



CENGAGE

*NEW PERSPECTIVES*

# HTML5, CSS3, and JavaScript

Sixth Edition

Carey

# Want to turn C's into A's? Obviously, right?

But the right way to go about it isn't always so obvious. Go digital to get the grades. MindTap's customizable study tools and eTextbook give you everything you need all in one place.

Engage with your course content, enjoy the flexibility of studying anytime and anywhere, stay connected to assignment due dates and instructor notifications with the MindTap Mobile app...

*and most of all...EARN BETTER GRADES.*



TO GET STARTED VISIT  
[WWW.CENGAGE.COM/STUDENTS/MINDTAP](http://WWW.CENGAGE.COM/STUDENTS/MINDTAP)

 CENGAGE  
Learning

MindTap®



**NEW PERSPECTIVES ON**

# **HTML5, CSS3, and JavaScript**

*6th Edition*

**Patrick Carey**



---

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

This is an electronic version of the print textbook. Due to electronic rights restrictions, some third party content may be suppressed.

Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. The publisher reserves the right to remove content from this title at any time if subsequent rights restrictions require it. For valuable information on pricing, previous editions, changes to current editions, and alternate formats, please visit [www.cengage.com/highered](http://www.cengage.com/highered) to search by ISBN#, author, title, or keyword for materials in your areas of interest.

Important Notice: Media content referenced within the product description or the product text may not be available in the eBook version.

**New Perspectives on HTML5, CSS3, and JavaScript**  
**6th Edition**  
**Patrick Carey**

SVP, GM Science, Technology & Math: Balraj S. Kalsi

Senior Product Director: Kathleen McMahon

Product Team Manager: Kristin McNary

Associate Product Manager: Kate Mason

Senior Director, Development: Julia Caballero

Senior Content Development Manager: Leigh  
Hefferon

Associate Content Developer: Maria Garguilo

Development Editor: Pam Conrad

Product Assistant: Jake Toth

VP, Marketing for Science, Technology, & Math: Jason  
Sakos

Marketing Director: Michele McTighe

Marketing Manager: Stephanie Albracht

Production Director: Patty Stephan

Senior Content Project Manager:  
Jennifer K. Feltri-George

Art Director: Diana Graham

Manufacturing Planner: Fola Orekoya

Cover image(s): iStockPhoto.com/moncsicsi

Compositor: GEX Publishing Services

**Notice to the Reader**

Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions. The publisher makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.

© 2018, 2013 Cengage Learning

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transmitted, stored or used in any form or by any means graphic, electronic, or mechanical, including but not limited to photocopying, recording, scanning, digitizing, taping, Web distribution, information networks, or information storage and retrieval systems, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the publisher.

For product information and technology assistance, contact us at  
**Cengage Learning Customer & Sales Support, 1-800-354-9706**

For permission to use material from this text or product, submit all  
requests online at **[www.cengage.com/permissions](http://www.cengage.com/permissions)**  
Further permissions questions can be emailed to  
**[permissionrequest@cengage.com](mailto:permissionrequest@cengage.com)**

Library of Congress Control Number: 2017939432

Softcover ISBN: 978-1-305-50392-2

Loose-leaf ISBN: 978-1-337-68576-4

**Cengage Learning**

20 Channel Center Street

Boston, MA 02210

USA

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

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

For your course and learning solutions, visit **[www.cengage.com](http://www.cengage.com)**

Purchase any of our products at your local college store or at our  
preferred online store **[www.cengagebrain.com](http://www.cengagebrain.com)**

Some of the product names and company names used in this book have  
been used for identification purposes only and may be trademarks or regis-  
tered trademarks of their respective manufacturers and sellers.

Disclaimer: Any fictional data related to persons or companies or URLs used  
throughout this book is intended for instructional purposes only. At the  
time this book was printed, any such data was fictional and not belonging  
to any real persons or companies.

Microsoft and the Windows logo are registered trademarks of Microsoft  
Corporation in the United States and/or other countries. Cengage Learning  
is an independent entity from Microsoft Corporation and not affiliated with  
Microsoft in any manner.

# Preface

The New Perspectives Series' critical-thinking, problem-solving approach is the ideal way to prepare students to transcend point-and-click skills and take advantage of all that HTML5, CSS3, and JavaScript have to offer.

In developing the New Perspectives Series, our goal was to create books that give students the software concepts and practical skills they need to succeed beyond the classroom. We've updated our proven case-based pedagogy with more practical content to make learning skills more meaningful to students. With the New Perspectives Series, students understand *why* they are learning *what* they are learning, and they are fully prepared to apply their skills to real-life situations.

*"I love this text because it provides detailed instructions and real-world application examples. It is ideal for classroom and online instruction. At the end of the term, my students comment on how much they've learned and put to use outside the classroom."*

—Bernice Howard  
St. Johns River Community  
College

## About This Book

This book provides thorough coverage of HTML5, CSS3, and JavaScript, and includes the following:

- Up-to-date coverage of using HTML5 to create structured websites.
- Instruction on the most current CSS3 styles to create visually-interesting pages and captivating graphical designs.
- Working with browser developer tools to aid in the creation and maintenance of fully-functioning websites.

*New for this edition!*

- Coverage of responsive design techniques to create website designs that can scale to mobile, tablet, and desktop devices.
- Hands-on study of new HTML elements and CSS styles including layouts using flexboxes and grid frameworks.
- Exploration of CSS3 styles for graphic design, including image borders, drop shadows, gradient fills, 2D and 3D transformations, and graphic filters.
- Exploration of responsive design for web tables.
- Coverage of CSS styles for animation and transitions.
- Coverage of JavaScript arrays, program loops, and conditional statements.
- Coverage of JavaScript methods for form validation and e-commerce.
- Coverage of custom objects, properties, and methods used in object-based programming.

## System Requirements

This book assumes that students have an Internet connection, a text editor, and a current browser that supports HTML5 and CSS3. The following is a list of the most recent versions of the major browsers at the time this text was published: Internet Explorer 11, Microsoft Edge 38, Firefox 52, Safari 11, Opera 43, and Google Chrome 57. More recent versions may have come out since the publication of this book. Students should go to the Web browser home page to download the most current version. All browsers interpret HTML5 and CSS3 code in slightly different ways. It is highly recommended that students have several different browsers installed on their systems for comparison and, if possible, access to a mobile browser or a mobile emulator. Students might also want to run older versions of these browsers to highlight compatibility issues. The screenshots in this book were produced using Google Chrome 57 running on Windows 10 (64-bit),

unless otherwise noted. If students are using different devices, browsers, or operating systems, their screens might vary from those shown in the book; this should not present any problems in completing the tutorials.

*"New Perspectives texts provide up-to-date, real-world application of content, making book selection easy. The step-by-step, hands-on approach teaches students concepts they can apply immediately."*

—John Taylor

Southeastern Technical  
College

## VISUAL OVERVIEW

# The New Perspectives Approach

## Context

Each tutorial begins with a problem presented in a “real-world” case that is meaningful to students. The case sets the scene to help students understand what they will do in the tutorial.

## Hands-on Approach

Each tutorial is divided into manageable sessions that combine reading and hands-on, step-by-step work. Colorful screenshots help guide students through the steps. **Trouble?** tips, which anticipate common mistakes or problems, help students stay on track and continue with the tutorial.

## Visual Overviews

Each session begins with a Visual Overview, a two-page spread that includes colorful, enlarged figures with numerous callouts and key term definitions, giving students a comprehensive preview of the topics covered in the session, as well as a handy study guide.

## PROSKILLS

## ProSkills Boxes

ProSkills boxes provide guidance for applying concepts to real-world, professional situations, involving one or more of the following soft skills: decision making, problem solving, teamwork, verbal communication, and written communication.

## KEY STEP

## Key Steps

Important steps are highlighted in yellow with attached margin notes to help students pay close attention to completing the steps correctly and avoid time-consuming rework.

## INSIGHT

## InSight Boxes

InSight boxes offer expert advice and best practices to help students achieve a deeper understanding of the concepts behind the software features and skills.

## TIP

## Margin Tips

Margin Tips provide helpful hints and shortcuts for more efficient use of the software. The Tips appear in the margin at key points throughout each tutorial, giving students extra information when and where they need it.

## REVIEW

## Assessment

## APPLY

Retention is a key component to learning. At the end of each session, a series of Quick Check questions helps students test their understanding of the material before moving on. Engaging end-of-tutorial Review Assignments and Case Problems have always been a hallmark feature of the New Perspectives Series. Colorful bars and brief descriptions accompany the exercises, making it easy to understand both the goal and level of challenge a particular assignment holds.

## CHALLENGE

## CREATE

## REFERENCE

## Reference

## GLOSSARY/INDEX

Within each tutorial, Reference boxes appear before a set of steps to provide a succinct summary or preview of how to perform a task. In addition, each book includes a combination Glossary/Index to promote easy reference of material.

INTRODUCTORY

COMPREHENSIVE

## Our Complete System of Instruction

### Coverage To Meet Your Needs

Whether you're looking for just a small amount of coverage or enough to fill a semester-long class, we can provide you with a textbook that meets your needs.

- Introductory books contain an average of 5 to 8 tutorials and include essential skills on the books concepts.
- Comprehensive books, which cover additional concepts and skills in depth, are great for a full-semester class, and contain 9 to 12+ tutorials.

So, if you are looking for just the essential skills or more complete in-depth coverage of a topic, we have an offering available to meet your needs. Go to our website or contact your Cengage Learning sales representative to find out what else we offer.

### MindTap

MindTap is a personalized learning experience with relevant assignments that guide students to analyze, apply, and improve thinking, allowing you to measure skills and outcomes with ease.

For instructors: personalized teaching becomes yours with a Learning Path that is built with key student objectives. Control what students see and when they see it. Use as-is, or match to your syllabus exactly: hide, rearrange, add, or create your own content.

For students: a unique Learning Path of relevant readings, multimedia, and activities that guide you through basic knowledge and comprehension to analysis and application.

Better outcomes: empower instructors and motivate students with analytics and reports that provide a snapshot of class progress, time in course, engagement, and completion rates.

The MindTap for HTML5, CSS3, and JavaScript includes coding labs, study tools, and interactive quizzing, all integrated into an eReader that includes the full content of the printed text.

### Instructor Resources

We offer more than just a book. We have all the tools you need to enhance your lectures, check students' work, and generate exams in a new, easier-to-use and completely revised package. This book's Instructor's Manual, Cengage testbank, PowerPoint presentations, data files, solution files, figure files, and a sample syllabus are all available at [sso.cengage.com](http://sso.cengage.com).

## Acknowledgments

I would like to thank the people who worked so hard to make this book possible. Special thanks to my developmental editor, Pam Conrad, for her hard work, attention to detail, and valuable insights, and to Associate Content Developer, Maria Garguilo, who has worked tirelessly in overseeing this project and made my task so much easier with enthusiasm and good humor. Other people at Cengage who deserve credit are Kathleen McMahon, Sr. Product Manager; Kate Mason, Associate Product Manager; Jake Toth, Product Assistant; Jen Feltri-George, Senior Content Project Manager; Diana Graham, Art Director; Fola Orekoya, Manufacturing Planner; GEX Publishing Services, Compositor, as well as the MQA tester Danielle Shaw.



Feedback is an important part of writing any book, and thanks go to the following reviewers for their helpful ideas and comments: Alison Consol, Wake Technical Community College; Dana Hooper, The University of Alabama; Kenneth Kleiner, Fayetteville Technical Community College; and Laurie Crawford, Franklin University.

I want to thank my wife Joan and my six children for their love, encouragement, and patience in putting up with a sometimes distracted husband and father. This book is dedicated to my grandchildren: Benedict, David, Elanor, and Nicholas.

– Patrick Carey

# BRIEF CONTENTS

## HTML

### Level I Tutorials

- Tutorial 1** Getting Started with HTML5 ..... HTML 1  
*Creating a Website for a Food Vendor*
- Tutorial 2** Getting Started with CSS ..... HTML 83  
*Designing a Website for a Fitness Club*

### Level II Tutorials

- Tutorial 3** Designing a Page Layout ..... HTML 169  
*Creating a Website for a Chocolatier*
- Tutorial 4** Graphic Design with CSS ..... HTML 257  
*Creating a Graphic Design for a Genealogy Website*
- Tutorial 5** Designing for the Mobile Web ..... HTML 341  
*Creating a Mobile Website for a Daycare Center*

### Level III Tutorials

- Tutorial 6** Working with Tables and Columns. .... HTML 433  
*Creating a Program Schedule for a Radio Station*
- Tutorial 7** Designing a Web Form. .... HTML 499  
*Creating a Survey Form*
- Tutorial 8** Enhancing a Website with Multimedia. .... HTML 585  
*Working with Sound, Video, and Animation*
- Tutorial 9** Getting Started with JavaScript. .... HTML 665  
*Creating a Countdown Clock*
- Tutorial 10** Exploring Arrays, Loops, and Conditional Statements. .... HTML 735  
*Creating a Monthly Calendar*
- Tutorial 11** Working with Events and Styles. .... HTML 809  
*Designing an Interactive Puzzle*
- Tutorial 12** Working with Document Nodes and Style Sheets ..... HTML 891  
*Creating a Dynamic Document Outline*
- Tutorial 13** Programming for Web Forms. .... HTML 969  
*Creating Forms for Orders and Payments*
- Tutorial 14** Exploring Object-Based Programming. .... HTML 1061  
*Designing an Online Poker Game*
- Appendix A** Color Names with Color Values, and HTML Character Entities ..... HTML A1
- Appendix B** HTML Elements and Attributes ..... HTML B1
- Appendix C** Cascading Styles and Selectors ..... HTML C1
- Appendix D** Making the Web More Accessible. .... HTML D1

**Appendix E** Designing for the Web . . . . . HTML E1

**Appendix F** Page Validation with XHTML. . . . . HTML F1

**Glossary** **REF 1**

**Index** **REF 13**

# TABLE OF CONTENTS

<b>Preface</b> .....	<b>iii</b>
----------------------	------------

## HTML LEVEL I TUTORIALS

### Tutorial 1 Getting Started with HTML5

<i>Creating a Website for a Food Vendor</i> .....	<b>HTML 1</b>
---	---------------

#### SESSION 1.1.....HTML 2

Exploring the World Wide Web .....	HTML 4
Networks .....	HTML 4
Locating Information on a Network .....	HTML 4
Web Pages and Web Servers .....	HTML 5
Introducing HTML .....	HTML 5
The History of HTML .....	HTML 5
Tools for Working with HTML .....	HTML 6
Testing your Code .....	HTML 7
Supporting the Mobile Web .....	HTML 7
Exploring an HTML Document .....	HTML 7
The Document Type Declaration .....	HTML 8
Introducing Element Tags .....	HTML 9
The Element Hierarchy .....	HTML 10
Introducing Element Attributes .....	HTML 11
Handling White Space .....	HTML 12
Viewing an HTML File in a Browser .....	HTML 12
Creating an HTML File .....	HTML 14
Creating the Document Head .....	HTML 15
Setting the Page Title .....	HTML 15
Adding Metadata to the Document .....	HTML 16
Adding Comments to your Document .....	HTML 18
Session 1.1 Quick Check .....	HTML 21

#### SESSION 1.2.....HTML 22

Writing the Page Body .....	HTML 24
Using Sectioning Elements .....	HTML 24
Comparing Sections in HTML4 and HTML5 .....	HTML 26
Using Grouping Elements .....	HTML 26
Using Text-Level Elements .....	HTML 29
Linking an HTML Document to a Style Sheet .....	HTML 32
Working with Character Sets and Special Characters .....	HTML 33
Character Encoding .....	HTML 33
Character Entity References .....	HTML 34

Working with Inline Images .....	HTML 36
Line Breaks and Other Empty Elements .....	HTML 38
Working with Block Quotes and Other Elements .....	HTML 39
Session 1.2 Quick Check .....	HTML 45

#### SESSION 1.3.....HTML 46

Working with Lists .....	HTML 48
Ordered Lists .....	HTML 48
Unordered Lists .....	HTML 49
Description Lists .....	HTML 51
Navigation Lists .....	HTML 55
Working with Hypertext Links .....	HTML 57
Turning an Inline Image into a Link .....	HTML 59
Specifying the Folder Path .....	HTML 60
Absolute Paths .....	HTML 61
Relative Paths .....	HTML 61
Setting the Base Path .....	HTML 62
Linking to a Location within a Document .....	HTML 63
Marking Locations with the id Attribute .....	HTML 63
Linking to an id .....	HTML 63
Anchors and the name Attribute .....	HTML 63
Linking to the Internet and Other Resources .....	HTML 64
Linking to a Web Resource .....	HTML 65
Linking to an E-Mail Address .....	HTML 65
Linking to a Phone Number .....	HTML 67
Working with Hypertext Attributes .....	HTML 68
Session 1.3 Quick Check .....	HTML 70
Review Assignments .....	HTML 71
Case Problems .....	HTML 74

### Tutorial 2 Getting Started with CSS

<i>Designing a Website for a Fitness Club</i> .....	<b>HTML 83</b>
---	----------------

#### SESSION 2.1.....HTML 84

Introducing CSS .....	HTML 86
Types of Style Sheets .....	HTML 86
Viewing a Page Using Different Style Sheets .....	HTML 87
Exploring Style Rules .....	HTML 90
Browser Extensions .....	HTML 90
Embedded Style Sheets .....	HTML 91
Inline Styles .....	HTML 92
Style Specificity and Precedence .....	HTML 92



Style Inheritance .....	HTML 93
Browser Developer Tools .....	HTML 93
Creating a Style Sheet. ....	HTML 95
Writing Style Comments .....	HTML 95
Defining the Character Encoding. ....	HTML 96
Importing Style Sheets .....	HTML 96
Working with Color in CSS .....	HTML 97
Color Names. ....	HTML 97
RGB Color Values .....	HTML 98
HSL Color Values. ....	HTML 99
Defining Semi-Opaque Colors .....	HTML 100
Setting Text and Background Colors .....	HTML 101
Employing Progressive Enhancement. ....	HTML 104
Session 2.1 Quick Check .....	HTML 105

## **SESSION 2.2 .....HTML 106**

Exploring Selector Patterns .....	HTML 108
Contextual Selectors .....	HTML 108
Attribute Selectors. ....	HTML 111
Working with Fonts .....	HTML 115
Choosing a Font .....	HTML 115
Exploring Web Fonts .....	HTML 118
The @font-face Rule. ....	HTML 118
Setting the Font Size .....	HTML 121
Absolute Units .....	HTML 121
Relative Units .....	HTML 121
Scaling Fonts with ems and rems. ....	HTML 122
Using Viewport Units .....	HTML 123
Sizing Keywords. ....	HTML 123
Controlling Spacing and Indentation .....	HTML 125
Working with Font Styles. ....	HTML 127
Aligning Text Horizontally and Vertically. ....	HTML 128
Combining All Text Formatting in a Single Style	HTML 128
Session 2.2 Quick Check. ....	HTML 131

## **SESSION 2.3. ....HTML 132**

Formatting Lists .....	HTML 134
Choosing a List Style Type .....	HTML 134
Creating an Outline Style. ....	HTML 134
Using Images for List Markers .....	HTML 137
Setting the List Marker Position. ....	HTML 138
Working with Margins and Padding .....	HTML 139
Setting the Padding Space. ....	HTML 140
Setting the Margin and the Border Spaces .....	HTML 142

Using Pseudo-Classes and Pseudo-Elements .....	HTML 145
Pseudo-Classes. ....	HTML 145
Pseudo-classes for Hypertext. ....	HTML 148
Pseudo-Elements .....	HTML 151
Generating Content with CSS .....	HTML 152
Displaying Attribute Values .....	HTML 153
Inserting Quotation Marks .....	HTML 154
Session 2.3 Quick Check .....	HTML 157
Review Assignments .....	HTML 158
Case Problems. ....	HTML 160

## **HTML LEVEL II TUTORIALS**

### **Tutorial 3 Designing a Page Layout**

<i>Creating a Website for a Chocolatier. ....</i>	<b>HTML 169</b>
---	-----------------

## **SESSION 3.1 .....HTML 170**

Introducing the display Style .....	HTML 172
Creating a Reset Style Sheet. ....	HTML 172
Exploring Page Layout Designs. ....	HTML 176
Fixed, Fluid, and Elastic Layouts. ....	HTML 176
Working with Width and Height .....	HTML 178
Setting Maximum and Minimum Dimensions ...	HTML 178
Centering a Block Element .....	HTML 181
Vertical Centering .....	HTML 182
Floating Page Content. ....	HTML 183
Clearing a Float .....	HTML 187
Refining a Floated Layout. ....	HTML 191
Working with Container Collapse. ....	HTML 195
Session 3.1 Quick Check .....	HTML 199

## **SESSION 3.2 .....HTML 200**

Introducing Grid Layouts .....	HTML 202
Overview of Grid-Based Layouts .....	HTML 202
Fixed and Fluid Grids .....	HTML 203
CSS Frameworks. ....	HTML 204
Setting up a Grid. ....	HTML 204
Designing the Grid Rows .....	HTML 208
Designing the Grid Columns .....	HTML 209
Adding the Page Content. ....	HTML 210
Outlining a Grid. ....	HTML 216
Introducing CSS Grids. ....	HTML 219
Defining a CSS Grid .....	HTML 219
Assigning Content to Grid Cells. ....	HTML 220
Session 3.2 Quick Check .....	HTML 223

**SESSION 3.3. . . . .HTML 224**

Positioning Objects . . . . .	HTML 226
The CSS Positioning Styles . . . . .	HTML 226
Relative Positioning . . . . .	HTML 226
Absolute Positioning . . . . .	HTML 227
Fixed and Inherited Positioning . . . . .	HTML 230
Using the Positioning Styles . . . . .	HTML 230
Handling Overflow . . . . .	HTML 240
Clipping an Element . . . . .	HTML 243
Stacking Elements . . . . .	HTML 244
Session 3.3 Quick Check . . . . .	HTML 246
Review Assignments . . . . .	HTML 247
Case Problems . . . . .	HTML 249

**Tutorial 4 Graphic Design with CSS***Creating a Graphic Design for a Genealogy Website . . . HTML 257***SESSION 4.1. . . . .HTML 258**

Creating Figure Boxes . . . . .	HTML 260
Exploring Background Styles . . . . .	HTML 264
Tiling a Background Image . . . . .	HTML 265
Attaching the Background Image . . . . .	HTML 267
Setting the Background Image Position . . . . .	HTML 267
Defining the Extent of the Background . . . . .	HTML 268
Sizing and Clipping an Image . . . . .	HTML 269
The background Property . . . . .	HTML 270
Adding Multiple Backgrounds . . . . .	HTML 272
Working with Borders . . . . .	HTML 273
Setting Border Width and Color . . . . .	HTML 274
Setting the Border Design . . . . .	HTML 274
Creating Rounded Corners . . . . .	HTML 277
Applying a Border Image . . . . .	HTML 281
Session 4.1 Quick Check . . . . .	HTML 285

**SESSION 4.2. . . . .HTML 286**

Creating Drop Shadows . . . . .	HTML 288
Creating a Text Shadow . . . . .	HTML 288
Creating a Box Shadow . . . . .	HTML 290
Applying a Color Gradient . . . . .	HTML 296
Creating a Linear Gradient . . . . .	HTML 296
Gradients and Color Stops . . . . .	HTML 299
Creating a Radial Gradient . . . . .	HTML 301
Repeating a Gradient . . . . .	HTML 305
Creating Semi-Transparent Objects . . . . .	HTML 307
Session 4.2 Quick Check . . . . .	HTML 309

**SESSION 4.3. . . . .HTML 310**

Transforming Page Objects . . . . .	HTML 312
Transformations in Three Dimensions . . . . .	HTML 316
Understanding Perspective . . . . .	HTML 317
Exploring CSS Filters . . . . .	HTML 320
Working with Image Maps . . . . .	HTML 324
Defining a Client-Side Image Map . . . . .	HTML 324
Applying an Image Map . . . . .	HTML 328
Session 4.3 Quick Check . . . . .	HTML 330
Review Assignments . . . . .	HTML 331
Case Problems . . . . .	HTML 334

**Tutorial 5 Designing for the Mobile Web***Creating a Mobile Website for a Daycare Center . . . HTML 341***SESSION 5.1. . . . .HTML 342**

Introducing Responsive Design . . . . .	HTML 344
Introducing Media Queries . . . . .	HTML 345
The @media Rule . . . . .	HTML 346
Media Queries and Device Features . . . . .	HTML 347
Applying Media Queries to a Style Sheet . . . . .	HTML 349
Exploring Viewports and Device Width . . . . .	HTML 352
Creating a Mobile Design . . . . .	HTML 355
Creating a Pulldown Menu with CSS . . . . .	HTML 356
Testing your Mobile Website . . . . .	HTML 359
Creating a Tablet Design . . . . .	HTML 363
Creating a Desktop Design . . . . .	HTML 367
Session 5.1 Quick Check . . . . .	HTML 371

**SESSION 5.2. . . . .HTML 372**

Introducing Flexible Boxes . . . . .	HTML 374
Defining a Flexible Box . . . . .	HTML 374
Cross-Browser Flexboxes . . . . .	HTML 375
Setting the Flexbox Flow . . . . .	HTML 375
Working with Flex Items . . . . .	HTML 377
Setting the Flex Basis . . . . .	HTML 377
Defining the Flex Growth . . . . .	HTML 378
Defining the Shrink Rate . . . . .	HTML 379
The flex Property . . . . .	HTML 381
Applying a Flexbox Layout . . . . .	HTML 382
Reordering Page Content with Flexboxes . . . . .	HTML 388
Exploring Flexbox Layouts . . . . .	HTML 390
Aligning Items along the Main Axis . . . . .	HTML 390
Aligning Flex Lines . . . . .	HTML 391
Aligning Items along the Cross Axis . . . . .	HTML 392

Creating a Navicon Menu . . . . .	HTML 394
Session 5.2 Quick Check . . . . .	HTML 399
<b>SESSION 5.3. . . . .</b>	<b>HTML 400</b>
Designing for Printed Media . . . . .	HTML 402
Previewing the Print Version . . . . .	HTML 402
Applying a Media Query for Printed Output . . . . .	HTML 403
Working with the @page Rule . . . . .	HTML 405
Setting the Page Size . . . . .	HTML 405
Using the Page Pseudo-Classes . . . . .	HTML 406
Page Names and the Page Property . . . . .	HTML 406
Formatting Hypertext Links for Printing . . . . .	HTML 412
Working with Page Breaks. . . . .	HTML 415
Preventing Page Breaks . . . . .	HTML 416
Working with Widows and Orphans. . . . .	HTML 418
Session 5.3 Quick Check . . . . .	HTML 421
Review Assignments . . . . .	HTML 422
Case Problems. . . . .	HTML 425

## HTML LEVEL III TUTORIALS

### Tutorial 6 Working with Tables and Columns

<i>Creating a Program Schedule for a Radio Station. . . . .</i>	<b>HTML 433</b>
---	-----------------

#### SESSION 6.1. . . . .HTML 434

Introducing Web Tables . . . . .	HTML 436
Marking Tables and Table Rows. . . . .	HTML 436
Marking Table Headings and Table Data . . . . .	HTML 438
Adding Table Borders with CSS . . . . .	HTML 442
Spanning Rows and Columns . . . . .	HTML 447
Creating a Table Caption . . . . .	HTML 453
Session 6.1 Quick Check . . . . .	HTML 457

#### SESSION 6.2. . . . .HTML 458

Creating Row Groups . . . . .	HTML 460
Creating Column Groups. . . . .	HTML 464
Exploring CSS Styles and Web Tables. . . . .	HTML 467
Working with Width and Height. . . . .	HTML 468
Applying Table Styles to Other Page Elements . . . . .	HTML 472
Tables and Responsive Design. . . . .	HTML 474
Designing a Column Layout . . . . .	HTML 478
Setting the Number of Columns . . . . .	HTML 478
Defining Columns Widths and Gaps. . . . .	HTML 481
Managing Column Breaks . . . . .	HTML 484
Spanning Cell Columns . . . . .	HTML 485

Session 6.2 Quick Check . . . . .	HTML 488
Review Assignments . . . . .	HTML 489
Case Problems. . . . .	HTML 491

### Tutorial 7 Designing a Web Form

<i>Creating a Survey Form. . . . .</i>	<b>HTML 499</b>
--	-----------------

#### SESSION 7.1. . . . .HTML 500

Introducing Web Forms. . . . .	HTML 502
Parts of a Web Form. . . . .	HTML 502
Forms and Server-Based Programs . . . . .	HTML 503
Starting a Web Form . . . . .	HTML 504
Interacting with the Web Server . . . . .	HTML 505
Creating a Field Set. . . . .	HTML 507
Marking a Field Set. . . . .	HTML 507
Adding a Field Set Legend. . . . .	HTML 508
Creating Input Boxes. . . . .	HTML 510
Input Types. . . . .	HTML 510
Input Types and Virtual Keyboards . . . . .	HTML 514
Adding Field Labels. . . . .	HTML 514
Designing a Form Layout. . . . .	HTML 516
Defining Default Values and Placeholders . . . . .	HTML 523
Session 7.1 Quick Check. . . . .	HTML 527

#### SESSION 7.2. . . . .HTML 528

Entering Date and Time Values . . . . .	HTML 530
Creating a Selection List . . . . .	HTML 531
Working with Select Attributes . . . . .	HTML 533
Grouping Selection Options. . . . .	HTML 535
Creating Option Buttons. . . . .	HTML 537
Creating Check Boxes . . . . .	HTML 540
Creating a Text Area Box. . . . .	HTML 542
Session 7.2 Quick Check . . . . .	HTML 545

#### SESSION 7.3. . . . .HTML 546

Entering Numeric Data . . . . .	HTML 548
Creating a Spinner Control . . . . .	HTML 548
Creating a Range Slider . . . . .	HTML 550
Suggesting Options with Data Lists . . . . .	HTML 553
Working with Form Buttons. . . . .	HTML 556
Creating a Command Button . . . . .	HTML 556
Creating Submit and Reset Buttons. . . . .	HTML 556
Designing a Custom Button . . . . .	HTML 559

Validating a Web Form . . . . .	HTML 559
Identifying Required Values . . . . .	HTML 559
Validating Based on Data Type . . . . .	HTML 561
Testing for a Valid Pattern . . . . .	HTML 562
Defining the Length of the Field Value . . . . .	HTML 564
Applying Inline Validation . . . . .	HTML 565
Using the <code>focus</code> Pseudo-Class . . . . .	HTML 565
Pseudo-Classes for Valid and Invalid Data . . . . .	HTML 567
Session 7.3 Quick Check . . . . .	HTML 570
Review Assignments . . . . .	HTML 571
Case Problems . . . . .	HTML 574

## **Tutorial 8 Enhancing a Website with Multimedia**

*Working with Sound, Video, and Animation . . . . .* **HTML 585**

### **SESSION 8.1 . . . . .HTML 586**

Introducing Multimedia on the Web . . . . .	HTML 588
Understanding Codecs and Containers . . . . .	HTML 588
Understanding Plug-Ins . . . . .	HTML 589
Working with the <code>audio</code> Element . . . . .	HTML 591
Browsers and Audio Formats . . . . .	HTML 591
Applying Styles to the Media Player . . . . .	HTML 594
Providing a Fallback to an Audio Clip . . . . .	HTML 596
Exploring Embedded Objects . . . . .	HTML 598
Plug-In Attributes . . . . .	HTML 598
Plug-Ins as Fallback Options . . . . .	HTML 599
Session 8.1 Quick Check . . . . .	HTML 599

### **SESSION 8.2 . . . . .HTML 600**

Exploring Digital Video . . . . .	HTML 602
Video Formats and Codecs . . . . .	HTML 602
Using the HTML5 <code>video</code> Element . . . . .	HTML 603
Adding a Text Track to Video . . . . .	HTML 607
Making Tracks with WebVTT . . . . .	HTML 608
Placing the Cue Text . . . . .	HTML 611
Applying Styles to Track Cues . . . . .	HTML 612
Using Third-Party Video Players . . . . .	HTML 616
Exploring the Flash Player . . . . .	HTML 617
Embedding Videos from YouTube . . . . .	HTML 618
HTML5 Video Players . . . . .	HTML 619
Session 8.2 Quick Check . . . . .	HTML 621

### **SESSION 8.3 . . . . .HTML 622**

Creating Transitions with CSS . . . . .	HTML 624
Introducing Transitions . . . . .	HTML 624

Setting the Transition Timing . . . . .	HTML 626
Delaying a Transition . . . . .	HTML 629
Creating a Hover Transition . . . . .	HTML 629
Animating Objects with CSS . . . . .	HTML 634
The <code>@keyframes</code> Rule . . . . .	HTML 634
Applying an Animation . . . . .	HTML 637
Controlling an Animation . . . . .	HTML 640
Session 8.3 Quick Check . . . . .	HTML 651
Review Assignments . . . . .	HTML 652
Case Problems . . . . .	HTML 655

## **Tutorial 9 Getting Started with JavaScript**

*Creating a Countdown Clock . . . . .* **HTML 665**

### **SESSION 9.1 . . . . .HTML 666**

Introducing JavaScript . . . . .	HTML 668
Server-Side and Client-Side Programming . . . . .	HTML 668
The Development of JavaScript . . . . .	HTML 669
Working with the <code>script</code> Element . . . . .	HTML 670
Loading the <code>script</code> Element . . . . .	HTML 670
Inserting the <code>script</code> Element . . . . .	HTML 671
Creating a JavaScript Program . . . . .	HTML 673
Adding Comments to your JavaScript Code . . . . .	HTML 673
Writing a JavaScript Command . . . . .	HTML 674
Understanding JavaScript Syntax . . . . .	HTML 675
Debugging your Code . . . . .	HTML 677
Opening a Debugger . . . . .	HTML 677
Inserting a Breakpoint . . . . .	HTML 679
Applying Strict Usage of JavaScript . . . . .	HTML 680
Session 9.1 Quick Check . . . . .	HTML 681

### **SESSION 9.2 . . . . .HTML 682**

Introducing Objects . . . . .	HTML 684
Object References . . . . .	HTML 685
Referencing Object Collections . . . . .	HTML 685
Referencing an Object by ID and Name . . . . .	HTML 687
Changing Properties and Applying Methods . . . . .	HTML 688
Object Properties . . . . .	HTML 688
Applying a Method . . . . .	HTML 688
Writing HTML Code . . . . .	HTML 689
Working with Variables . . . . .	HTML 693
Declaring a Variable . . . . .	HTML 693
Variables and Data Types . . . . .	HTML 694
Using a Variable . . . . .	HTML 695



Working with Date Objects . . . . .	HTML 695
Creating a Date Object . . . . .	HTML 696
Applying Date Methods . . . . .	HTML 697
Setting Date and Time Values . . . . .	HTML 700
Session 9.2 Quick Check . . . . .	HTML 701

### **SESSION 9.3 . . . . .HTML 702**

Working with Operators and Operands . . . . .	HTML 704
Using Assignment Operators . . . . .	HTML 704
Calculating the Days Left in the Year . . . . .	HTML 705
Working with the Math Object . . . . .	HTML 707
Using Math Methods . . . . .	HTML 707
Using Math Constants . . . . .	HTML 712
Working with JavaScript Functions . . . . .	HTML 714
Calling a Function . . . . .	HTML 716
Creating a Function to Return a Value . . . . .	HTML 717
Running Timed Commands . . . . .	HTML 718
Working with Time-Delayed Commands . . . . .	HTML 718
Running Commands at Specified Intervals . . . . .	HTML 718
Controlling How JavaScript Works with Numeric Values . . . . .	HTML 720
Handling Illegal Operations . . . . .	HTML 720
Defining a Number Format . . . . .	HTML 721
Converting Between Numbers and Text . . . . .	HTML 721
Session 9.3 Quick Check . . . . .	HTML 723
Review Assignments . . . . .	HTML 724
Case Problems . . . . .	HTML 726

## **Tutorial 10 Exploring Arrays, Loops, and Conditional Statements**

<i>Creating a Monthly Calendar . . . . .</i>	<b>HTML 735</b>
--	-----------------

### **SESSION 10.1 . . . . .HTML 736**

Introducing the Monthly Calendar . . . . .	HTML 738
Reviewing the Calendar Structure . . . . .	HTML 739
Adding the calendar() Function . . . . .	HTML 740
Introducing Arrays . . . . .	HTML 741
Creating and Populating an Array . . . . .	HTML 742
Working with Array Length . . . . .	HTML 745
Reversing an Array . . . . .	HTML 747
Sorting an Array . . . . .	HTML 748
Extracting and Inserting Array Items . . . . .	HTML 749
Using Arrays as Data Stacks . . . . .	HTML 750
Session 10.1 Quick Check . . . . .	HTML 753

### **SESSION 10.2 . . . . .HTML 754**

Working with Program Loops . . . . .	HTML 756
Exploring the for Loop . . . . .	HTML 756
Exploring the while Loop . . . . .	HTML 758
Exploring the do/while Loop . . . . .	HTML 759
Comparison and Logical Operators . . . . .	HTML 760
Program Loops and Arrays . . . . .	HTML 761
Array Methods to Loop Through Arrays . . . . .	HTML 764
Running a Function for Each Array Item . . . . .	HTML 765
Mapping an Array . . . . .	HTML 765
Filtering an Array . . . . .	HTML 766
Session 10.2 Quick Check . . . . .	HTML 769

### **SESSION 10.3 . . . . .HTML 770**

Introducing Conditional Statements . . . . .	HTML 772
Exploring the if Statement . . . . .	HTML 773
Nesting if Statements . . . . .	HTML 775
Exploring the if else Statement . . . . .	HTML 777
Using Multiple else if Statements . . . . .	HTML 778
Completing the Calendar App . . . . .	HTML 780
Setting the First Day of the Month . . . . .	HTML 781
Placing the First Day of the Month . . . . .	HTML 782
Writing the Calendar Days . . . . .	HTML 783
Highlighting the Current Date . . . . .	HTML 785
Displaying Daily Events . . . . .	HTML 787
Managing Program Loops and Conditional Statements . . . . .	HTML 790
Exploring the break Command . . . . .	HTML 790
Exploring the continue Command . . . . .	HTML 790
Exploring Statement Labels . . . . .	HTML 791
Session 10.3 Quick Check . . . . .	HTML 793
Review Assignments . . . . .	HTML 794
Case Problems . . . . .	HTML 796

## **Tutorial 11 Working with Events and Styles**

<i>Designing an Interactive Puzzle . . . . .</i>	<b>HTML 809</b>
--	-----------------

### **SESSION 11.1 . . . . .HTML 810**

Introducing JavaScript Events . . . . .	HTML 812
Creating an Event Handler . . . . .	HTML 815
Using the Event Object . . . . .	HTML 819
Exploring Object Properties . . . . .	HTML 823
Object Properties and Inline Styles . . . . .	HTML 824
Creating Object Collections with CSS Selectors . . . . .	HTML 826
Session 11.1 Quick Check . . . . .	HTML 829

**SESSION 11.2 . . . . .HTML 830**

Working with Mouse Events . . . . .	HTML 832
Introducing the Event Model . . . . .	HTML 836
Adding an Event Listener . . . . .	HTML 837
Removing an Event Listener . . . . .	HTML 839
Controlling Event Propagation . . . . .	HTML 841
Exploring Keyboard Events . . . . .	HTML 843
Changing the Cursor Style . . . . .	HTML 847
Session 11.2 Quick Check . . . . .	HTML 853

**SESSION 11.3 . . . . .HTML 854**

Working with Functions as Objects . . . . .	HTML 856
Function Declarations and Function Operators . . . . .	HTML 856
Anonymous Functions . . . . .	HTML 856
Passing Variable Values into Anonymous Functions . . . . .	HTML 859
Displaying Dialog Boxes . . . . .	HTML 865
Session 11.3 Quick Check . . . . .	HTML 872
Review Assignments . . . . .	HTML 873
Case Problems . . . . .	HTML 877

**Tutorial 12 Working with Document Nodes and Style Sheets***Creating a Dynamic Document Outline . . . . .* **HTML 891****SESSION 12.1 . . . . .HTML 892**

Introducing Nodes . . . . .	HTML 894
Nodes and Document Structure . . . . .	HTML 894
Restructuring the Node Tree . . . . .	HTML 897
Creating and Appending Nodes . . . . .	HTML 900
Working with Node Types, Names, and Values . . . . .	HTML 905
Looping through the Child Nodes Collection . . . . .	HTML 905
Node Properties . . . . .	HTML 907
Session 12.1 Quick Check . . . . .	HTML 913

**SESSION 12.2 . . . . .HTML 914**

Creating a Nested List . . . . .	HTML 916
Working with Attribute Nodes . . . . .	HTML 925
Creating Heading IDs . . . . .	HTML 927
Creating Hypertext Links . . . . .	HTML 930
Session 12.2 Quick Check . . . . .	HTML 935

**SESSION 12.3 . . . . .HTML 936**

Working with Style Sheets . . . . .	HTML 938
The styleSheets Collection . . . . .	HTML 938
The styleSheet Object . . . . .	HTML 938

Working with Style Sheet Rules . . . . .	HTML 944
Style Rule Objects . . . . .	HTML 945
Adding and Removing Style Rules . . . . .	HTML 946
Exploring Calculated Styles . . . . .	HTML 951
Session 12.3 Quick Check . . . . .	HTML 953
Review Assignments . . . . .	HTML 954
Case Problems . . . . .	HTML 957

**Tutorial 13 Programming for Web Forms***Creating Forms for Orders and Payments . . . . .* **HTML 969****SESSION 13.1 . . . . .HTML 970**

Exploring the Forms Object . . . . .	HTML 972
Working with Form Elements . . . . .	HTML 974
Working with Input Fields . . . . .	HTML 975
Setting the Field Value . . . . .	HTML 975
Navigating between Fields . . . . .	HTML 977
Working with Selection Lists . . . . .	HTML 978
Working with Options Buttons and Check Boxes . . . . .	HTML 982
Working with Check Boxes . . . . .	HTML 984
Formatting Numeric Values . . . . .	HTML 986
Applying Form Events . . . . .	HTML 991
Working with Hidden Fields . . . . .	HTML 993
Session 13.1 Quick Check . . . . .	HTML 995

**SESSION 13.2 . . . . .HTML 996**

Sharing Data between Forms . . . . .	HTML 998
Appending Form Data . . . . .	HTML 998
Examining the location Object . . . . .	HTML 1000
Working with Text Strings . . . . .	HTML 1001
Extracting Substrings from a Text String . . . . .	HTML 1002
Searching within a Text String . . . . .	HTML 1003
Introducing Regular Expressions . . . . .	HTML 1006
Matching a Substring . . . . .	HTML 1006
Setting Regular Expression Flags . . . . .	HTML 1008
Defining Character Types and Character Classes . . . . .	HTML 1008
Specifying Repeating Characters . . . . .	HTML 1012
Using Escape Sequences . . . . .	HTML 1013
Specifying Alternate Patterns and Grouping . . . . .	HTML 1014
Programming with Regular Expressions . . . . .	HTML 1016
Regular Expression Methods . . . . .	HTML 1017
Replacing URI Encoded Characters . . . . .	HTML 1020
Writing URL Data to a Web Form . . . . .	HTML 1021
Session 13.2 Quick Check . . . . .	HTML 1025

**SESSION 13.3 . . . . .HTML 1026**

Validating Data with JavaScript . . . . .	HTML 1028
Introducing the Constraint Validation API. . . . .	HTML 1029
Exploring the ValidityState Object. . . . .	HTML 1030
Creating a Custom Validation Message . . . . .	HTML 1030
Responding to Invalid Data . . . . .	HTML 1032
Validating Data with Pattern Matching . . . . .	HTML 1035
Validating a Selection List . . . . .	HTML 1037
Testing a Form Field Against a Regular Expression . . . . .	HTML 1039
Testing for Legitimate Card Numbers . . . . .	HTML 1041
Session 13.3 Quick Check . . . . .	HTML 1046
Review Assignments . . . . .	HTML 1047
Case Problems. . . . .	HTML 1049

**Tutorial 14 Exploring Object-Based Programming**

<i>Designing an Online Poker Game . . . . .</i>	<b>HTML 1061</b>
---	------------------

**SESSION 14.1 . . . . .HTML 1062**

Working with Nested Functions . . . . .	HTML 1064
Introducing Custom Objects. . . . .	HTML 1070
Object Literals . . . . .	HTML 1071
Dot Operators and Bracket Notation . . . . .	HTML 1072
Creating a Custom Method . . . . .	HTML 1075
Session 14.1 Quick Check. . . . .	HTML 1079

**SESSION 14.2 . . . . .HTML 1080**

Defining an Object Type . . . . .	HTML 1082
Creating an Object with the new Operator . . . . .	HTML 1082
Constructor Functions . . . . .	HTML 1082
Combining Object Classes . . . . .	HTML 1085
Working with Object Prototypes. . . . .	HTML 1095
Defining a Prototype Method. . . . .	HTML 1096
Session 14.2 Quick Check . . . . .	HTML 1105

**SESSION 14.3 . . . . .HTML 1106**

Combining Objects . . . . .	HTML 1108
Creating a Prototype Chain . . . . .	HTML 1108
The Base Object . . . . .	HTML 1109
Using the apply ( ) and call ( ) Methods . . . . .	HTML 1110
Combining Objects and Arrays . . . . .	HTML 1113
Applying the every ( ) Array method . . . . .	HTML 1114
Creating an Object Literal with the forEach ( ) Method . . . . .	HTML 1117
Applying a for...in loop . . . . .	HTML 1119
Session 14.3 Quick Check . . . . .	HTML 1128
Review Assignments . . . . .	HTML 1129
Case Problems. . . . .	HTML 1132

**Appendix A Color Names with Color Values, and HTML Character Entities . . . . .HTML A1****Appendix B HTML Elements and Attributes . . . .HTML B1****Appendix C Cascading Styles and Selectors . . .HTML C1****Appendix D Making the Web****More Accessible . . . . .HTML D1****Appendix E Designing for the Web. . . . .HTML E1****Appendix F Page Validation with XHTML. . . . .HTML F1****GLOSSARY . . . . . REF 1****INDEX . . . . .REF 13**

## OBJECTIVES

**Session 1.1**

- Explore the history of the web
- Create the structure of an HTML document
- Insert HTML elements and attributes
- Insert metadata into a document
- Define a page title

**Session 1.2**

- Mark page structures with sectioning elements
- Organize page content with grouping elements
- Mark content with text-level elements
- Insert inline images
- Insert symbols based on character codes

**Session 1.3**

- Mark content using lists
- Create a navigation list
- Link to files within a website with hypertext links
- Link to e-mail addresses and telephone numbers

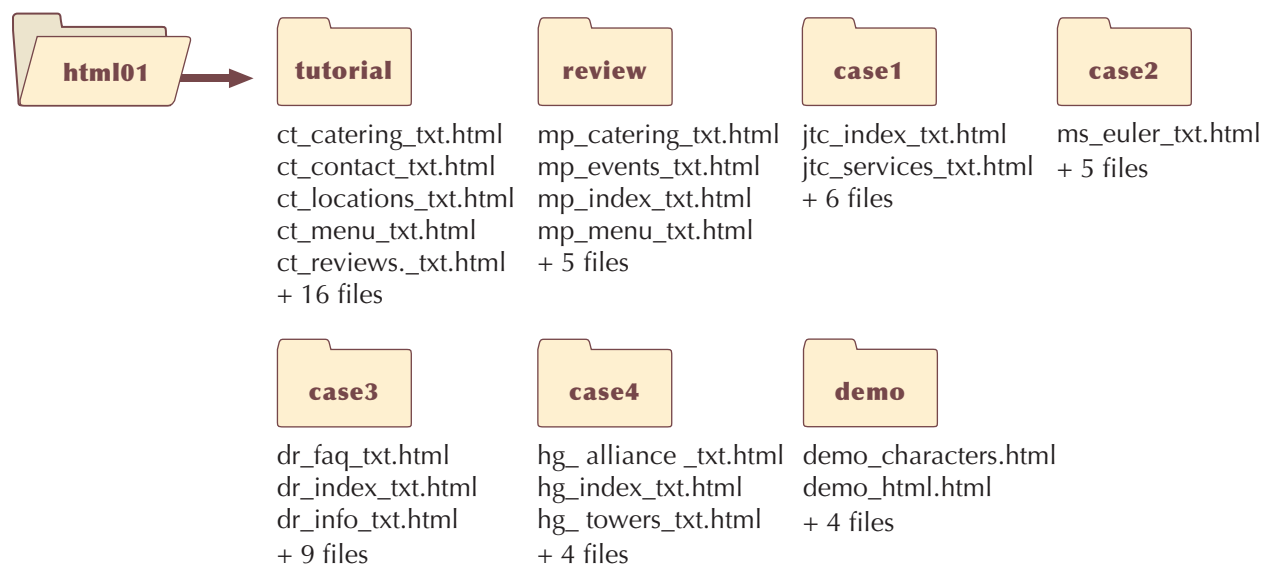
# Getting Started with HTML5

## *Creating a Website for a Food Vendor*

### Case | *Curbside Thai*

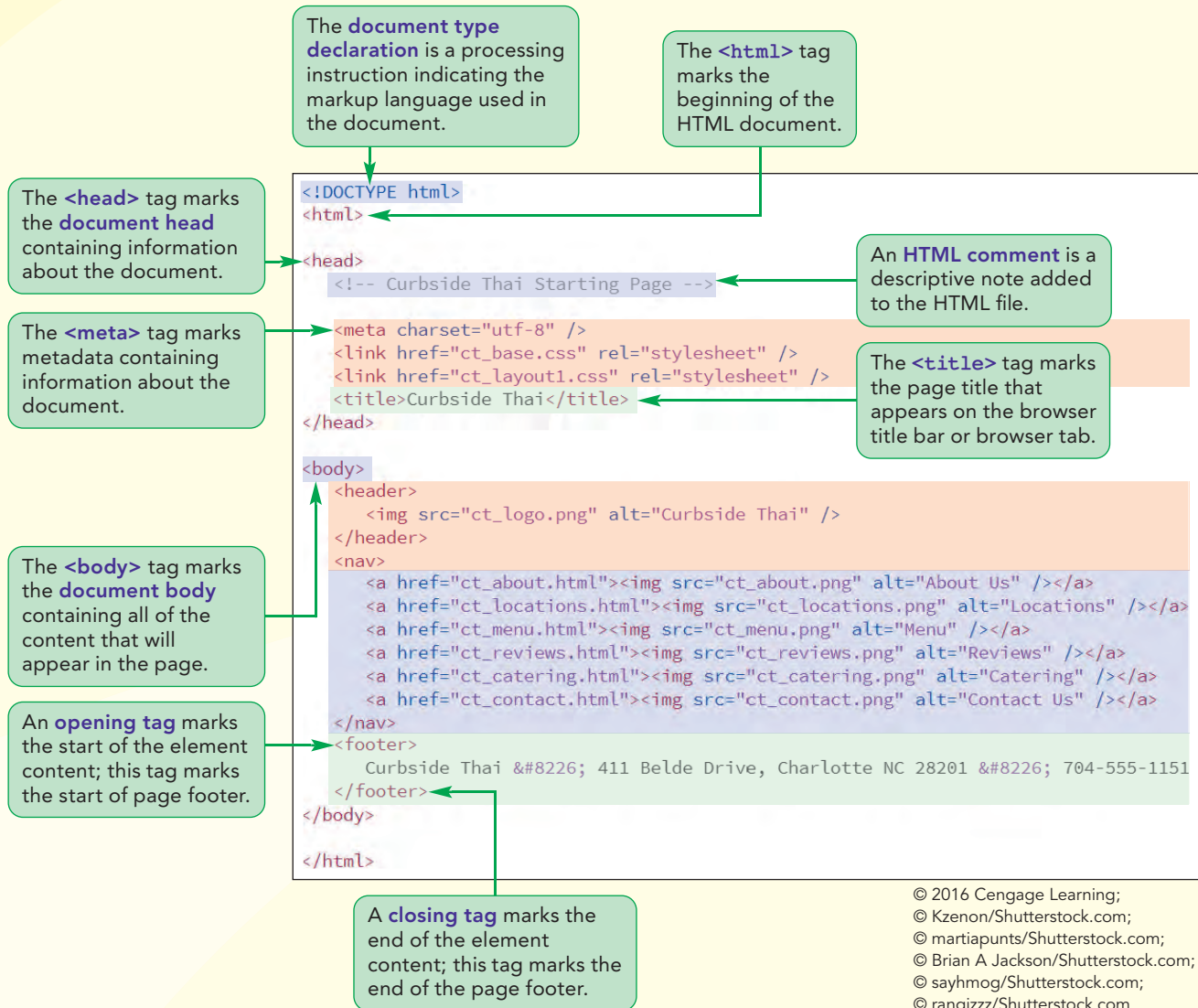
Sajja Adulet is the owner and master chef of Curbside Thai, a restaurant owner and now food truck vendor in Charlotte, North Carolina that specializes in Thai dishes. Sajja has hired you to develop the company's website. The website will display information about Curbside Thai including the truck's daily locations, menu, catering opportunities, and contact information. Sajja wants the pages to convey the message that customers will get the same great food and service whether they order in the restaurant or from the food truck. Some of the materials for these pages have already been completed by a former employee and Sajja needs you to finish the job by converting that work into a collection of web page documents. To complete this task, you'll learn how to write and edit HTML5 code and how to get your HTML files ready for display on the World Wide Web.

### STARTING DATA FILES





# Session 1.1 Visual Overview:



© 2016 Cengage Learning;  
© Kzenon/Shutterstock.com;  
© martiapunts/Shutterstock.com;  
© Brian A Jackson/Shutterstock.com;  
© sayhmog/Shutterstock.com;  
© rangizzz/Shutterstock.com

# The Structure of an HTML Document

Document as it appears in the browser.



The exact layout of the document elements is determined by a style sheet and not by the document markup.

## Exploring the World Wide Web

It is no exaggeration to say that the World Wide Web has had as profound an effect on human communication as the printing press. One key difference is that operation of the printing press was limited to a few select tradesmen but on the web everyone has his or her own printing press; everyone can be a publisher of a website. Before creating your first website, you'll examine a short history of the web because that history impacts the way you write code for your web pages. You'll start by exploring the basic terminology of computer networks.

### Networks

A **network** is a structure in which information and services are shared among devices known as **nodes** or **hosts**. A host can be any device that is capable of sending and/or receiving data electronically. The most common hosts that you will work with are desktop computers, laptops, tablets, mobile phones, and printers.

A host that provides information or a service to other devices on the network is called a **server**. For example, a print server is a network host that provides printing services and a file server is a host that provides storage space for saving and retrieving files. The device that receives these services is called a **client**. A common network design is the **client-server network**, in which the clients access information provided by one or more servers. You might be using such a network to access your data files for this tutorial.

Networks are classified based on the range of devices they cover. A network confined to a small geographic area, such as within a building or department, is referred to as a **local area network** or **LAN**. A network that covers a wider area, such as several buildings or cities, is called a **wide area network** or **WAN**. Wide area networks typically consist of two or more interconnected local area networks. The largest WAN in existence is the **Internet**, which incorporates an almost uncountable number of networks and hosts involving computers, mobile devices (such as phones, tablets, and so forth), MP3 players, and gaming systems.

### Locating Information on a Network

The biggest obstacle to effectively using the Internet is the network's sheer scope and size. Most of the early Internet tools required users to master a bewildering array of terms, acronyms, and commands. Because network users had to be well versed in computers and network technology, Internet use was largely limited to programmers and computer specialists working for universities, large businesses, and the government.

The solution to this problem was developed in 1989 by Timothy Berners-Lee and other researchers at the CERN nuclear research facility near Geneva, Switzerland. They needed an information system that would make it easy for their researchers to locate and share data on the CERN network. To meet this need, they developed a system of hypertext documents. **Hypertext** is a method of organization in which data sources are interconnected through a series of links or **hyperlinks** that users activate to jump from one data source to another. Hypertext is ideally suited for the Internet because end users don't need to know where a particular document, information source, or service is located—they only need to know how to activate the link. The effectiveness of this technique quickly spread beyond Geneva and was adopted with other networks across the Internet. The totality of these interconnected hypertext documents became known as the **World Wide Web**. The fact that the Internet and the World Wide Web are synonymous in many users' minds is a testament to the success of the hypertext approach.

## Web Pages and Web Servers

Documents on the web are stored on **web servers** in the form of **web pages** and accessed through a software program called a **web browser**. The browser retrieves the document from the web server and renders it locally in a form that is readable on a client device. However, because there is a wide selection of client devices ranging from desktop computers to mobile phones to screen readers that relay data aurally, each web page must be written in code that is compatible with every device. How does the same document work with so many different devices? To understand, you need to look at how web pages are created.

## Introducing HTML

A web page is a simple text file written in **HTML (Hypertext Markup Language)**. You've already read about hypertext, but what is a markup language? A **markup language** is a language that describes the content and structure of a document by "marking up" or tagging, different document elements. For example, this tutorial contains several document elements such as the tutorial title, main headings, subheadings, paragraphs, figures, figure captions, and so forth. Using a markup language, each of these elements could be tagged as a distinct item within the "tutorial document." Thus, a Hypertext Markup Language is a language that supports both the tagging of distinct document elements and connecting documents through hypertext links.

## The History of HTML

In the early years, no single organization defined the rules or **syntax** of HTML. Browser developers were free to define and modify the language in different ways which, of course, led to problems as different browsers supported different "flavors" of HTML and a web page that was written based on one browser's standard might appear totally different when rendered by another browser. Ultimately, a group of web designers and programmers called the **World Wide Web Consortium**, or the **W3C**, settled on a set of standards or specifications for all browser manufacturers to follow. The W3C has no enforcement power, but, because using a uniform language is in everyone's best interest, the W3C's recommendations are usually followed, though not always immediately. Each new version of HTML goes through years of discussion and testing before it is formally adopted as the accepted standard. For more information on the W3C and its services, see its website at [www.w3.org](http://www.w3.org).

By 1999, HTML had progressed to the fourth version of the language, **HTML 4.01**, which provided support for multimedia, online commerce, and interactive scripts running within the web page. However, there were still many incompatibilities in how HTML was implemented across different browsers and how HTML code was written by web developers. The W3C sought to take control of what had been a haphazard process and enforce a stricter set of standards in a different version of the language called **XHTML (Extensible Hypertext Markup Language)**. By 2002, the W3C had released the specifications for XHTML 1.1. But XHTML 1.1 was intended to be only a minor upgrade on the way to XHTML 2.0, which would correct many of the deficiencies found in HTML 4.01 and become the future language of the web. One problem was that XHTML 2.0 would not be backward compatible with HTML and, as a result, older websites could not be easily brought into the new standard.

Web designers rebelled at this development and, in response, the **Web Hypertext Application Technology Working Group (WHATWG)** was formed in 2004 with the mission to develop a rival version to XHTML 2.0, called **HTML5**. Unlike XHTML 2.0, HTML5 would be compatible with earlier versions of HTML and would not apply the same strict standards that XHTML demanded. For several years, it was unclear which specification would win out; but by 2006, work on XHTML 2.0 had completely stalled



**TIP**

You can find out which browsers and browser versions support the features of HTML5 by going to the website [caniuse.com](http://caniuse.com).

and the W3C issued a new charter for WHATWG to develop HTML5 as the de facto standard for the next generation of HTML. Thus today, HTML5 is the current version of the HTML language and it is supported by all current browsers and devices. You can learn more about WHATWG and its current projects at [www.whatwg.org](http://www.whatwg.org).

As HTML has evolved, features and code found in earlier versions of the language are often **deprecated**, or phased out, and while deprecated features might not be part of HTML5, that doesn't mean that you won't encounter them in your work—indeed, if you are maintaining older websites, you will often need to interpret code from earlier versions of HTML. Moreover, there are still many older browsers and devices in active use that do not support HTML5. Thus, a major challenge for website designers is writing code that takes advantage of HTML5 but is still accessible to older technology.

Figure 1-1 summarizes some of the different versions of HTML that have been implemented over the years. You can read detailed specifications for these versions at the W3C website.

**Figure 1-1****HTML version history**

Version	Date	Description
HTML 1.0	1989	The first public version of HTML
HTML 2.0	1995	HTML version that added interactive elements including web forms
HTML 3.2	1997	HTML version that provided additional support for web tables and expanded the options for interactive form elements and a scripting language
HTML 4.01	1999	HTML version that added support for style sheets to give web designers greater control over page layout and appearance, and provided support for multimedia elements such as audio and video
XHTML 1.0	2001	A reformulation of HTML 4.01 using the XML markup language in order to provide enforceable standards for HTML content and to allow HTML to interact with other XML languages
XHTML 2.0	discontinued in 2009	The follow-up version to XHTML 1.1 designed to fix some of the problems inherent in HTML 4.01 syntax
HTML 5.0	2012	The current HTML version providing support for mobile design, semantic page elements, column layout, form validation, offline storage, and enhanced multimedia

© 2016 Cengage Learning

This book focuses on HTML5, but you will also review some of the specifications for HTML 4.01 and XHTML 1.1. Note that in the figures that follow, code that was introduced starting with HTML5 will be identified with the label **[HTML5]**.

## Tools for Working with HTML

Because HTML documents are simple text files, the first tool you will need is a text editor. You can use a basic text editor such as Windows Notepad or TextEdit for the Macintosh, but it is highly recommended that you use one of the many inexpensive editors that provide built-in support for HTML. Some of the more popular HTML editors are Notepad++ ([notepad-plus-plus.org](http://notepad-plus-plus.org)), UltraEdit ([www.ultraedit.com](http://www.ultraedit.com)), CoffeeCup ([www.coffeecup.com](http://www.coffeecup.com)), BBEdit ([www.barebones.com](http://www.barebones.com)) and ConTEXT ([www.contexteditor.org](http://www.contexteditor.org)). These editors include such features as syntax checking to weed out errors, automatic insertion of HTML code, and predesigned templates with the initial code already prepared for you.

These enhanced editors are a good way to start learning HTML and they will be all you need for most basic projects, but professional web developers working on large websites will quickly gravitate toward using a web **IDE (Integrated Development Environment)**, which is a software package providing comprehensive coverage of all phases of the development process from writing HTML code to creating scripts for programs running on web servers. Some of the popular IDEs for web development include Adobe Dreamweaver ([www.adobe.com](http://www.adobe.com)), Aptana Studio ([www.aptana.com](http://www.aptana.com)), NetBeans IDE ([netbeans.org](http://netbeans.org)) and Komodo IDE ([komodoide.com](http://komodoide.com)). Web IDEs can be very expensive, but most software companies will provide a free evaluation period for you to test their product to see if it meets your needs.

## Testing your Code

### TIP

You can analyze each browser for its compatibility with HTML5 at the website [www.html5test.com](http://www.html5test.com).

Once you've written your code, you can test whether your HTML code employs proper syntax and structure by validating it at the W3C validation website ([validator.w3.org](http://validator.w3.org)). **Validators**, like the one available through the W3C website, are programs that test code to ensure that it contains no syntax errors. The W3C validator will highlight all of the syntax errors in your document with suggestions about how to fix those errors.

Finally, you'll need to test it to ensure that your content is rendered correctly. You should test your code under a variety of screen resolutions, on several different browsers and, if possible, on different versions of the same browser because users are not always quick to upgrade their browsers. What may look good on a widescreen monitor might look horrible on a mobile phone. At a minimum you should test your website using the following popular browsers: Google Chrome, Internet Explorer, Apple Safari, Mozilla Firefox, and Opera.

It is not always possible to load multiple versions of the same browser on one computer, so, in order to test a website against multiple browser versions, professional designers will upload their code to online testing services that report on the website's compatibility across a wide range of browsers, screen resolutions, and devices, including both desktop and mobile devices. Among the popular testing services are BrowserStack ([www.browserstack.com](http://www.browserstack.com)), CrossBrowserTesting ([www.crossbrowsertesting.com](http://www.crossbrowsertesting.com)), and Browsera ([www.browsersa.com](http://www.browsersa.com)). Most of these sites charge a monthly connection fee with a limited number of testing minutes, so you should not upload your code until you are past the initial stages of development.

## Supporting the Mobile Web

Currently, the most important factor impacting website design is the increased use of mobile devices to access the Internet. By the end of 2014, the number of mobile Internet users exceeded the number of users accessing the web through laptop or desktop devices. The increased reliance on mobile devices means that web designers must be careful to tailor their websites to accommodate both the desktop and mobile experience. You'll explore the challenge of designing for the mobile web in more detail in Tutorial 5.

## Exploring an HTML Document

Now that you have reviewed the history of the web and some of the challenges in developing your own website, you will look at the code of an actual HTML file. To get you started, Sajja Adulet has provided you with the `ct_start.html` file containing the code for the initial page users see when they access the Curbside Thai website. Open Sajja's file now.

**TIP**

All HTML files have the file extension .html or .htm.

**To open the ct\_start.html file:**

1. Use the editor of your choice to open the **ct\_start.html** file from the html01 ► tutorial folder.

Figure 1-2 shows the complete contents of the file as viewed in the Notepad++ editor.

**Figure 1-2****Elements and attributes from an HTML document**

```

<!DOCTYPE html>
<html>

  <head>
    <title>Curbside Thai</title>

    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link href="ct_base.css" rel="stylesheet" type="text/css" />
    <link href="ct_layout1.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <header>
      
    </header>

    <nav>
      <a href="ct_about.html"></a>
      <a href="ct_locations.html"></a>
      <a href="ct_menu.html"></a>
      <a href="ct_reviews.html"></a>
      <a href="ct_catering.html"></a>
      <a href="ct_contact.html"></a>
    </nav>

    <footer>
      Curbside Thai ☎8226; 411 Belde Drive, Charlotte NC 28201 ☎8226; 704-555-1151
    </footer>
  </body>
</html>

```

**Trouble?** Depending on your editor and its configuration, the text style applied to your code might not match that shown in Figure 1-2. This is not a problem. Because HTML documents are simple text files, any text styles are a feature of the editor and have no impact on how the document is rendered by the browser.

2. Scroll through the document to become familiar with its content but do not make any changes to the text.

## The Document Type Declaration

The first line in an HTML file is the document type declaration or doctype, which is a processing instruction indicating the markup language used in the document. The browser uses the document type declaration to know which standard to use to display the content. For HTML5, the doctype is entered as

```
<!DOCTYPE html>
```

You might also see the doctype entered in lowercase letters as

```
<!doctype html>
```

Both are accepted by all browsers. Older versions of HTML had more complicated doctypes. For example, the doctype for HTML 4.01 is the rather foreboding

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

You might even come across older HTML files that do not have a doctype. Because early versions of HTML did not require a doctype, many browsers interpret the absence of the doctype as a signal that the page should be rendered in **quirks mode**, based on styles and practices from the 1990s and early 2000s. When the doctype is present, browsers will render the page in **standards mode**, employing the most current specifications of HTML. The difference between quirks mode and standards mode can mean the difference between a nicely laid-out page and a confusing mess, so, as a result, you should always put your HTML5 file in standards mode by including the doctype.

## Introducing Element Tags

The fundamental building block in every HTML document is the **element tag**, which marks an element in the document. A **starting tag** indicates the beginning of that element, while an **ending tag** indicates the ending. The general syntax of a two-sided element tag is

```
<element>content</element>
```

where *element* is the name of the element, *content* is the element's content, *<element>* is the starting tag, and *</element>* is the ending tag. For example, the following code marks a paragraph element:

```
<p>Welcome to Curbside Thai.</p>
```

Here the *<p></p>* tags are the starting and ending HTML tags that indicate the presence of a paragraph and the text *Welcome to Curbside Thai.* comprises the paragraph text.

Not every element tag encloses document content. **Empty elements** are elements that are either nontextual (such as images) or contain directives to the browser about how the page should be treated. An empty element is entered using one of the following forms of the **one-sided element tag**:

```
<element />
```

or

```
<element>
```

For example, the following **br** element, which is used to indicate the presence of a line break in the text, is entered with the one-sided tag:

```
<br />
```

Note that, while this code could also be entered as *<br>*, the ending slash */>* form is the required form in XHTML documents as well as other markup languages. While HTML5 allows for either form, it's a good idea to get accustomed to using the ending slash */>* form if you intend to work with other markup languages in the future. We'll follow the */>* convention in the code in this book.

Elements can contain other elements, which are called **nested elements**. For example, in the following code, the **em** element (used to mark emphasized text) is nested within the paragraph element by placing the **em** markup tag completely within the **p** markup tag.

**Proper syntax:**

```
<p>Welcome to <em>Curbside Thai</em>.</p>
```

Note that when nesting one element inside of another, the entire code of the inner element must be contained within the outer element, including opening and closing tags. Thus, it would not be correct syntax to place the closing tag for the `em` element outside of the `p` element as in the following code:

**Improper syntax:**

```
<p>Welcome to <em>Curbside Thai</p>.</em>
```

Now that you've examined the basics of tags, you'll look at how they're used within an HTML file.

## The Element Hierarchy

The entire structure of an HTML document can be thought of as a set of nested elements in a hierarchical tree. At the top of the tree is the `html` element, which marks the entire document. Within the `html` element is the `head` element used to mark information about the document itself and the `body` element used to mark the content that will appear in the web page. Thus, the general structure of an HTML file, like the one shown in Figure 1-2, is

```
<!DOCTYPE html>
<html>
  <head>
    head content
  </head>

  <body>
    body content
  </body>
</html>
```

where *head content* and *body content* are nested elements that mark the content of the document head and body. Note that the `body` element is always placed after the `head` element.

**REFERENCE**

### Creating the Basic Structure of an HTML File

- To create the basic structure of an HTML file, enter the tags

```
<!DOCTYPE html>
<html>
  <head>
    head content
  </head>

  <body>
    body content
  </body>
</html>
```

where *head*, *content*, and *body content* contain nested elements that mark the content of the head and body sections.



**TIP**

Attributes can be listed in any order but they must come after the element name and be separated from each other by a blank space; each attribute value must be enclosed within single or double quotation marks.

## Introducing Element Attributes

Elements will often contain one or more **element attributes**. Each attribute provides additional information to the browser about the purpose of the element or how the element should be handled by the browser. The general syntax of an element attribute within a two-sided tag is

```
<element attr1="value1" attr2="value2" ...>
    content
</element>
```

Or, for a one-sided tag

```
<element attr1="value1" attr2="value2" ... />
```

where *attr1*, *attr2*, and so forth are attributes associated with *element* and *value1*, *value2*, and so forth are the corresponding attribute values. For example, the following code adds the `id` attribute with the value "intro" to the `<p>` tag in order to identify the paragraph as an introductory paragraph.

```
<p id="intro">Welcome to Curbside Thai.</p>
```

HTML editors will often color-code attributes and their values. The attributes in Figure 1-2 are rendered in a blue font while the corresponding attribute values are rendered in magenta.

Each element has its own set of attributes but, in addition to these element-specific attributes, there is a core set of attributes that can be applied to almost every HTML element. Figure 1-3 lists some of the most commonly used core attributes; others are listed in Appendix B.

**Figure 1-3**

**Commonly used core HTML attributes**

Attribute	Description
<code>class="text"</code>	Defines the general classification of the element
<code>dir="ltr rtl auto"</code>	Defines the text direction of the element content as left-to-right, right-to-left, or determined by the browser
<code>hidden</code>	Indicates that the element should be hidden or is no longer relevant <b>[HTML5]</b>
<code>id="text"</code>	Provides a unique identifier for the element
<code>lang="text"</code>	Specifies the language of the element content
<code>style="definition"</code>	Defines the style or appearance of the element content
<code>tabindex="integer"</code>	Specifies the tab order of the element (when the tab button is used to navigate the page)
<code>title="text"</code>	Assigns a title to the element content

© 2016 Cengage Learning

Some attributes do not require a value, so, as a result, HTML supports **attribute minimization** in which no value is shown in the document. For example, the `hidden` attribute used in the following code does not require a value, its mere presence indicates that the marked paragraph should be hidden in the rendered page.

```
<p hidden>Placeholder Text</p>
```

Attribute minimization is another example of how HTML5 differs from other markup languages such as XHTML in which minimization is not allowed and all attributes must have attribute values.

### Adding an Attribute to an Element

- To add an attribute to an element, enter

```
<element attr1="value1" attr2="value2" ...>
  content
</element>
```

where *attr1*, *attr2*, and so forth are HTML attributes associated with *element* and *value1*, *value2*, and so forth are the corresponding attribute values.

## Handling White Space

Because an HTML file is a text file, it is composed only of text characters and white-space characters. A **white-space character** is any empty or blank character such as a space, tab, or line break. When the browser reads an HTML file, it ignores the presence of white-space characters between element tags and makes no distinction between spaces, tabs, or line breaks. Thus, a browser will treat the following two pieces of code in exactly the same way:

```
<p>Welcome to <em>Curbside Thai</em>.</p>
```

and

```
<p>
  Welcome to <em>Curbside Thai</em>.
</p>
```

The browser will also collapse consecutive occurrences of white-space characters into a single occurrence. This means that the text of the paragraph in the following code is still treated as “Welcome to Curbside Thai” because the extra white spaces between “Curbside” and “Thai” are ignored by the browser.

```
<p>
  Welcome to <em>Curbside      Thai</em>.
</p>
```

The bottom line is that it doesn’t matter how you lay out your HTML code because the browser is only interested in the text content and not how that text is entered. This means you can make your file easier to read by indenting lines and by adding extra white-space characters to separate one code block from another. However, this also means that any formatting you do for the page text to make the code more readable, such as tabs or extra white spaces, is *not* transferred to the web page.

## Viewing an HTML File in a Browser

The structure of the HTML file shown in Figure 1-2 should now be a little clearer, even if you don’t yet know how to interpret the meaning and purpose of each of element and attribute. To see what this page looks like, open it within a web browser.

### To open the `ct_start.html` file in a web browser:

1. Open your web browser. You do not need to be connected to the Internet to view local files stored on your computer.

2. After your browser loads its home page, open the `ct_start.html` file from the `html01 ► tutorial` folder. Figure 1-4 shows the page as it appears on a mobile phone and on a tablet device. The two devices have different screen widths, which affects how the page is rendered.

Figure 1-4

The Curbside Thai starting page as rendered by a mobile and tablet device



© 2016 Cengage Learning; © Kzenon/Shutterstock.com;  
 © martiapunts/Shutterstock.com; © Brian A Jackson/Shutterstock.com;  
 © sayhmog/Shutterstock.com; © rangizz/Shutterstock.com;  
 BenBois/openclickart; Jmlevick/openclickart

**Trouble?** If you're not sure how to open a local file with your browser, check for an Open or Open File command under the browser's File menu. You can also open a file by double-clicking the file name from within Windows Explorer or Apple Finder.

3. Reduce the width of your browser window and note that when the width falls below a certain value (in this case 480 pixels), the layout automatically changes to a stacked row of images (as shown in the mobile device image in Figure 1-4) that are better suited to the narrower layout.
4. Increase the width of the browser window and confirm that the layout changes to a 2×3 grid of images (as shown in the tablet device image in Figure 1-4), which is a design more appropriate for the wider window.

Figure 1-4 illustrates an important principle: *HTML does not describe the document's appearance, it only describes the document's content and structure.* The same HTML document can be rendered completely differently between one device and another or between one screen size and another. The actual appearance of the document is determined by style sheets—a topic you'll explore later in this tutorial.

## Creating an HTML File

Now that you've studied the structure of an HTML file, you'll start creating your own documents for the Curbside Thai website. Sajja wants you to create a web page containing information about the restaurant. Start by inserting the doctype and the markup tags for the `html`, `head`, and `body` elements.

### TIP

HTML filenames should be entered in lowercase letters and have no blank spaces.

#### To begin writing the HTML file:

1. Using the editor of your choice, create a new blank HTML file in the `html01 ► tutorial` folder, saving the file as **ct\_about.html**.
2. Enter the following code into the file:

```
<!DOCTYPE html>
<html>

<head>
</head>

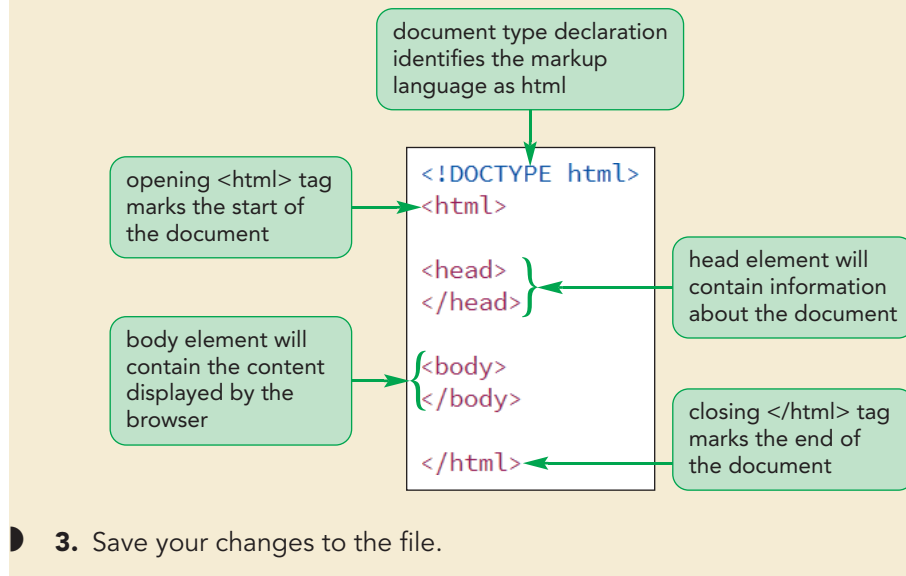
<body>
</body>

</html>
```

Figure 1-5 shows the initial elements in the document.

Figure 1-5

#### Initial structure of the `ct_about.html` file



Next, you'll add elements to the document head.



## PROSKILLS

### Written Communication: Writing Effective HTML Code

Part of writing good HTML code is being aware of the requirements of various browsers and devices, as well as understanding the different versions of the language. Here are a few guidelines for writing good HTML code:

- Become well versed in the history of HTML and the various versions of HTML and XHTML. Unlike other languages, HTML's history does impact how you write your code.
- Know your market. Do you have to support older browsers, or have your clients standardized on one particular browser or browser version? Will your web pages be viewed on a single device such as a computer, or do you have to support a variety of devices?
- Test your code on several different browsers and browser versions. Don't assume that if your page works in one browser, it will work in other browsers or even in earlier versions of the same browser. Also check on the speed of the connection. A large file that performs well with a high-speed connection might be unusable with a slower connection.
- Read the documentation on the different versions of HTML and XHTML at the W3C website and keep up to date with the latest developments in the language.

To effectively communicate with customers and users, you need to make sure your website content is always readable. Writing good HTML code is a great place to start.

## Creating the Document Head

The document head contains **metadata**, which is content that describes the document or provides information about how the document should be processed by the browser. Figure 1-6 describes the different metadata elements found in the document head.

Figure 1-6

HTML metadata elements

Element	Description
<code>head</code>	Contains a collection of metadata elements that describe the document or provide instructions to the browser
<code>base</code>	Specifies the document's location for use with resolving relative hypertext links
<code>link</code>	Specifies an external resource that the document is connected to
<code>meta</code>	Provides a generic list of metadata values such as search keywords, viewport properties, and the file's character encoding
<code>script</code>	Provides programming code for programs to be run within the document
<code>style</code>	Defines the display styles used to render the document content
<code>title</code>	Stores the document's title or name, usually displayed in the browser title bar or on a browser tab

© 2016 Cengage Learning

The first metadata you'll add to the About Curbside Thai web page is the `title` element.

### Setting the Page Title

The `title` element is part of the document head because it's not actually displayed as part of the web page, but rather appears externally within the browser title bar or browser tab. Page titles are defined using the following `title` element

```
<title>document title</title>
```

where `document title` is the text of the title. Add a page title to the Curbside Thai page now.



### Adding a Document Title

- To define the document title, enter the following tag into the document head:  

```
<title>document title</title>
```

 where *document title* is the text that will appear on the browser title bar or a browser tab.

#### TIP

Document titles should be no more than 64 characters in length to ensure that the text fits on the browser title bar or a browser tab.

### To insert the document title:

1. Directly after the opening `<head>` tag, insert the following `title` element, indented to make the code easier to read.

```
<title>About Curbside Thai</title>
```

Figure 1-7 highlights the code for the page title.

Figure 1-7

### Entering the document title

title text that appears in the browser title bar or on a browser tab

```
<!DOCTYPE html>
<html>

<head>
  <title>About Curbside Thai</title>
</head>
```

2. Save your changes to the file.

## Adding Metadata to the Document

Another metadata is the `meta` element, which is used for general lists of metadata values. The `meta` element structure is

```
<meta attributes />
```

where *attributes* define the type of metadata that is to be added to a document. Figure 1-8 lists the attributes of the `meta` element.

Figure 1-8

### Attributes of the meta element

Attribute	Description
<code>charset="encoding"</code>	Specifies the character encoding used in the HTML document <b>[HTML5]</b>
<code>content="text"</code>	Provides the value associated with the <code>http-equiv</code> or <code>name</code> attributes
<code>http-equiv="content-type default-style refresh"</code>	Provides an HTTP header for the document's content, default style, or refresh interval (in seconds)
<code>name="text"</code>	Sets the name associated with the metadata

© 2016 Cengage Learning

For example, you can use the following `meta` element to provide a collection of keywords for the Curbside Thai website that would aid web search engines, such as Google or Bing search tools, to locate the page for potential customers:

```
<meta name="keywords" content="Thai, restaurant, Charlotte, food" />
```

In this tag, the `name` attribute defines the type of metadata and the `content` attribute provides the data values. HTML does not specify a set of values for the `name` attribute, but commonly used names include `keywords`, `description`, `author`, and `viewport`.

Another use of the `meta` element is to define the character encoding used in the HTML file. **Character encoding** is the process by which the computer converts text into a sequence of bytes when it stores the text and then converts those bytes back into characters when the text is read. The most common character encoding in use is **UTF-8**, which supports almost all of the characters you will need. To indicate that the document is written using UTF-8, you add the following `meta` element to the document head:

```
<meta charset="utf-8" />
```

The `charset` attribute was introduced in HTML5 and replaces the following more complicated expression used in earlier versions of HTML:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

### TIP

The `title` element and the `charset` `meta` element are both required in a valid HTML5 document.

## Adding Metadata to the Document

### REFERENCE

- To define the character encoding used in the document, enter  

```
<meta charset="encoding" />
```

 where *encoding* is the character encoding used in the document.
- To define search keywords associated with the document, enter  

```
<meta name="keywords" content="terms" />
```

 where *terms* is a comma-separated list of keyword terms.

Add `meta` elements to the document head now, providing the character set and a list of keywords describing the page.

### TIP

The `<meta>` tag that defines the character encoding should always be the first `meta` element in the document head.

### To insert metadata:

1. Directly after the opening `<head>` tag, insert the following `meta` elements, indented to make the code easier to read:

```
<meta charset="utf-8" />
<meta name="keywords"
  content="Thai, restaurant, Charlotte, food" />
```

Figure 1-9 highlights the newly added `meta` elements used in the document head.

Figure 1-9

## Adding metadata to a document



2. Save your changes to the file.
3. Open the **ct\_about.html** file in your browser. Confirm that the browser tab or browser title bar contains the text “About Curbside Thai”. There should be no text displayed in the browser window because you have not added any content to the page body yet.

Before continuing with your edits to the `ct_about.html` file, you should document your work. You can do this with a comment.

## Adding Comments to your Document

A comment is descriptive text that is added to the HTML file but that does not appear in the browser window when the page is displayed. Comments can include the name of the document’s author, the date the document was created, and the purpose for which the document was created. Comments are added with the following markup:

```
<!-- comment -->
```

where *comment* is the text of the comment or note. For example, the following code inserts a comment describing the page you’re creating for Curbside Thai:

```
<!-- General Information about Curbside Thai -->
```

### TIP

Always include comments when working with a team so that you can document the development process for other team members.

A comment can be spread across several lines as long as the comment text begins with `<!--` and ends with `-->`. Because comments are ignored by the browser, they can be added anywhere within a document, though it’s good practice to always include a comment in the document head in order to describe the document content that follows.

### REFERENCE

#### Adding a Comment to an HTML Document

- To insert a comment anywhere within your HTML document, enter

```
<!-- comment -->
```

where *comment* is the text of the HTML comment.

Add comments to the `ct_about.html` file indicating the document’s author, date of creation, and purpose.

HTML comments must be closed with the --> characters.

### To add a comment to the document:

1. Return to the **ct\_about.html** file in your HTML editor.
2. Directly after the opening **<head>** tag, insert the following comment text, indented to make the code easier to read:

```
<!--  
  New Perspectives on HTML5 and CSS3, 7th Edition  
  Tutorial 1  
  Tutorial Case  
  General Information about Curbside Thai  
  Author: your name  
  Date:   the date  
  
  Filename: ct_about.html  
-->
```

where ***your name*** is your name and ***the date*** is the current date. Figure 1-10 highlights the newly added comment in the file.

Figure 1-10

### Adding a comment to the document

Comment added to the document

```
<head>  
  <!--  
    New Perspectives on HTML5 and CSS3, 7th Edition  
    Tutorial 1  
    Tutorial Case  
    General Information about Curbside Thai  
    Author: your name  
    Date:   the date  
  
    Filename: ct_about.html  
  -->  
  <meta charset="utf-8" />  
  <meta name="keywords" content="Thai, restaurant, Charlotte, food" />  
  <title>About Curbside Thai</title>  
</head>
```

3. Save your changes to the file.

### Conditional Comments and Internet Explorer

Another type of comment you will encounter in many HTML files is a **conditional comment**, which encloses content that should only be run by particular versions of the Internet Explorer browser. The general form of the conditional comment is

```
<!--[if operator IE version]>
  content
<![endif]-->
```

where *operator* is a logical operator (such as less than or greater than), *version* is the version number of an Internet Explorer browser, and *content* is the HTML code that will be run only if the conditional expression is true. The following code uses the *lt* (less than) logical operator to warn users that they need to upgrade their browser if they are running Internet Explorer prior to version 8.

```
<!--[if lt IE 8]>
  <p>Upgrade your browser to view this page.</p>
<![endif]-->
```

Other logical operators include *lte* (less than or equal to), *gt* (greater than), *gte* (greater than or equal to) and *!* (not). For example, the following code uses the logical operator *!* to display the paragraph text only when the browser is *not* Internet Explorer:

```
<!--[if !IE]>
  <p>You are not running Internet Explorer.</p>
<![endif]-->
```

Note that if you omit the version number, the conditional comment is applied to all Internet Explorer versions.

The need for conditional comments arose because Internet Explorer significantly differed from other browsers in how it implemented HTML and there was a need to separate the code meant for the IE browser from code meant for other browsers. This is not as much of a problem with recent versions of Internet Explorer, but you may still need to use conditional comments if you are writing code that will be compatible with versions of Internet Explorer earlier than IE 8.

In the next session, you'll continue your work on the `ct_about.html` file by adding content to the page body.

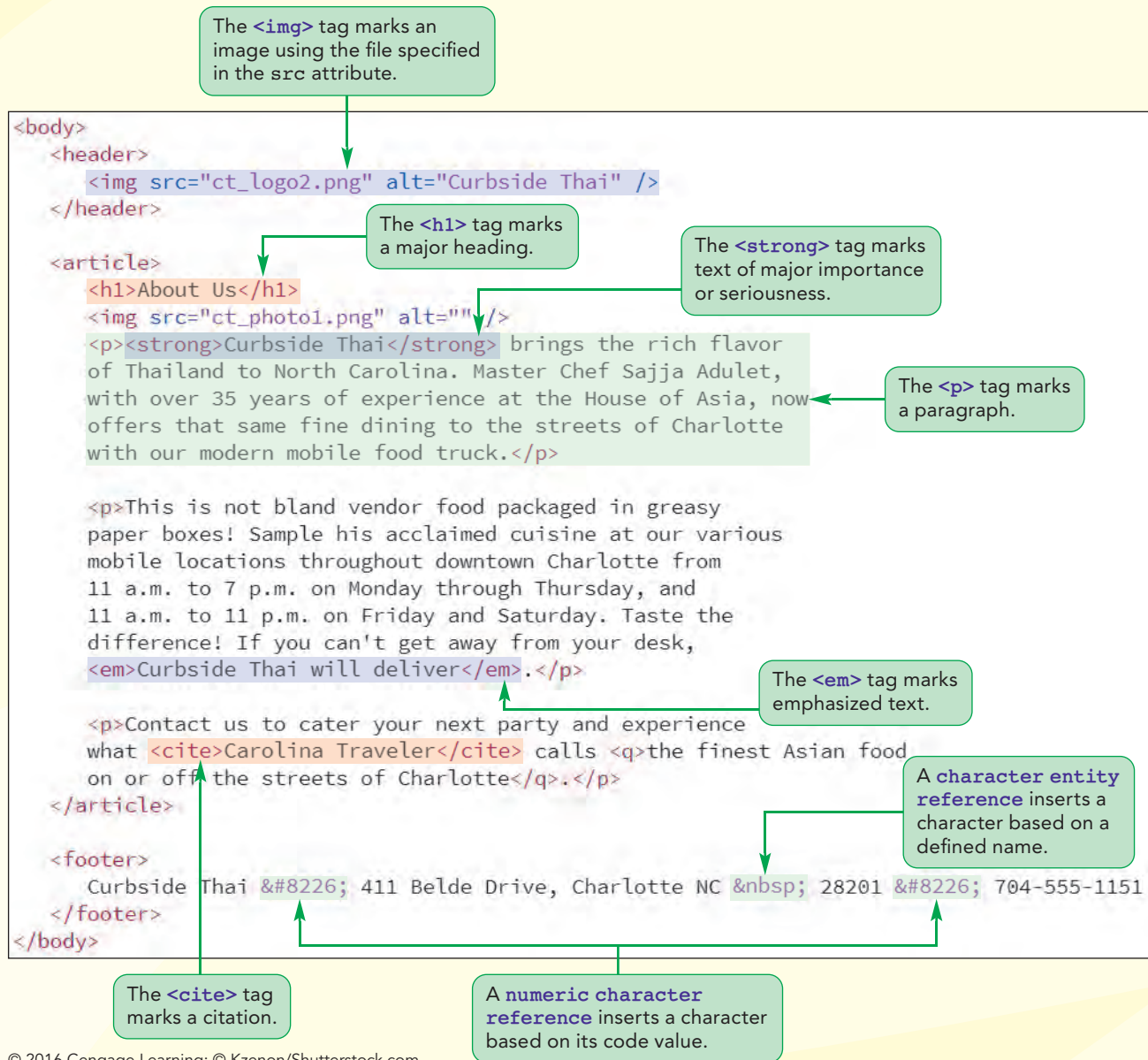


### Session 1.1 Quick Check

1. What is a markup language?
2. What is XHTML? How does XHTML differ from HTML?
3. What is the W3C? What is the WHATWG?
4. What is a doctype? What is the doctype for an HTML5 document?
5. What is incorrect about the following code? Suggest a possible revision of the code to correct the error.  

```
<p><strong>Curbside Thai now delivers!</p></strong>
```
6. Provide code to mark *Curbside Thai Employment Opportunities* as the document title.
7. Provide code to create metadata adding the keywords *food truck*, *North Carolina*, and *dining* to the document.
8. Provide code to tell the browser that the character encoding UTF-16 is used in the document.
9. Provide code to add the comment *Created by Sajja Adulet* to the document.

# Session 1.2 Visual Overview:



# HTML Page Elements

The opening paragraph of the article is marked with the `<p>` tag.

Images are added to the web page.

## Curbside Thai

The main heading of the article is marked with the `<h1>` tag.

### About Us


The restaurant name marked with the `<strong>` tag to indicate its importance.

Curbside Thai brings the rich flavor of Thailand to North Carolina. Owner and chef Sajja Adulet, with over 35 years of experience as the award-winning master chef at the House of Asia, now offers that same fine dining to the streets of Charlotte through our modern mobile food truck.

This is not bland vendor food packaged in greasy paper boxes! Sample our acclaimed cuisine at our various mobile locations throughout downtown Charlotte from 11 a.m. to 7 p.m. (M-R) and 11 a.m. to 11 p.m. on Friday and Saturday. Taste the difference! If you can't get away from your desk, *Curbside Thai will deliver.*

Contact us to cater your next party and experience what *Carolina Traveler* calls "the finest Asian food on or off the streets of Charlotte."

A citation to a magazine is marked with the `<cite>` tag.



Nonbreaking space is inserted with the `&nbsp;` character entity reference.

---

Curbside Thai • 411 Belde Drive, Charlotte NC 28201 • 704-555-1151

An example of emphasized text is marked with the `<em>` tag.

Bullet characters are inserted with the `&#8226;` numeric character reference.

## Writing the Page Body

Now that you have created the document head of the About Curbside Thai web page, you'll begin writing the document body. You will start with general markup tags that identify the major sections of the page body and then work inward to more specific content within those sections.

### Using Sectioning Elements

The first task in designing the page body is to identify the page's major topics. A page typically has a header, one or more articles that are the chief focus of the page, and a footer that provides contact information for the author or company. HTML marks these major topical areas using the **sectioning elements** described in Figure 1-11.

**Figure 1-11** HTML sectioning elements

Element	Description
address	Marks contact information for an individual or group
article	Marks a self-contained composition in the document such as a newspaper story <b>[HTML5]</b>
aside	Marks content that is related to a main article <b>[HTML5]</b>
body	Contains the entire content of the document
footer	Contains closing content that concludes an article or section <b>[HTML5]</b>
h1, h2, h3, h4, h5, h6	Marks major headings with h1 representing the heading with the highest rank, h2 representing next highest-ranked heading, and so forth
header	Contains opening content that introduces an article or section <b>[HTML5]</b>
nav	Marks a list of hypertext or navigation links <b>[HTML5]</b>
section	Marks content that shares a common theme or purpose on the page <b>[HTML5]</b>

© 2016 Cengage Learning

For example, a news blog page might contain several major topics. To identify these areas, the HTML code for the blog might include the following elements to mark off the page's header, navigation list, article, aside, and footer.

```
<body>
  <header>
</header>
  <nav>
</nav>
  <article>
</article>
  <aside>
</aside>
  <footer>
</footer>
</body>
```

#### TIP

Sectioning elements can be nested within each other; for example, an article might contain its own header, footer, and collection of navigation links.

These sectioning elements are also referred to as **semantic elements** because the tag name describes the purpose of the element and the type of content it contains. Even without knowing much about HTML, the page structure defined in the above code is easily understood because of the tag names.

## REFERENCE

*Defining Page Sections*

- To mark the page header, use the `header` element.
- To mark self-contained content, use the `article` element.
- To mark a navigation list of hypertext links, use the `nav` element.
- To mark a sidebar, use the `aside` element.
- To mark the page footer, use the `footer` element.
- To group general content, use the `section` element.

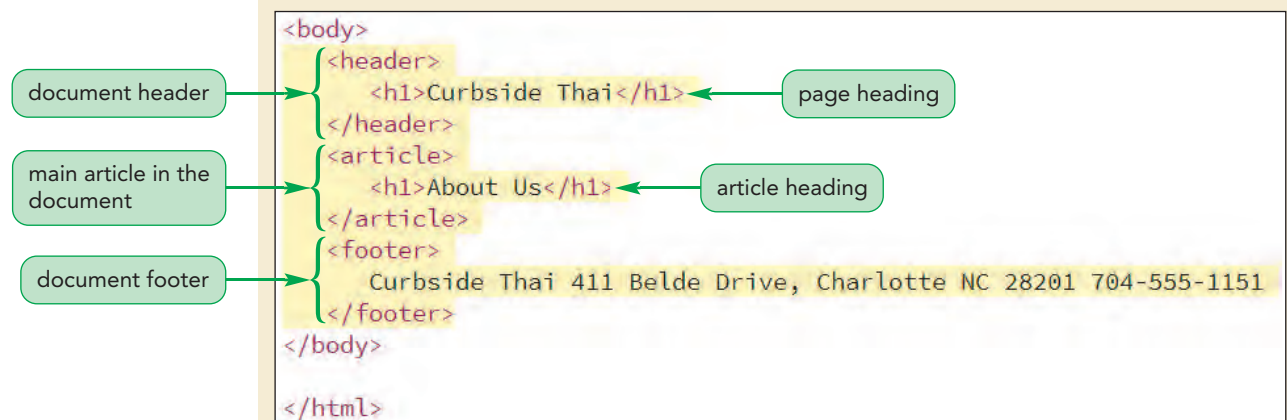
The About Curbside Thai page will have a simple structure containing a header, a single article, and a footer. Within the header, there will be an `h1` element providing the page title (not to be confused with the document title, which is displayed on the browser title bar or a browser tab). Add this structure to the document body.

**To define the sections in the page body:**

1. If you took a break after the previous session, return to the `ct_about.html` file in your HTML editor.
2. Directly after the opening `<body>` tag, insert the following HTML code, indented to make the code easier to read:

```
<header>
  <h1>Curbside Thai</h1>
</header>
<article>
  <h1>About Us</h1>
</article>
<footer>
  Curbside Thai 411 Belde Drive, Charlotte NC 28201 704-555-
  1151
</footer>
```

Figure 1-12 highlights the sectioning elements used in the page body.

**Figure 1-12****Adding sectioning elements to the page body**

3. Save your changes to the file.



## Comparing Sections in HTML4 and HTML5

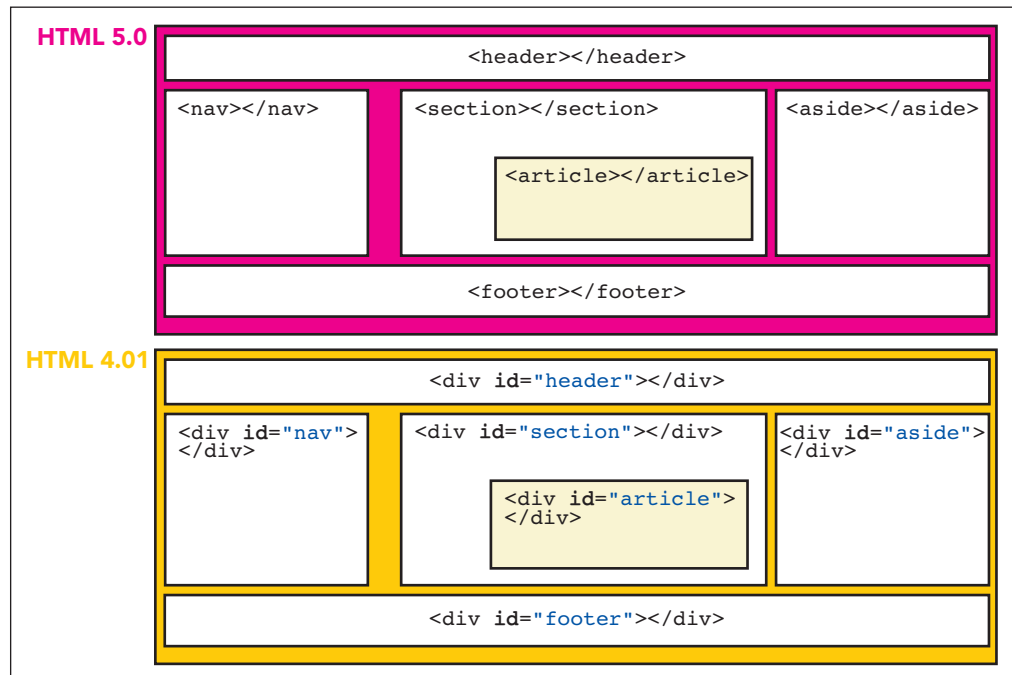
Many of the sectioning elements described in Figure 1-11 were introduced in HTML5. Prior to HTML5, sections were defined as divisions created using the following `div` element:

```
<div id="id">
    content
</div>
```

where *id* is a name that uniquely identifies the division. Figure 1-13 shows how the same page layout marked up using sectioning elements in HTML5 would have been defined in HTML 4.01 using `div` elements.

Figure 1-13

### Sections in HTML 5.0 vs. divisions in HTML 4.01



© 2016 Cengage Learning

One problem with `div` elements is that there are no rules for the ids. One web designer might identify the page heading with the id *header* while another designer might use *heading* or *top*. The lack of consistency makes it harder for search engines to identify the page's main topics. The advantage of the HTML5 sectioning elements is that their tag name indicates their purpose in the document, leading to greater uniformity in how pages are designed and interpreted.

## Using Grouping Elements

Within sectioning elements are **grouping elements**. Each grouping element organizes similar content into a distinct group, much like a paragraph groups sentences that share a common theme. Figure 1-14 describes all the HTML grouping elements.

Figure 1-14

## HTML grouping elements

Element	Description
<code>blockquote</code>	Contains content that is quoted from another source, often with a citation and often indented on the page
<code>div</code>	Contains a generic grouping of elements within the document
<code>dl</code>	Marks a description list containing one or more <code>dt</code> elements with each followed by one or more <code>dd</code> elements
<code>dt</code>	Contains a single term from a description list
<code>dd</code>	Contains the description or definition associated with a term from a description list
<code>figure</code>	Contains an illustration, photo, diagram, or similar object that is cross-referenced elsewhere in the document [ <b>HTML5</b> ]
<code>figcaption</code>	Contains the caption associated with a figure [ <b>HTML5</b> ]
<code>hr</code>	Marks a thematic break such as a scene change or a transition to a new topic (often displayed as a horizontal rule)
<code>main</code>	Marks the main content of the document or application; only one <code>main</code> element should be used in the document [ <b>HTML5</b> ]
<code>ol</code>	Contains an ordered list of items
<code>ul</code>	Contains an unordered list of items
<code>li</code>	Contains a single item from an ordered or unordered list
<code>p</code>	Contains the text of a paragraph
<code>pre</code>	Contains a block of preformatted text in which line breaks and extra spaces in the code are retained (often displayed in a monospace font)

© 2016 Cengage Learning

For example, the following code shows three paragraphs nested within a page article with each paragraph representing a group of similar content:

```
<article>
  <p>Content of 1st paragraph.</p>
  <p>Content of 2nd paragraph.</p>
  <p>Content of 3rd paragraph.</p>
</article>
```

When a browser encounters a sectioning element or a grouping element, the default style is to start the enclosed content on a new line, separating it from any content that appears before it. Thus, each of these paragraphs will be started on a new line as will the article itself. Note that the exact appearance of the paragraphs and the space between them depends on the styles applied by the browser to those elements. You'll learn more about styles later in this tutorial.

## REFERENCE

## Defining Page Groups

- To mark a paragraph, use the `p` element.
- To mark an extended quote, use the `blockquote` element.
- To mark the main content of a page or section, use the `main` element.
- To mark a figure box, use the `figure` element.
- To mark a generic division of page content, use the `div` element.

Sajja has written up the article describing Curbside Thai in a text file. Enter his text into the `article` element in the About Curbside Thai web page and use `p` elements to mark the paragraphs in the article.

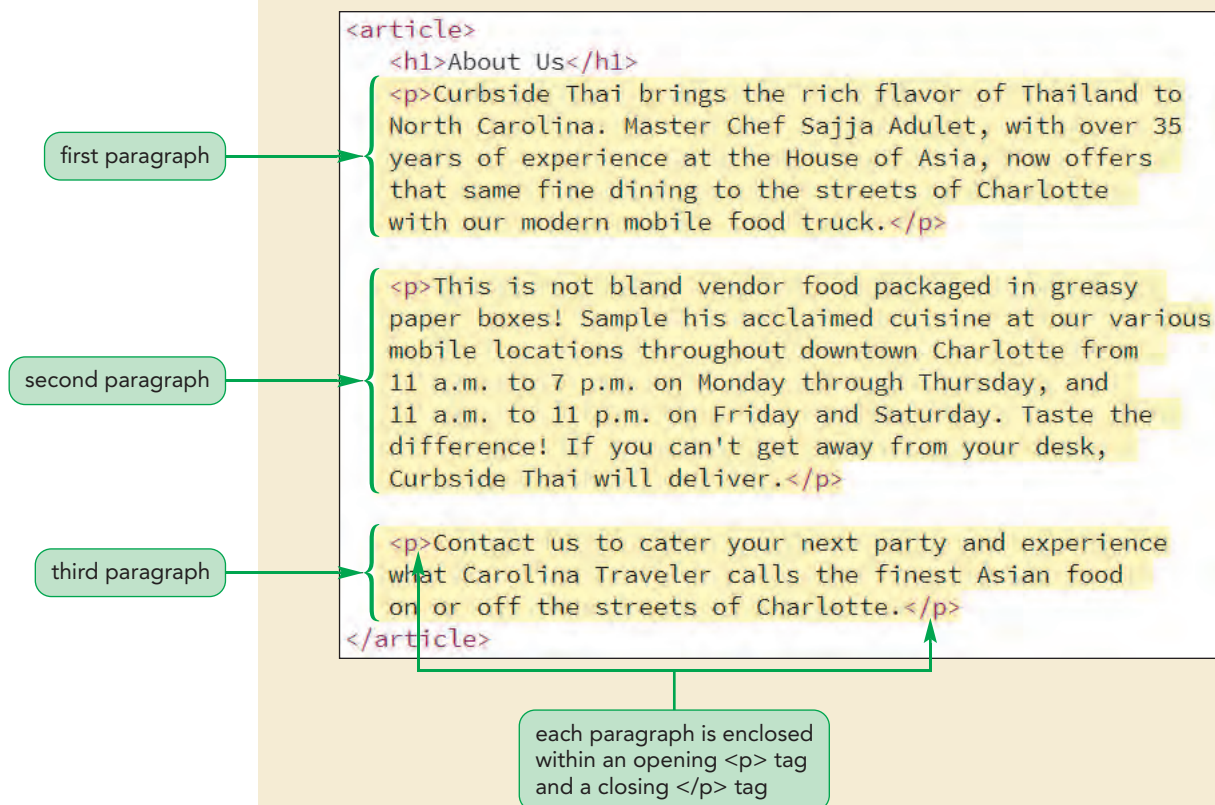
### To group the page text into paragraphs:

1. Use a text editor to open the **ct\_pages.txt** file from the **html01 ► tutorial** folder.
2. Select and copy the three paragraphs of text directly after the About Us title.
3. Close the file, but do not save any changes you may have inadvertently made to the document.
4. Return to the **ct\_about.html** file in your HTML editor.
5. Directly after the `<h1>About Us</h1>` line within the page article, insert a new blank line and paste the text you copied.
6. Enclose each of the three paragraphs of pasted content between an opening `<p>` tag and a closing `</p>` tag. Indent the code within the `article` element to make the code easier to read.

Figure 1-15 highlights the newly added code for the three paragraphs of article text

Figure 1-15

### Grouping article content by paragraphs



7. Save your changes to the file.

## Using Text-Level Elements

Within each grouping element are **text-level elements**, which act like phrases or characters within a paragraph. Unlike sectioning or grouping elements that start content on a new line and mark a self-contained block of content, text-level elements appear in line with the surrounding content and are known as **inline elements**. For example, the *italicized* or **boldface** text in this paragraph is considered inline content because it appears alongside the surrounding text. Figure 1-16 describes some of the many text-level elements in HTML.

Figure 1-16 HTML text-level elements

Element	Description
<code>a</code>	Marks content that acts as a hypertext link
<code>abbr</code>	Marks an abbreviation or acronym
<code>b</code>	Indicates a span of text to which attention should be drawn (text usually appears in bold)
<code>br</code>	Represents a line break within the grouping element
<code>cite</code>	Marks a citation to a title or author of a creative work (text usually appears in italics)
<code>code</code>	Marks content that represents computer code (text usually appears in a monospace font)
<code>data</code>	Associates a data value with the marked text with the <code>value</code> attribute providing the value <b>[HTML5]</b>
<code>dfn</code>	Marks a defined term for which a definition is given elsewhere in the document
<code>em</code>	Indicates content that is emphasized or stressed (text usually appears in italics)
<code>i</code>	Indicates a span of text that expresses an alternate voice or mood (text usually appears in italics)
<code>kbd</code>	Marks text that represents user input, typically from a computer keyboard or a voice command
<code>marks</code>	Contains a row of text that is marked or highlighted for reference purposes <b>[HTML5]</b>
<code>q</code>	Marks content that is quoted from another source
<code>s</code>	Marks content that is no longer accurate or relevant (text is usually struck through)
<code>samp</code>	Marks text that represents the sample output from a computer program or application
<code>small</code>	Marks side comments (text usually in small print)
<code>span</code>	Contains a generic run of text within the document
<code>strong</code>	Indicates content of strong importance or seriousness (text usually appears in bold)
<code>sub</code>	Marks text that should be treated as a text subscript
<code>sup</code>	Marks text that should be treated as a text superscript
<code>time</code>	Marks a time value or text string <b>[HTML5]</b>
<code>u</code>	Indicates text that appears stylistically different from normal text (text usually appears underlined)
<code>var</code>	Marks text that is treated as a variable in a mathematical expression or computer program
<code>wbr</code>	Represents where a line break should occur, if needed, for a long text string <b>[HTML5]</b>

© 2016 Cengage Learning

The following HTML code demonstrates how to employ text-level elements to mark select phrases or characters within a paragraph.

```
<p>
  Contact us to cater your next party and experience what
  <cite>Carolina Traveler</cite> calls <q>the finest
  Asian food on or off the streets of Charlotte.</q>
</p>
```

Two text-level elements are used in this paragraph: the `cite` element to mark the citation to the *Carolina Traveler* magazine and the `q` element to mark the direct quote from the magazine's review of Curbside Thai. Both the citation and the quoted material will appear specially formatted within the paragraph alongside the other, unmarked, text.

## REFERENCE

### Defining Text-Level Content

- To mark emphasized text, use the `em` element.
- To mark text of great importance, use the `strong` element.
- To mark a citation, use the `cite` element.
- To mark a selection of quoted material, use the `q` element.
- To mark a subscript, use the `sub` element; to mark a superscript, use the `sup` element.
- To mark a generic selection of text-level content, use the `span` element.

Use text-level elements in the About Curbside Thai web page to mark examples of emphasized text, strongly important text, citations, and quoted material.

### To apply text-level elements to a page:

1. Go to the first paragraph within the page article and enclose the opening words *Curbside Thai* within a set of opening and closing `<strong>` tags. You use the `<strong>` tags when you want to strongly reinforce the importance of the text, such as the restaurant name, for the reader.
2. In the second paragraph, enclose the phrase, *Curbside Thai will deliver* within a set of opening and closing `<em>` tags to emphasize this text.
3. Go the third paragraph and mark *Carolina Traveler* using the `cite` element and then mark the extended quote, *the finest Asian food on or off the streets of Charlotte*, using the `q` element.

Figure 1-17 highlights the application of the four text-level elements to the paragraph text.

Figure 1-17

### Marking text-level content

```

<article>
  <h1>About Us</h1>
  <p><strong>Curbside Thai</strong> brings the rich flavor of Thailand to
  North Carolina. Master Chef Sajja Adulet, with over 35
  years of experience at the House of Asia, now offers
  that same fine dining to the streets of Charlotte
  with our modern mobile food truck.</p>

  <p>This is not bland vendor food packaged in greasy
  paper boxes! Sample his acclaimed cuisine at our various
  mobile locations throughout downtown Charlotte from
  11 a.m. to 7 p.m. on Monday through Thursday, and
  11 a.m. to 11 p.m. on Friday and Saturday. Taste the
  difference! If you can't get away from your desk,
  <em>Curbside Thai will deliver</em>.</p>

  <p>Contact us to cater your next party and experience
  what <cite>Carolina Traveler</cite> calls <q>the finest Asian food
  on or off the streets of Charlotte</q>.</p>
</article>

```

strong and important text marked with the `<strong>` tag

emphasized text marked with the `<em>` tag

citation marked with the `<cite>` tag

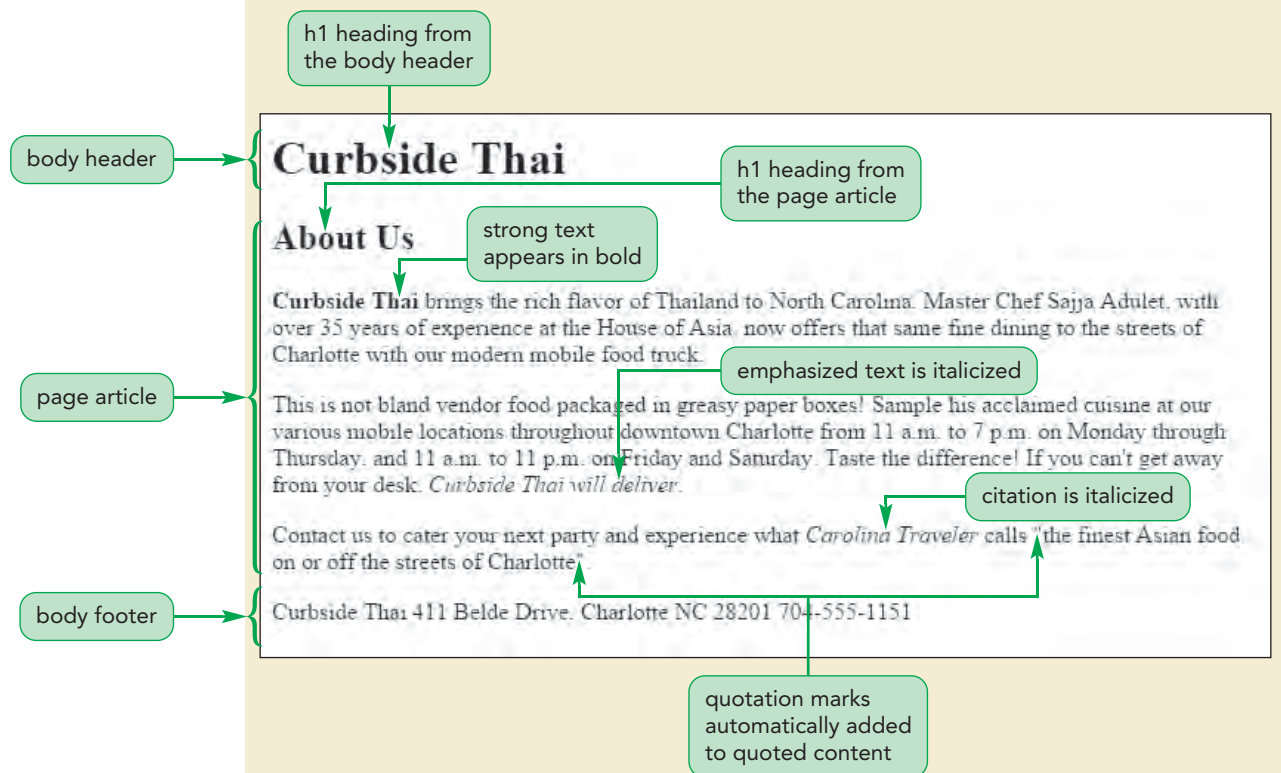
quoted material marked with the `<q>` tag

4. Save your changes to the file.
5. Open the **ct\_about.html** file in your browser to view how your browser renders the page content.

Figure 1-18 shows the current appearance of the page.

Figure 1-18

The About Curbside Thai page as rendered by the browser



**Trouble?** Depending on your browser and/or device, you might see some minor differences in the appearance of the About Curbside Thai web page from that shown in Figure 1-18.

In rendering the page, the browser made the following stylistic choices for the different page elements:

- The h1 heading from the body header is assigned the largest font and is displayed in bold to emphasize its importance. The h1 heading from the page article is given a slightly smaller font but is still displayed in bold.
- Strong text is displayed in bold while emphasized text is displayed in italics.
- Citations are displayed in italic while quoted material is automatically surrounded by quotation marks.

It needs to be emphasized again that all of these stylistic choices are not determined by the markup tags; they are default styles used by the browser. Different browsers and different devices might render these page elements differently. To exert more control over your page's appearance, you can apply a style sheet to document contents.



## Linking an HTML Document to a Style Sheet

A **style sheet** is a set of rules specifying how page elements are displayed. Style sheets are written in the **Cascading Style Sheets (CSS)** language. Like HTML, the CSS language was developed and enhanced as the web grew and changed and, like HTML, CSS specifications are managed by the W3C. To replace the browser's internal style sheet with one of your own, you can link your HTML file to a style sheet file using the following `link` element:

```
<link href="file" rel="stylesheet" />
```

where *file* is a text file containing the CSS style sheet. Because the `link` element can also be used to link to data other than style sheets, the `rel` attribute is required to tell the browser that it is linking to style sheet data. Note that older browsers might include `type="text/css"` as part of the `link href` element.

### Linking an HTML Document to an External Style Sheet

- To link an HTML document to an external style sheet file, add the following element to the document head:

```
<link href="file" rel="stylesheet" />
```

where *file* is a text file containing the CSS style rules.

#### TIP

Because the `link` element is another example of metadata, it's always added to the document head.

Sajja has supplied you with two CSS files that he wants applied to his website. The `ct_base.css` file contains styles specifying the appearance of text-level elements. The `ct_layout2.css` file contains styles that govern the arrangement of sectioning and grouping elements on the page. Link the `ct_about.html` file to both of these style sheets now.

### To link an HTML document to a style sheet:

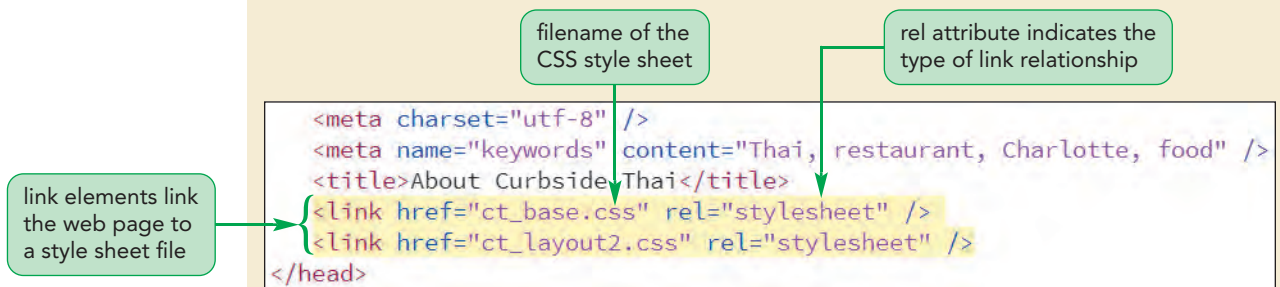
1. Return to the `ct_about.html` file in your HTML editor.
2. Directly before the closing `</head>` tag, insert the following `link` elements:

```
<link href="ct_base.css" rel="stylesheet" />
<link href="ct_layout2.css" rel="stylesheet" />
```

Figure 1-19 highlights the two style sheet links added to the document.

Figure 1-19

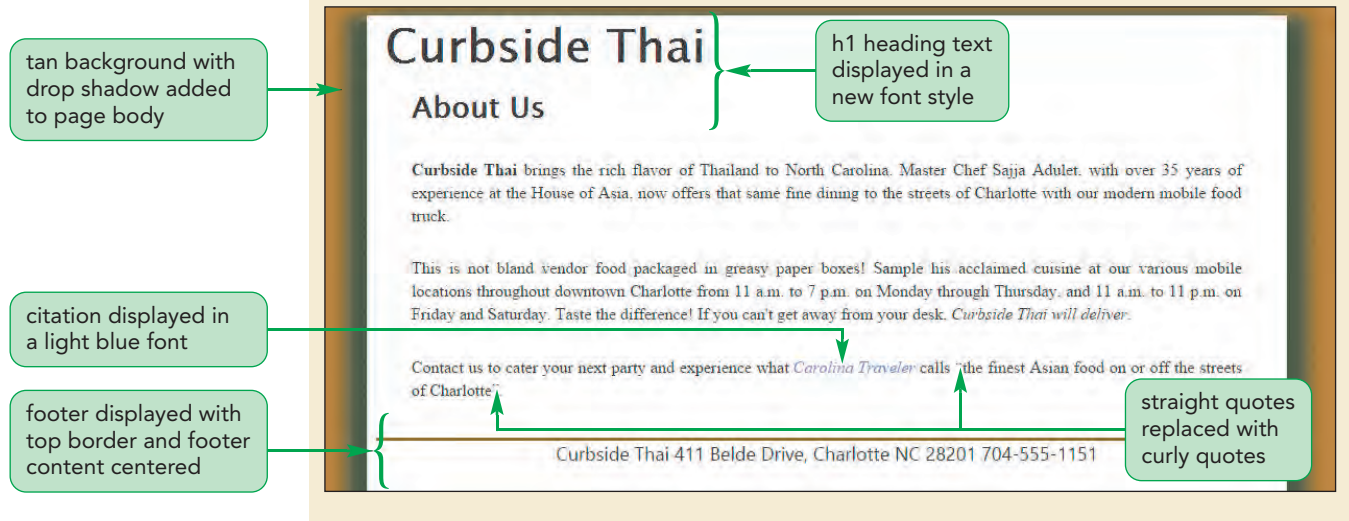
### Linking to style sheets



3. Save your changes to the file and then reload the `ct_about.html` file in your browser. Figure 1-20 shows the new appearance of the page using the style sheets provided by Sajja.

Figure 1-20

The About Curbside Thai page rendered under a new style sheet



Applying these style sheets to the HTML code causes the page body to be displayed on a tan background with a drop shadow, the font used in the two h1 headings has changed, a top border has been added to the footer to set it off from the preceding content, and the citation to the *Carolina Traveler* magazine is displayed in a light blue font. The effect makes the page content easier to read and more pleasing to the eye.

Sajja is concerned that the contact information in the page footer is difficult to read. He wants you to add bullet characters ( • ) separating the name of the restaurant, the street address, and the restaurant phone number. However, this character is not represented by any keys on your keyboard. How then, do you insert this symbol into the web page?

## Working with Character Sets and Special Characters

Every character that your browser is capable of rendering belongs to a collection of characters and symbols called a **character set**. The character set used for the English alphabet is the **American Standard Code for Information Interchange** more simply known as **ASCII**. A more extended character set, called **Latin-1** or the **ISO 8859-1** character set, supports 255 characters and can be used by most languages that employ the Latin alphabet, including English, French, Spanish, and Italian. **Unicode**, the most extended character set, supports up to 65,536 symbols and can be used with any of the world's languages.

### Character Encoding

#### TIP

You can explore different character encoding values by opening the `demo_characters.html` file in the `html01 ► demo` folder.

Each character from a character set is associated with an encoding value that can then be stored and read by a computer program. For example, the copyright symbol © from the Unicode character set is encoded with the number 169. If you know the encoding value, you can insert the corresponding character directly into your web page using the following character encoding reference:

```
&#code;
```

where *code* is the encoding reference number. Thus, to display the © symbol in your web page, you would enter

```
&#169;
```

into your HTML file.

## Character Entity References

Another way to insert a special symbol is to use a character entity reference, which is a short memorable name used in place of the encoding reference number. Character entity references are inserted using the syntax

```
&char;
```

where *char* is the character's entity reference. The character entity reference for the copyright symbol is *copy*, so to display the © symbol in your web page, you could insert the following expression into your HTML code:

```
&copy;
```

In the last session, you learned that HTML will collapse consecutive occurrences of white space into a single white-space character. You can force HTML to display extra white space by using the following character entity reference

```
&nbsp;
```

where *nbsp* stands for *nonbreaking space*. When you want to display extra white space, you need to insert the nonbreaking space character reference in the HTML code for each space you want to display.

### REFERENCE

#### Inserting Symbols from a Character Set

- To insert a symbol based on the character encoding reference number, enter

```
&#code;
```

where *code* is the character encoding reference number.

- To insert a symbol based on a character entity reference, enter

```
&char;
```

where *char* is the name assigned to the character.

- To insert a white-space character, use

```
&nbsp;
```

For the footer in the About Curbside Thai page, use the bullet symbol ( • ), which has the encoding value 8226, to separate the restaurant name, address, and phone number. Use the `&nbsp;` character reference to insert an extra blank space prior to the postal code in the restaurant address.

### To insert a character encoding reference number and an entity reference:

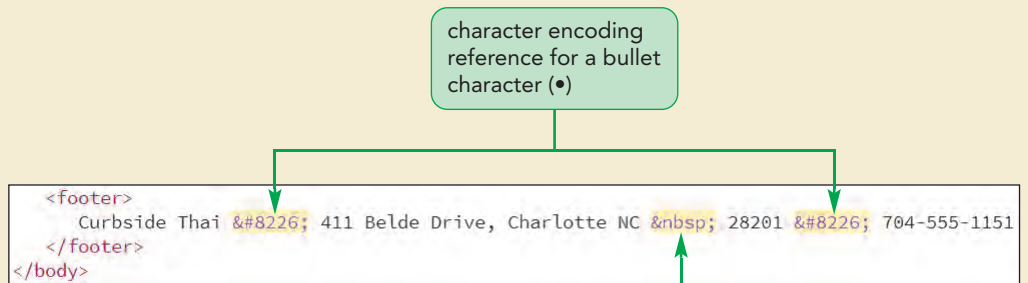
1. Return to the **ct\_about.html** file in your HTML editor.
2. Go to the `footer` element and insert the character encoding number `&#8226;` directly after the word *Thai* and after the postal code `28201`. Insert the character reference `&nbsp;` directly before the postal code.

Figure 1-21 highlights the character codes and references added to the footer.

Character encoding reference numbers must always begin with `&#` and end with a semicolon, otherwise the code won't be recognized as a code number.

Figure 1-21

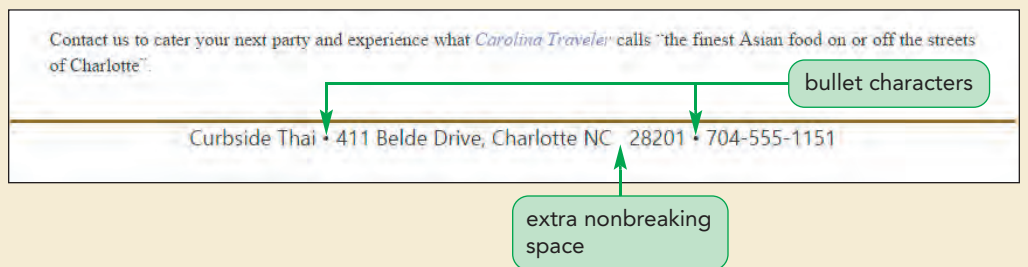
#### Inserting special characters



3. Save your changes to the file and then reload the `ct_about.html` file in your browser. Confirm that the footer now shows the characters displayed in Figure 1-22.

Figure 1-22

#### Revised page footer



## INSIGHT

### Presentational Attributes

Early versions of HTML supported **presentational elements** and **presentational attributes** designed to describe how each element should be rendered by web browsers. For example, to align text on a page, web authors would use the following `align` attribute

```
<element align="alignment">content</element>
```

where *alignment* is either `left`, `right`, `center`, or `justify`. Thus, to center an `h1` heading on a page, they would use the following code:

```
<h1 align="center">Curbside Thai</h1>
```

Almost all presentational elements and attributes are now deprecated in favor of style sheets, but you may still see them in the code from older websites. Using a deprecated attribute like `align` would probably not cause your web page to fail, however, it's still best practice to adhere to a standard in which HTML is used only to describe the content and structure of the document and style sheets are used to format its appearance.

So far your work on the Curbside Thai page has been limited to textual content. Next, you'll explore how to add graphical content to your web page.

## Working with Inline Images

Most web pages include **embedded content**, which is content imported from another resource, often nontextual, such as graphic images, audio soundtracks, video clips, or interactive games. To support this type of content, HTML provides the **embedded elements** listed in Figure 1-23.

Figure 1-23

HTML embedded elements

Element	Description
<code>audio</code>	Represents a sound clip or audio stream <b>[HTML5]</b>
<code>canvas</code>	Contains programming scripts used to construct bitmap images and graphics <b>[HTML5]</b>
<code>embed</code>	Contains general embedded content including application or interactive content
<code>iframe</code>	Contains the contents of an external web page or Internet resource
<code>img</code>	Contains a graphic image retrieved from an image file
<code>object</code>	Contains general embedded content including application or interactive content
<code>video</code>	Represents a video clip or video stream with captions <b>[HTML5]</b>

© 2016 Cengage Learning

**TIP**

Always include the `alt` attribute; it is required in XHTML code and is highly recommended as a way of accommodating users running nonvisual web browsers.

These elements are also known as **interactive elements** because they allow for interaction between the user and the embedded object. For example, embedded audio or video content usually contains player buttons to control the playback.

Images are inserted into a web page using the following `img` element

```

```

where *file* is the name of the image file. If the browser cannot display images, the text in the `alt` attribute is used in place of the image. As with other one-sided tags, the `img` element can be entered without the closing slash as

```

```