4.5.

```
<input type="file" name="fileupload" value="myresume.doc" />
```

FIGURE 14.9

Uploading files is easy with the file select control.



Exercise 14.2: Creating a Form with Several Types of input Controls

▼ To Do

It's time for another exercise. For this exercise, you'll work with all the types of controls presented in this section.

To start, open the template you created in Exercise 14.1, and create three form templates. You will use the first form for a survey, the second to gain access to a member's area of a fictitious Mime Academy, and the third to search the site. Refer to the following code:

```
<form action="http://www.fakeurl.com/cgi-bin/survey.cgi method="post"
</form>
<form action="http://www.fakeurl.com/cgi-bin/members.cgi method="post"
</form>
<form action="http://www.fakeurl.com/cgi-bin/search.cgi method="get"
</form>
```

Notice that you will be using different scripts for each form, and that the third uses the get method.

Now it's time to add form controls to each form. For the survey, you will want to ask a few simple multiple-choice questions. For the first question, use check boxes and assign the same name to them all. Place the text prompt before the input element so that it shows up in the browser window to the left of the check box. The following code shows you how:

```
<form action="http://www.fakeurl.com/cgi-bin/survey.cgi method="post"
Take today's survey
I like miming because: (check all that apply)
▼ It's fun<input type="checkbox" name="whyilikeit" value="fun" />
```

▼ I meet interesting people☐
 />
 It's who I am☐
 The money☐

The second question for this form is another multiple-choice question, but you only want to accept one answer. For this, use radio buttons that have the same assigned name. Rather than placing the text prompts before the button, place the text prompts after so that they will show up to the right of the buttons. Doing this ensures that the buttons are lined up.

The most difficult thing about being a mime is:
☐
☐
☐

Next, enter a file select element that enables users to select and upload a file when they submit the survey. Add submit and reset buttons. Finally, in order to track the survey, add a hidden control that is hard-coded to return the value of 061300. By using this hidden control, you can track all survey responses by date and change the control every day. After you add the hidden control, enter the closing form tag.

Upload a picture of yourself miming!

 </form>

The second and third forms are much easier. The second form involves adding text and password controls for a person's user name and password, and finishes with a submit button. Limit the maximum number of characters that can be entered to 20 and 10 respectively, and then limit the password field 10 characters, as in the following:

```
<form action="http://www.fakeurl.com/cgi-bin/members.cgi method="post">
Enter our Member's Only area!
Username:<input type="text" name="username" maxlength="20" />
Password:<input type="password" name="userpassword" size="10" maxlength="10" />
<input type="submit" value="Submit" />
</form>
```

The final form is simply a text control and an image button. The image button acts as a submit button. It contains a graphic that I've created in place of the normal button face.

▼ <form action="http://www.fakeurl.com/cgi-bin/search.cgi method="get">
 Search our database<input type="text" name="searchstring" />

 </form>

- ▼ The forms, form controls, and other information are complete; however, in order for them to look good in an HTML document, you will need to format them.

Note

I have found that formatting my forms after I create them enables me to concentrate on each aspect separately, which produces better results.

The visual appearance of the page divides nicely into five elements: three forms, a title graphic that will appear at the top of the Web page, and another graphic to add interest.

To organize your items on the page, first center the title graphic at the top of the page by using the following code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Anne's Mime Academy</title>
</head>
<body>
<div align="center">
  
</div>
```

Moving on to the graphic and forms, notice that two forms are rather short and one is long. Think of the remainder of the page as having two columns in which you can place the two smaller forms, including seating the “interest” graphic on one side and the longer form on the other. How are you going to do this? Tables!

Start by creating a master table that contains two columns and three rows. In the following code, I’ve commented the role that each row and column of the table will play:

```
<table border="0">
<tr>
  <td></td> <!-- Row 1, Column 1: for the interest graphic -->
  <td></td> <!-- Row 1, Column 2: use for nested table holding the first form
-->
</tr>
<tr>
  <td></td> <!-- Row 2, Column 1: for another nested table and the second form
-->
  <td></td> <!-- Row 2, Column 2: first form continues -->
</tr>
<tr>
  <td></td> <!-- Row 3, Column 1: for the third nested table and third form --
>
  <td></td> <!-- Row 3, Column 2: first form continues -->
</tr>
</table>
```



- ▼ Now use the following code to add the nested tables:

```
<table border="0">
<tr>
  <td></td> <!-- Row 1, Column 1: for the interest graphic -->
  <td><!-- Row 1, Column 2: use for nested table holding the first form -->
    <table border="0">
      <tr>
        <td></td>
        <td></td>
      </tr>
      <!-- add 12 more table rows here like the previous one -->
    </table>
  </td>
</tr>
<tr>
  <td><!-- Row 2, Column 1: use for nested table holding the second form -->
    <table border="0">
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
    </table>
  </td>
  <td></td> <!-- Row 2, Column 2: first form continues -->
</tr>
<tr>
  <td><!-- Row 3, Column 1: for the third nested table and third form -->
    <table border="0">
      <tr>
        <td></td>
        <td></td>
      </tr>
      <tr>
        <td></td>
        <td></td>
      </tr>
    </table>
  </td>
</tr>
</table>
```

- ▼ Now add all the graphic and form elements to their appropriate locations in the tables and put the entire thing together.

INPUT

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Anne's Mime Academy</title>
</head>
<body>
<div align="center">
  
</div>
<table border="0">
<tr>
  <td align="center"></td>
  <td rowspan="3">
    <table border="0" bgcolor="#ffcc33">
      <form action="http://www.fakeurl.com/cgi-bin/survey.cgi"
method="post">
        <tr>
          <td align="center" colspan="2">Take today's survey</td>
        </tr>
        <tr>
          <td colspan="2">I like miming because: (check all that
apply)</td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td>It's fun<input type="checkbox" name="whyilikeit"
value="fun" /></td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td>I meet interesting people<input type="checkbox"
name="whyilikeit" value="people" /></td>
        </tr>
        <tr>
          <td>&nbsp;</td>
          <td>It's who I am<input type="checkbox" name="whyilikeit"
value="personality" /></td>
        </tr>
```

```

        <tr>
            <td>&nbsp;</td>
            <td>The money<input type="checkbox" name="whyilikeit"
value="money" /></td>
        </tr>
        <tr>
            <td colspan="2">The most difficult thing about being a mime
is:</td>
            <td></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="radio" name="difficult" value="makeup" />
Putting on the makeup</td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="radio" name="difficult" value="unitard" />
Wearing a spandex unitard</td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input type="radio" name="difficult" value="nottalking"
/>Not talking</td>
        </tr>
        <tr>
            <td colspan="2">Upload a picture of yourself miming!</td>
            <td></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="file" name="picture" /></td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input type="submit"
value="submit" />
            <input type="reset" value="reset" /></td>
        </tr>
        <input type="hidden" name="surveynumber" value="091899" />
    </form>
</table>
</td>
</tr>

```

```

<tr>
  <td valign="top">
    <table border="0" bgcolor="#ffff99" width="100%">
      <form action="http://www.fakeurl.com/cgi-bin/members.cgi"
method="post">
        <tr>
          <td align="center" colspan="2">Enter our Member's Only
area!</td>
          <td></td>
        </tr>
        <tr>
          <td>Username:</td>
          <td><input type="text" name="username" maxlength="20" /></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input type="password" name="userpassword" size="10"
maxlength="10" /></td>
        </tr>
        <tr>
          <td align="center" colspan="2"><input type="submit"
value="submit" /></td>
          <td></td>
        </tr>
      </form>
    </table>
  </td>
</tr>
<tr>
  <td>
    <table border="0" width="100%" bgcolor="#ffcccc">
      <form action="http://www.fakeurl.com/cgi-bin/search.cgi"
method="get">
        <tr>
          <td>Search our database</td>
          <td><input type="text" name="searchstring" /></td>
        </tr>
        <tr>
          <td align="center" colspan="2"><input type="image"
src="searchbtn.gif" name="searchbtn" /></td>
          <td></td>
        </tr>
      </form>
    </table>
  </td>
</tr>
</table>
</body>
</html>

```

▼ Figure 14.10 shows the completed Web page, complete with multiple forms and controls.

OUTPUT**FIGURE 14.10**

Anne's Mime Academy has three forms on one page.

The screenshot shows a web browser window titled "Anne's Mime Academy - Microsoft Internet Explorer". The page content includes a "Welcome to Anne's Mime Academy" header with a cartoon character. Below the header, there are three forms:

- Form 1:** A survey titled "Take today's survey". It includes a heading "I like miming because: (check all that apply)" and several checkboxes: "It's fun", "I meet interesting people", "It's who I am", "The money", and "The most difficult thing about being a mime is:". Below these are radio buttons for "Putting on the makeup", "Wearing a spandex unitard", and "Not talking". At the bottom of the survey is a section "Upload a picture of yourself miming!" with a "Browse" button, a "Submit" button, and a "Reset" button.
- Form 2:** A login form titled "Enter our Member's Only area". It has fields for "Username:" and "Password:", followed by a "Submit" button.
- Form 3:** A search form titled "Search our database". It has a text input field and a "Search" button.

Labels with arrows point to various UI elements: "Password control" points to the password field; "Text control" points to the username field; "form 1" points to the survey; "Check box" points to the "It's fun" checkbox; "Custom button" points to the "Search" button; "Radio button" points to the "Putting on the makeup" radio button; "Submit button" points to the "Submit" button of the login form; "Reset button" points to the "Reset" button of the survey; and "File select control" points to the "Browse" button.

Using Other Control Elements

In addition to form controls you can create using the `input` element, there are three that are elements in and of themselves.

Using the button Element

A button you create using the `button` element is similar to the buttons you create with the `input` element, except that these buttons can contain content within the opening and closing tags that actually appear on the buttons.

Note

The `button` element (as opposed to the `input` element of `type="button"`) is not supported by Netscape Navigator.

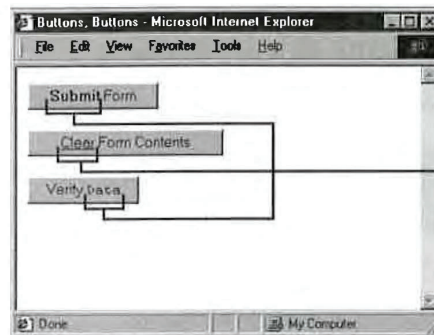
You can create three different types of buttons: submit, reset, and custom. To create them, start by entering the opening tag of the `button` element, followed by the `name` attribute. The `name` attribute names the control. Next, enter the `type` attribute and choose

the type of button you want to create. After that, close the opening tag and enter the text you want to appear in the button, and then finish things off with a closing button tag. The following code snippet creates the three types of buttons. Figure 14.11 shows three button elements, each with a different purpose. Note that I've used two line break tags, `<br>`, to give the buttons some room.

```
<button name="mysubmitbutton" type="submit"><b>Submit</b> Form</button><br /><br />
<button name="myresetbutton" type="reset"><u>Clear</u> Form Contents</button><br /><br />
<button name="mycustombutton" type="button">Verify <code>Data</code></button>
```

FIGURE 14.11

The button element enables you to include content and HTML elements inside buttons.



Because you can include other HTML element within a button element, the following example is perfectly acceptable:

```
<button name="myresetbutton" type="reset"><b>Clear</b> Form Contents</button>
```

Using the `<b>` element within the button boldfaces the word `Clear`.

You also can include images that are displayed as your button. Just enter the `<img>` element within the contents of the button, as in the following:

```
<button name="mysubmitbutton" type="submit">
  </button>
```

Create Large Text-Entry fields with `textarea`

The `textarea` element creates a large text-entry field where people can enter as much information as you allow. To create a `textarea` control, begin with the opening tag with the element name and then enter the `rows` and `cols` attributes before you close the tag. These mandatory attributes define the `textarea` region by rows and columns. You also can enter a helpful message that will appear in the text-entry field as an initial value. Following this, be sure to include the closing `textarea` tag. Figure 14.12 shows a `textarea` element in action.

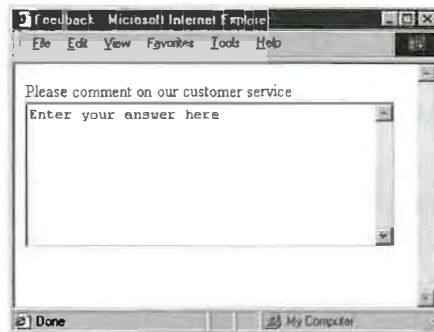
```

<p>Please comment on our customer service.
<textarea name="question4" rows="10" cols="60">
Enter your answer here
</textarea>
</p>

```

FIGURE 14.12

Use textarea to create large text-entry areas.



Note

You might be asking, "Where is the `isindex` element?" The `isindex` element creates a single-line text entry field much like the text control of the `input` element, and has been deprecated by the W3C. The syntax is `<isindex prompt="Prompt for user input">`. While writing this chapter, I tested this in the latest versions of Internet Explorer and Navigator and found that they no longer support `isindex`, so covering it in detail is not necessary.

Creating Menus with `select` and `option`

The `select` element creates a menu of choices from which the user can choose, much like the menus found in most software applications. By itself, `select` doesn't do much more than define how the menu will appear, either as a pull-down menu or scrollable menu. You must include at least one `option` or `optgroup` element to create the menus.

To create a pull-down `select` element with menu choices represented by `option` elements, start by entering the opening tag followed by a control name:

```
<select name="location">
```

Next, enter your menu items using the `option` element.

```

<select name="location">
  <option>Indiana</option>
  <option>Fuji</option>
  <option>Timbuktu</option>
  <option>Alaska</option>

```

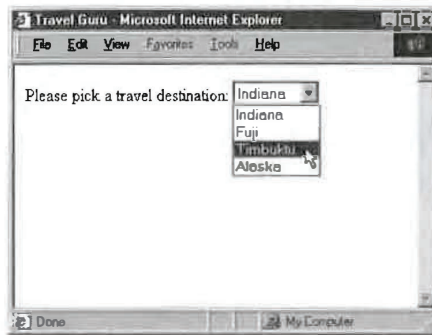
Finally, close the select element with the end tag.

```
<p>Please pick a travel destination:
<select name="location">
  <option>Indiana</option>
  <option>Fuji</option>
  <option>Timbuktu</option>
  <option>Alaska</option>
</select>
</p>
```

Figure 14.13 shows a select control that creates a pull-down menu that offers four options.

FIGURE 14.13

You can use select form controls to create pull-down menus.



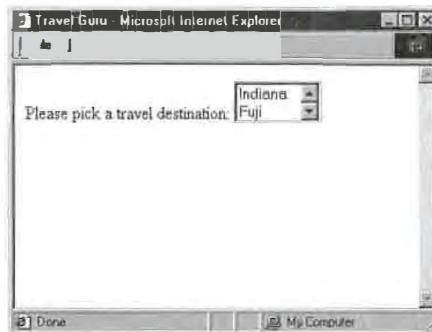
To create a scrollable list of items, just include the size attribute in the opening select tag, as in the following.

```
<select name="location" size="2">
```

Figure 14.14 shows the same select element as Figure 14.13, except that the size attribute is set to 2. Notice that it now turns into a scrollable menu.

FIGURE 14.14

You also can create scrollable lists using the select element.



The select element restricts the display area of the scroll menu to two items. Users will need to scroll to see more options.

To send back to the server data other than what is displayed, you can use the value attribute in conjunction with your option elements. This is useful if you are processing the data and want a numerical value or other coded value associated with the menu choice. The following code, for example, causes bw499 to be sent back for Courses instead of Basket Weaving 499.

```
<select name="courses">
  <option value="p101">Programming 101</option>
  <option value="e312">Ecomomics 312</option>
  <option value="pe221">Physical Education 221</option>
  <option value="bw499">Basket Weaving 499</option>
</select>
```

To preselect options, include the selected attribute in an option element, as in the following:

```
<select name="courses">
  <option value="p101">Programming 101</option>
  <option value="e312">Ecomomics 312</option>
  <option value="pe221" SELECTED>Physical Education 221</option>
  <option value="bw499">Basket Weaving 499</option>
</select>
```

This causes the menu to display Physical Education 221 as an initial value.

Thus far, you have created menus in which a user can select only one choice. You can easily change that to multiple choices by including the multiple attribute in the select element, as in the following:

```
<select name="courses" multiple>
```

Note

A user can choose multiple options Shift+clicking (for Windows) or Ctrl+clicking or Command+clicking (for Macintosh).

To finish off the creating menus discussion, I want to touch briefly on the optgroup element. Theoretically, this element organizes option components into groups and cascades the menus.

You should place optgroup elements in select elements. The optgroup elements also should contain option elements, as in the following.

```
<select name="futility">
  <optgroup label="Mild">
    <option>Swimming upstream</option>
    <option>Shaving with no shaving cream</option>
  </optgroup>
  <optgroup label="Extreme">
    <option>Squeezing blood from turnips</option>
    <option>Crying over spilled milk</option>
  </optgroup>
</select>
```

Note

Note that the `optgroup` element is not currently supported by the latest versions of Internet Explorer or Netscape Navigator.

Adding Extras

You've created all the form controls that will accept user input. Now it's time to add functionality and make the controls a bit friendlier.

Displaying Control labels

The `label` element displays helpful information for a form control. You should tie the `for` attribute to the control it labels. To create a label, begin with the opening `label` tag and then enter the `for` attribute. The value for this attribute, when present, must match the `id` attribute for the control it is labeling. Next, enter text that will serve as the label, and then close the element with the end `label` tag, as in the following.

```
<label for="control14">Who is your favorite NFL Quarterback?</label>
<input type="text" name="favqb" id="control14" />
```

Note

Although the `label` element is not supported by Netscape Navigator, the contents within the opening and closing `label` tags are displayed.

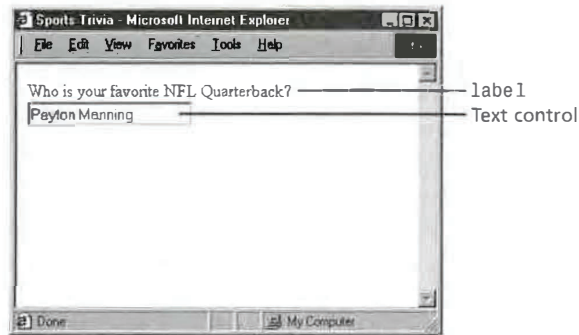
Figure 14.15 shows this text control with a label assigned to it.

If you define your form control within the `label` element, as shown in the following code, you can omit the `for` attribute:

```
<label>User name: <input type="text" name="username" /></label>
```

FIGURE 14.15

You can assign labels to any form control. Note that they are displayed with the control.



Grouping Controls with `fieldset` and `legend`

The `fieldset` element organizes form controls into groupings that appear in the Web browser. The `legend` element displays a caption for the `fieldset`. To create a `fieldset` element, start with the opening `fieldset` tag, followed by the `legend` element, as in the following code:

```
<fieldset>
  <legend>Oatmeal Varieties</legend>
```

Note

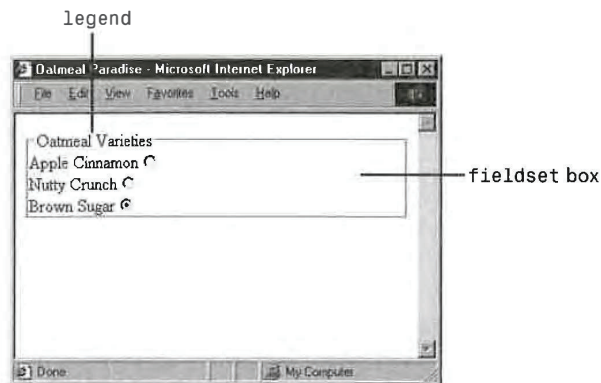
Netscape Navigator does not support `fieldset` or `legend`.

Notice that the `legend` contains text; however, it also could contain inline HTML elements for formatting the text. Next, enter your form controls and finish things off with the closing `fieldset` tag. Figure 14.16 shows the result.

```
<fieldset>
  <legend>Oatmeal Varieties</legend>
  <label>Apple Cinnamon<input type="radio" name="applecinnamon" /></label><br />
  <label>Nuttty Crunch<input type="radio" name="nuttycrunch" /></label><br />
  <label>Brown Sugar<input type="radio" name="brownsugar" /></label>
</fieldset>
```

FIGURE 14.16

The fieldset and legend elements enable you to organize your forms.



Changing the Default Tabbed Navigation

Using the keyboard to navigate some computer applications makes filling out forms much easier. Tabbed navigation enables users to press the Tab key to move from one form control to the next, in the order that you specify in your form.

Note

Although Netscape Navigator enables you to tab through form controls, you cannot change the order of tabbed navigation with the `tabindex` attribute.

To enable tabbed navigation, incorporate the `tabindex` attribute in all the form controls that you want to include this feature. Beginning with 1, order your form controls from start to finish. Controls that have duplicate `tabindex` values will be accessed in the order they appear in the HTML document. Following is an example of adding the attribute to one control:

```
<p>Enter your name: <input type="text" name="username" tabindex="1" /></p>
```

Using Access Keys

Access keys also make your forms easier to navigate. They assign a character to an element that moves the focus to that element when the users presses a key. To add an access key to a check box, use the following code:

```
<p>What are your interests?</p>
<p>Sports<input type="checkbox" name="sports" accesskey="S"></p>
<p>Music<input type="checkbox" name="music" accesskey="M"></p>
<p>Television<input type="checkbox" name="tv" accesskey="T"></p>
```

Note

Unfortunately, Netscape Navigator does not support the use of `accesskey`. While Internet Explorer does support it, users must press `Alt` in conjunction with the key in order for it to work. Other operating systems and Web browsers may require a different method.

Creating disabled and readonly Controls

You sometimes might want to display a form control without enabling your visitors to use the control or enter new information. To disable a control, add the `disabled` attribute to the form control, as in the following:

```
<p>What is the meaning of life?
<textarea name="question42" disabled>
Enter your answer here.
</textarea>
</p>
```

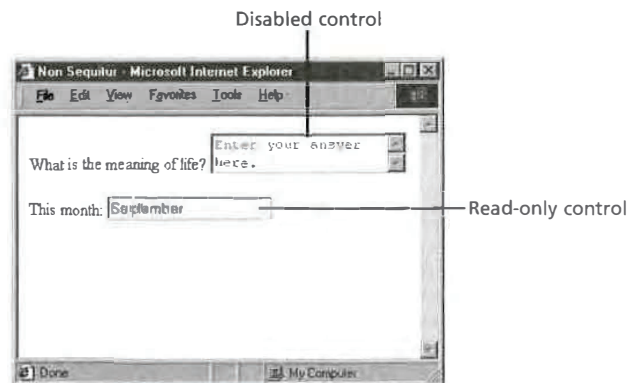
When displayed in a Web browser, the control will appear a light shade of gray, or dimmed, to indicate that it is unavailable. To create a read-only control, use the `readonly` attribute, as in the following example:

```
<p>This month: <input type="text" name="month" value="September" readonly /></p>
```

A read-only control appears normal; however, when visitors attempt to enter new information (or, in the case of buttons or check boxes, select them) they will find that they cannot change the value. Figure 14.17 shows both a disabled control and a read-only control.

FIGURE 14.17

Disabled controls are dimmed, while read-only controls appear normally—they just can't be changed.



Note

Netscape Navigator does not support the disabled or readonly attributes.

Exercise 14.3: Creating a Form with Advanced Form Control**▼ To Do**

Your final exercise for this chapter incorporates the form controls in the previous two sections into one large form in a Matchmaker Web page.

You begin with one large form. Open your template HTML document and create a form in the body of the document.

You'll want two major sections in this page: one section that asks questions about the visitor, and the other section that asks what the visitor is looking for in another person. This is perfectly suited for fieldset elements.

Create the first fieldset element now. Begin with the form tag and add a fieldset element and legend, as in the following:

```
<form action="http://www.fakeurl.com/cgi-bin/match.cgi" method="post">
<fieldset>
  <legend>Tell us about yourself.</legend>
```

Add a label for a select element, add the select element, and enter all the options. While you're at it, assign tabindexes and accesskeys as you go along.

```
<label for="control1">I have a funny bone that...</label>
<select name="humor" id="control1" tabindex="1" accesskey="h">
  <option value=".1">Please pick one</option>
  <option value=".1">.....</option>
  <option value="0">has been removed.</option>
  <option value="1">is my skull.</option>
  <option value="2">is ticklish.</option>
</select>
```

That wasn't too difficult. Now enter the second label and select element, as in the following:

```
<label for="control2">I like to take long walks...</label>
<select name="walks" id="control2" tabindex="2" accesskey="w">
  <option value=".1">Please pick one</option>
  <option value=".1">.....</option>
  <option value="Fields">in the corn fields at our farm.</option>
  <option value="Streets">on the streets of my city.</option>
  <option value="Beach">down the hall in my dorm.</option>
  <option value="Apartment">to the refrigerator.</option>
</select>
```

- ▼ Complete the `fieldset` by adding a final label and a `textarea` control. Be sure to include the closing `fieldset` tag.

```
<label for="control3">Why should we care?</label>
<textarea name="mycomments" id="control3" rows="5" cols="30"
tabindex="3"></textarea>
</fieldset>
```

You are almost halfway done. Use the following code to create a second `fieldset` that contains questions about what the visitor is looking for in another person.

```
<fieldset>
<legend>What are you looking for?</legend>
<label for="control4">I want...</label>
<select name="romance" id="control4" size="4" tabindex="4" accesskey="R">
  <option>somebody to watch movies with.</option>
  <option>a serious relationship.</option>
  <option>a Hot-N-Steamy romance.</option>
  <option>someone to clean the catbox.</option>
</select>
<label for="control5">Describe your perfect partner.</label>
<textarea name="lookingfor" id="control5" rows="5" cols="30"
tabindex="5"></textarea>
</fieldset>
```

With your two `fieldsets` complete, add a disabled control and a read-only control to make sure you've covered all your bases.

```
Check if you want to join our club.<input type="checkbox"
name="single" checked disabled />
Do you agree to pay us for this service?<input type="text"
name="pay" value="yes" readonly />
```

Now add the submit and reset buttons and close the form element. Notice that I'm including special graphics for the buttons by using the `button` element.

```
<button name="submit" type="submit" tabindex="6">

</button>
<button name="reset" type="reset" tabindex="7">

</button>
</form>
```

Now that your form is complete, you can add a title graphic at the top of the page and insert most of your form controls into a table. The complete code listing shows the final results.

INPUT

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
```

```

<title>Matchmaker Matchmaker</title>
</head>
<body>
<div align="center">

</div>
<form action="http://www.fakeurl.com/cgi-bin/match.cgi" method="post">
<table border="0" width="100%">
  <tr valign="top">
    <td width="50%">
      <table border="0">
        <tr>
          <td align="center">
            <fieldset>
              <legend>Tell us about yourself.</legend>
              <label for="control1">I have a funny bone that...</label><br>
            />
            <select name="humor" id="control1" tabindex="1"
accesskey="H">
              <option value=".1">Please pick one</option>
              <option value=".1">.....</option>
              <option value="0">has been removed.</option>
              <option value="1">is my skull.</option>
              <option value="2">is ticklish.</option>
            </select>
            <br />
            <label for="control2">I like to take long walks...</label>
            <select name="walks" id="control2" tabindex="2"
accesskey="W">
              <option value=".1">Please pick one</option>
              <option value=".1">.....</option>
              <option value="Fields">in the corn fields at our
farm.</option>
              <option value="Streets">on the streets of my
city.</option>
              <option value="Beach">down the hall in my dorm.</option>
              <option value="Apartment">to the refrigerator.</option>
            </select>
            <br />
            <label for="control3">Why should we care?</label>
            <br />
            <textarea name="mycomments" id="control3" rows="5" cols="30"
tabindex="3"></textarea>
          </fieldset>
        </td>
      </tr>
    </table>
  </td>
  <td width="50%">
    <table border="0">
      <tr>

```

```

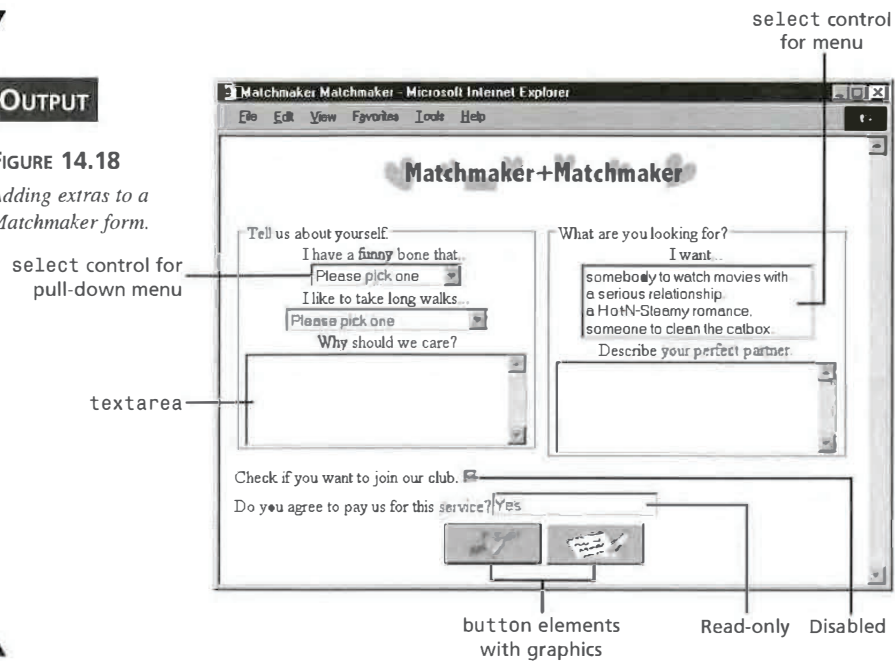
        <td align="center">
        <fieldset>
        <legend>What are you looking for?</legend>
        <label for="control4">I want...</label>
        <br />
        <select name="romance" id="control4" size="4" tabindex="4"
accesskey="R">
            <option>somebody to watch movies with.</option>
            <option>a serious relationship.</option>
            <option>a Hot-N-Steamy romance.</option>
            <option>someone to clean the catbox.</option>
        </select>
        <br />
        <label for="control5">Describe your perfect partner.</label>
        <textarea name="lookingfor" id="control5" rows="5" cols="30"
        ↪   tabindex="5"></textarea>
        </fieldset>
        </td>
    </tr>
    <tr>
        <td colspan="2">Check if you want to join our club.<input
type="checkbox"
        ↪   name="single" checked disabled /></td>
    </tr>
    <tr>
        <td colspan="2">Do you agree to pay us for this service?<input
        ↪   type="text" name="pay" value="Yes" readonly /></td>
    </tr>
</table>
<div align="center">
<table border="0">
    <tr>
        <td><button name="submit" type="submit" tabindex="6"></button></td>
        <td><button name="reset" type="reset" tabindex="7"></button></td>
    </tr>
</table>
</div>
</form>
</body>
</html>

```

▼ Figure 14.18 shows the final product of this exercise.

OUTPUT**FIGURE 14.18**

Adding extras to a Matchmaker form.



Planning Your Forms

Before you start creating complex forms for your Web pages, you should do some planning that will save you time and trouble in the long run.

First, decide what information you need to collect. That may sound obvious, but the time to think about this is when you aren't worried about the mechanics of creating the form.

Next, review this information and match each item with a type of form control. Ask yourself which type of control is most suited to the type of questions you are asking. If you need a yes or no answer, radio buttons or check boxes work great, while the textarea element is overkill. Try to make life easier for the person filling out the form by making the type of control fit the question. This way, analyzing the information using a script, if necessary, will be much easier.

You also need to coordinate with the person writing the CGI script to match with variables in the script the names you are going to assign to each control. There isn't much

point in naming every control before collaborating with the script author—after all, you will need all the names to match. You also can create look-up tables that contain expansive descriptions and allowable values of each form control.

Finally, you might want to consider validating form input through scripting. If you check the user's input in a client-side script (used *in* the Web page), you can ensure the data you receive when the form is submitted is what you want. You can assign script functions to each form element, including the submit button, using intrinsic events such as `onsubmit`.

Summary

As you can see, the wonderful world of forms is full of different types of form controls with a person visiting your Web site can interact. This truly is a way to make your Web pages interactive.

Do be cautious, however. Web surfers that are constantly bombarded with forms of one sort or another are likely to get tired of typing and move on to another site. You need to give them a reason for playing!

Table 14.1 summarizes the HTML tags used in this chapter. Remember these points and you can't go wrong:

- Use the `form` element to create your forms
- Always assign an action to a form
- Create form controls with the `input` element or the other form control elements
- Test your forms extensively

Table 14.1 HTML Tags from Day 14

Tag	Use
<code><form></code>	Creates an HTML form. You can have multiple forms within a document, but you cannot nest the forms.
<code>action</code>	An attribute of <code><form></code> that indicates the server-side script (with a URL path) that will process the form data.
<code>method</code>	An attribute of <code><form></code> that defines how the form data is sent to the server. Possible values are <code>get</code> and <code>post</code> .
<code><input></code>	A <code><form></code> element that creates controls for user input.

continues

Table 14.1 continued

<i>Tag</i>	<i>Use</i>
<code>type</code>	An attribute of <code><input></code> that indicates the type of form control. Possible values are <code>text</code> , <code>password</code> , <code>submit</code> , <code>reset</code> , <code>checkbox</code> , <code>radio</code> , <code>image</code> , <code>button</code> , <code>hidden</code> , and <code>file</code> , and are shown in the following list:
<code>text</code>	Creates a single-line text entry field.
<code>password</code>	Creates a single-line text entry field that masks user input.
<code>submit</code>	Creates a submit button that sends the form data to a server-side script.
<code>reset</code>	Creates a reset button that resets all form controls to their initial values.
<code>checkbox</code>	Creates a check box.
<code>radio</code>	Creates a radio button.
<code>image</code>	Creates a button from an image.
<code>button</code>	Creates a push button. The three types are <code>submit</code> , <code>reset</code> , and <code>push</code> , with no default action.
<code>hidden</code>	Creates a hidden form control that cannot be seen by the user.
<code>file</code>	Creates a file upload control that enables users to select a file with the form data to upload to the server.
<code><button></code>	Creates a button that can have HTML content.
<code><textarea></code>	A text-entry field with multiple lines.
<code><select></code>	A menu or scrolling list of items. Individual items are indicated by the <code><option></code> tag.
<code><option></code>	Individual items within a <code><select></code> element.
<code><label></code>	Creates a label associated with a form control.
<code><fieldset></code>	Organizes form controls into groups.
<code><legend></code>	Displays a caption for a <code>fieldset</code> element.

Workshop

If you've made it this far, I'm sure you still have a few questions, so I've included a few that I think are interesting. Afterward, test your retention by taking the quiz, and then expand your knowledge by tackling the exercises.

Q&A

Q I want to design a form that later on will be easy to analyze with a spreadsheet or database. How do I design my form to make this easier?

A First, make sure you have a default value for every form control in your form. That will fix the number of data entries and make importing it into a spreadsheet much easier. Don't be afraid to use values that can be manipulated programmatically. For example, I might make every Yes answer to my form a 1 and every No a 0 by using the value attributes. Then, when I import the data into my spreadsheet, I can chart and graph responses with 1s and 0s rather than text. You can use many other tricks, such as using lookup tables in your scripts to decode form responses. Experiment with what works best for you and your scripts.

Q I want to create a form and test it, but I don't have the script ready. Is there any way I can make sure the form is sending the right information with a working script?

A I run into this situation all the time! Fortunately, getting around it actually is very easy.

Within the opening form tag, modify the action attribute and make it a mailto link to your email address, as in the following:

```
<FORM ACTION="mailto:youremailaddress@isp.com" METHOD="post">
```

Now you can complete your test form and submit it without having a script ready. When you submit your form, it will be emailed to you as an attachment. Simply open the attachment in a text editor and voilà! Your form data is present.

Q I've heard that HTML 4.01 is under discussion. Will there be any changes to forms?

A HTML 4.01 currently is a Proposed Recommendation before the World Wide Web Consortium and forms have no major changes proposed. Following are those under consideration:

Adding the name attribute to the form element. This change would be for backward compatibility, enabling you to use the attribute to refer to the form in scripts.

Removing the reference to and example of the mailto value of the action attribute for the form element. It is unclear whether this would eliminate support for this action. The text of the Recommendation indicates that all actions are left up to the Web browser to implement.

Adding the missing ismap attribute to the input element to provide support for server-side image maps.

Adding an explanatory note for the `select` element, saying that when no option is chosen for a `select` element, no data is sent to the server for that element when the form is submitted.

Quiz

1. How many forms can you have on a Web page?
2. How do you create form controls such as radio buttons and check boxes?
3. Are passwords sent using a password control secure?
4. Explain the benefit of using hidden form controls.
5. What other technology do forms rely on?

Answers

1. Theoretically, you are limited only to the amount of memory on the client computer; however, you cannot nest forms. Another interesting tidbit is that each form control takes memory. I've run into situations in which I've used all the free memory of a computer by including too many form controls.
2. These form controls are created with the `input` element and have no life of their own, otherwise.
3. No!
4. Hidden form controls are more for you than the person filling out the form. By using unique `value` attributes, you can distinguish between different forms that may be sent to the same script or sent at different times.
5. In order for you to process the data forms, they must be paired with a server-side script through the `action` attribute. These scripts are written in Perl or other scripting languages.

Exercises

1. Check with your Internet Service Provider to see whether they have available scripts that you can use to process your forms. If you can use them, ask how the data is processed and which names you should use in your form controls. If you need to use forms and your ISP won't allow you to use their scripts, you should start looking elsewhere for a place to host your Web site.
2. Think of and visit some sites that might use forms, such as www.fedex.com. Look at which form controls they use, how they arrange them, and peek at the source to see the HTML code.

