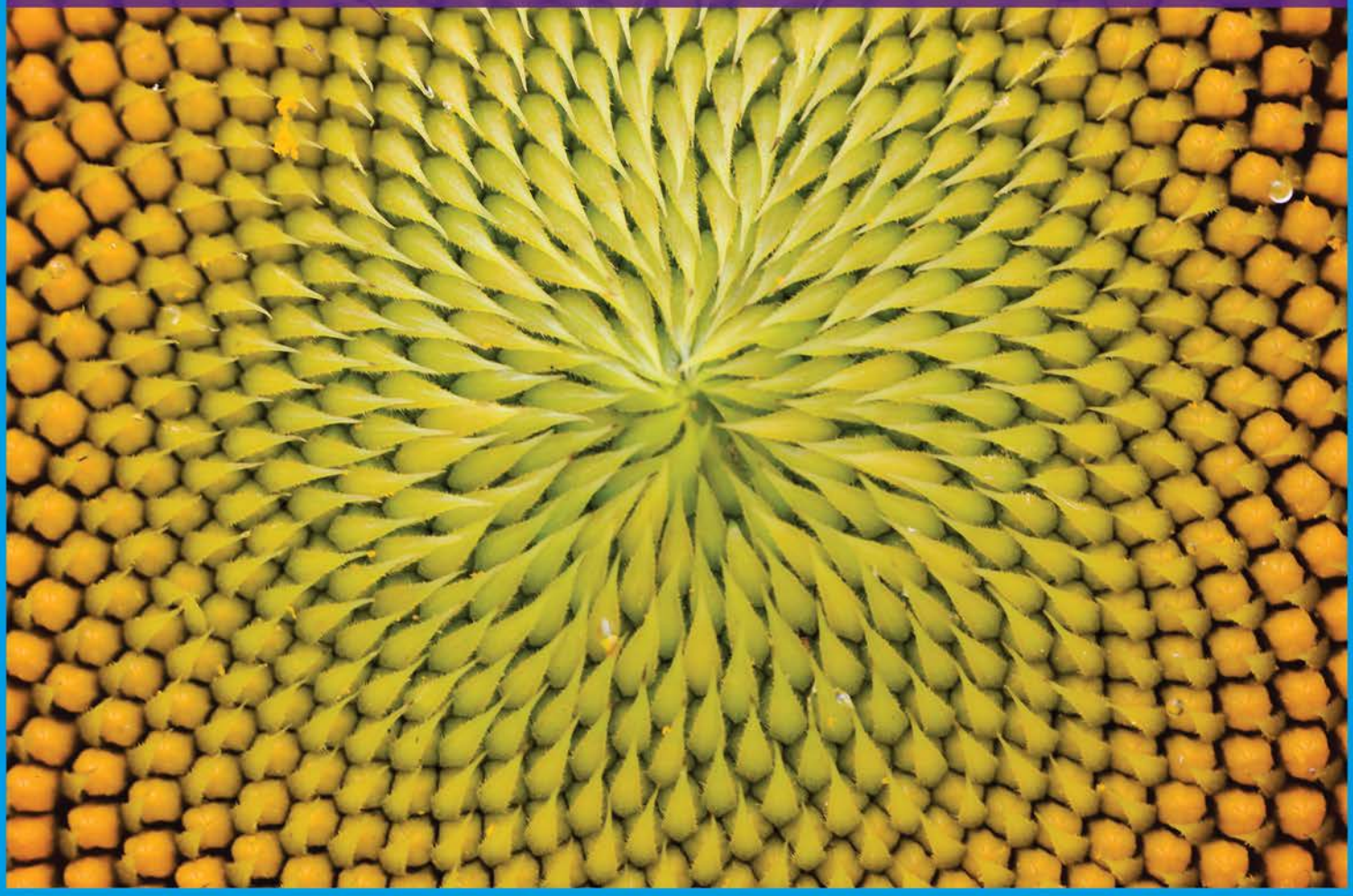
CENGAGE

NEW PERSPECTIVES

CAREY

HTML 5 and CSS

Comprehensive

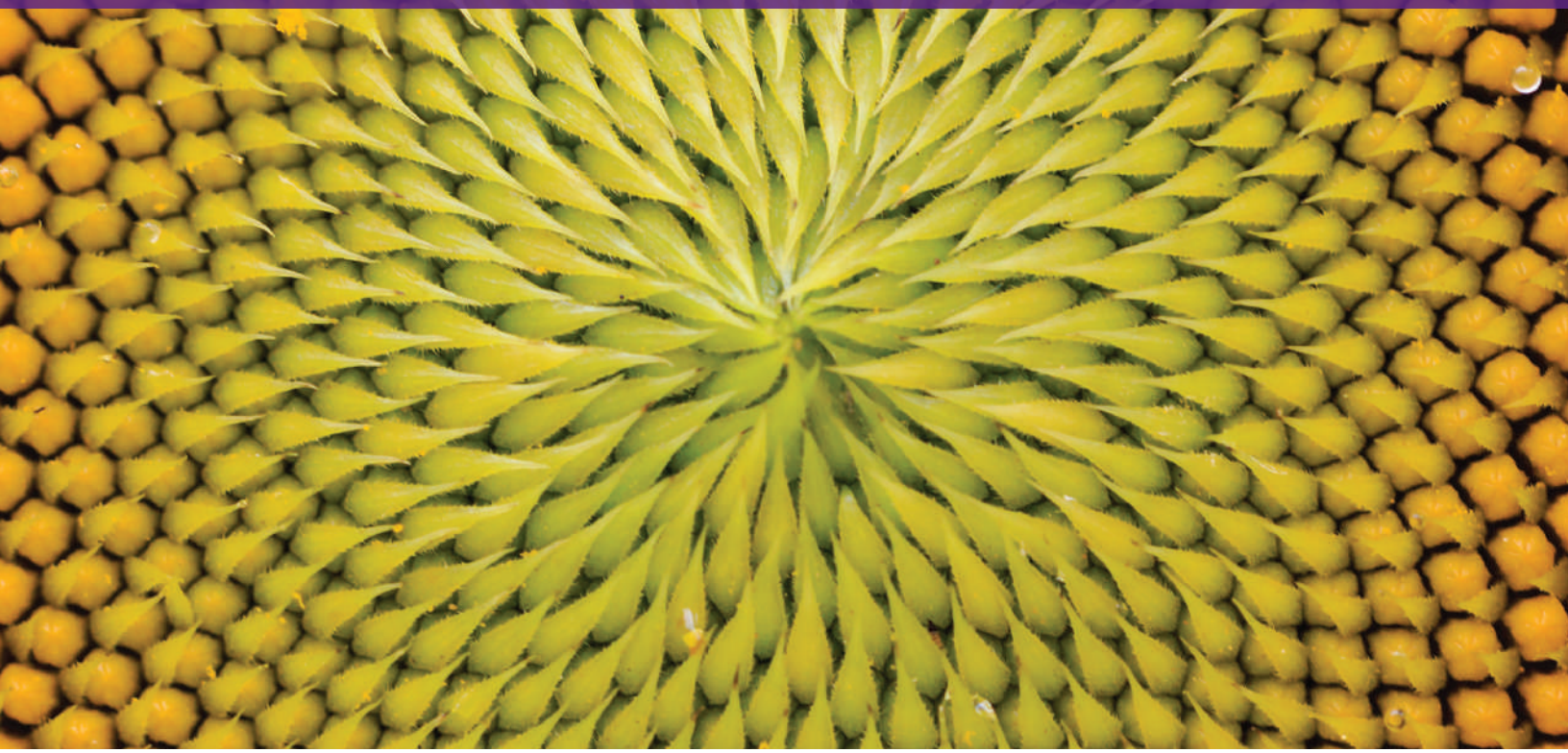


NEW PERSPECTIVES

PATRICK CAREY

HTML 5 and CSS

Comprehensive



CENGAGE

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

New Perspectives on HTML 5 and CSS, 8th Edition Comprehensive
Patrick Carey

SVP, Higher Education Product Management: Erin Joyner

VP, Product Management: Mike Schenk

Product Director: Lauren Murphy

Product Team Manager: Kristin McNary

Product Assistant: Tom Benedetto

Director, Learning Design: Rebecca von Gillern

Senior Manager, Learning Design: Leigh Hefferon

Learning Designer: Kate Mason

Vice President, Marketing – Science, Technology, & Math: Jason Sakos

Senior Marketing Director: Michele McTighe

Marketing Manager: Cassie L Cloutier

Marketing Development Manager: Samantha Best

Director, Content Creation: Juliet Steiner

Senior Manager, Content Creation: Patty Stephan

Content Manager: Christina Nyren

Director, Digital Production Services: Krista Kellman

Digital Delivery Lead: Justin Maniaci

Technical Editors: John Freitas and Danielle Shaw

Developmental Editors: Deb Kaufmann and Ann Shaffer

Production Service/Composition: Lumina Datamatics Ltd.

Design Director: Jack Pendleton

Designer: Erin Griffin

Text Designer: Althea Chen

Cover Template Designer: Wing-Ip Ngan, Ink Design, Inc.

Cover image(s): Mai Phongsook/ShutterStock.com

Library of Congress Control Number: 2019940290

ISBN: 978-0-357-10714-0

Cengage
200 Pier 4 Boulevard
Boston, MA 02210
USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com.**

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

To learn more about Cengage platforms and services, register or access your online learning solution, or purchase materials for your course, visit **www.cengage.com.**

Printed in the United States of America
Print Number: 01 Print Year: 2019

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 HTML 5 and CSS has 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 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."
—Customer at St. Johns River Community College

About This Book

This book provides thorough coverage of HTML 5 and CSS, and includes the following:

- Up-to-date coverage of using HTML 5 to create structured websites
- Instruction on the most current CSS 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 CSS grid styles for creating grid-based layouts.
- Exploration of new CSS styles for filters and transformations.
- New hands-on demo pages to interactively explore HTML and CSS concepts.
- New coding challenges for each tutorial to focus on specific tasks and concepts.
- New debugging challenges for each tutorial to explore how to fix malfunctioning websites.

System Requirements

This book assumes that students have an Internet connection, a text editor, and a current browser that supports HTML 5 and CSS. 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 44, Firefox 66, Safari 12.1, Opera 60, and Google Chrome 75. 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 HTML 5 and CSS 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 75 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."
—Customer at Southeastern Technical College

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 OVERVIEW

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.

TRY IT

Try It tips point to demo pages provided with the data folder for interactive exploration of key concepts.

REVIEW
CODE
DEBUG
APPLY
CHALLENGE
CREATE

Assessment
Retention is a key component to learning. At the end of each session, a series of Quick Check multiple choice questions helps students test their understanding of the material before moving on. New with this edition are Coding Challenges and debugging exercises that focus on a few key challenges. 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.

REFERENCE

Reference
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.

GLOSSARY/INDEX

Our Complete System of Instruction

INTRODUCTORY

COMPREHENSIVE

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 web site or contact your Cengage 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 HTML 5 and CSS 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, Cognero testbank, PowerPoint presentations, data files, solution files, figure files, and a sample syllabus are all available at 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 editors, Deb Kaufmann and Ann Shaffer, for their hard work, attention to detail, and valuable insights, and to Content Manager, Christina Nyren, 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 Kristin McNary, Program Team Lead;

Kate Mason, Learning Designer; Tom Benedetto, Product Assistant; Erin Griffin, Art Director; Fola Orekoya, Manufacturing Planner; Lumina Datamatics Ltd., Compositor, as well as John Freitas and Danielle Shaw, Technical Editors.

This book is dedicated to my wife Joan who is my inspiration and role model for her good humor, dedication, and tireless support.
– Patrick Carey

BRIEF CONTENTS

HTML

Level I Tutorials

Level II Tutorials

Level III Tutorials

TABLE OF CONTENTS

HTML LEVEL II TUTORIALS

HTML LEVEL III TUTORIALS

TUTORIAL 1

Getting Started with HTML 5

Creating a Website for a Food Vendor

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 email addresses and telephone numbers

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 HTML 5 code and how to get your HTML files ready for display on the World Wide Web.

STARTING DATA FILES

tutorial
ct_catering_txt.html
ct_contact_txt.html
ct_locations_txt.html
ct_menu_txt.html
ct_reviews._txt.html + 16 files

review
mp_catering_txt.html
mp_events_txt.html
mp_index_txt.html
mp_menu_txt.html + 5 files

code1
code1-1_txt.html

code2
code1-2_txt.html

code3
code1-3_txt.html + 7 files

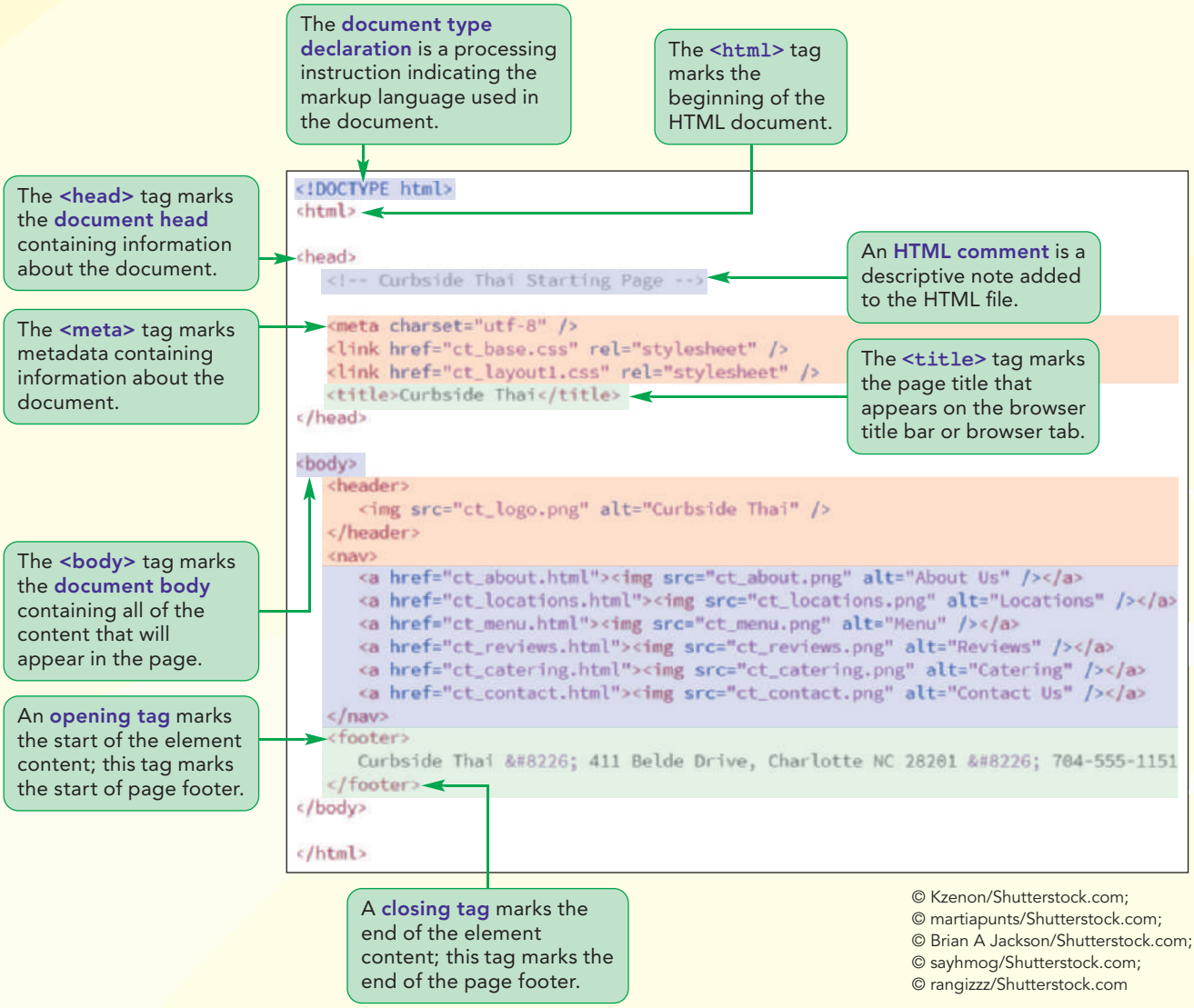
code4
code1-4_txt.html + 2 files

case1
jtc_index_txt.html
jtc_services_txt.html
+ 6 files

case2
dr_faq_txt.html
dr_index_txt.html
dr_info_txt.html + 9 files

demo
demo_characters.html
demo_html.html
+ 3 files

Session 1.1 Visual Overview:



The Structure of an HTML Document

Kzenon/Shutterstock.com; © Courtesy Patrick Carey; martiapunts/Shutterstock.com; Brian A Jackson/Shutterstock.com; A Studios/Shutterstock.com; rangizzz/Shutterstock.com

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 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 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 provides printing services; a file server provides storage space for saving and retrieving files. The device receiving 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.

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, and so 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** activated 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 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 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 in a form 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 tagging 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 that, 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*.

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 **HTML 5**. Unlike XHTML 2.0, HTML 5 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 and the W3C issued a new charter for WHATWG to develop HTML 5 as the de facto standard for the next generation of HTML. You can learn more about WHATWG and its current projects at *www.whatwg.org*. The current version of HTML is HTML 5.2, which achieved Recommendation status in 2017.

TIP

You can find out which browsers support the features of HTML 5 by going to the website *caniuse.com*.

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 HTML 5, 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 HTML 5. Thus, a major challenge for website designers is writing code that takes advantage of HTML 5 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	HTML version providing support for mobile design, semantic page elements, column layout, form validation, offline storage, and enhanced multimedia
HTML 5.2	2017	The current version of HTML 5

This book focuses on HTML 5, but you will also review some of the specifications for HTML 4.01 and XHTML 1.1. Deprecated features from older versions of HTML will be noted as such in the text.

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. These editors include syntax checking to weed out errors and automatic insertion of HTML code. Some of the more popular HTML editors are Notepad++ (*notepad-plus-plus.org*), Eclipse (*www.eclipse.org*), and CoffeeCup (*www.coffeecup.com*).

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*), Aptana Studio (*www.aptana.com*), NetBeans IDE (*netbeans.org*), and Komodo IDE (*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.

Content Management Systems and Frameworks

You can also invest in a **web content management system** (**wcms**) which provides authoring tools for website content and administration. Management systems provide prepackaged templates so that users can get websites up and running with only a minimal knowledge of HTML. Popular content management systems include WordPress (*www.wordpress.org*), Joomla (*www.joomla.org*), and Drupal (*www.drupal.org*). Content management systems are not without drawbacks. A wcms can be expensive to maintain and put extra load on server resources. In addition, the templates and authoring tools can be difficult to modify if they don't exactly meet your needs.

A website usually involves the integration of many technologies and languages beyond HTML, including databases for storing and retrieving data and programs running on the web server for managing electronic commerce and communication. Managing all those technologies is the job of a **web framework** that provides the foundation of the design and deployment of web applications. Popular frameworks include Ruby on Rails (*rubyonrails.org*), ASP.NET (*www.asp.net*), AngularJS (*angularjs.org*), and Django (*www.djangoproject.com*).

Choosing among all these tools might seem intimidating to you. The bottom line is that no matter what tools you use, the final code for the website is written in HTML. So, even if that code is generated by a framework or content management system, you still need to understand HTML to effectively manage your website. In this book, we'll try to keep things as simple as possible: just you, a text editor, and a web browser creating a foundation for future study.

Testing your Code

TIP

You can analyze each browser for its compatibility with HTML 5 at the website *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*). **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*), CrossBrowserTesting (*www.crossbrowsertesting.com*), and Browsera (*www.browsera.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.

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.

To open the ct_start.html file:

TIP

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

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**

two-sided tag enclosing element content

empty elements, which do not contain content

an element attribute

several elements nested within another element

```
<!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" />
   <link href="ct_layout1.css" rel="stylesheet" />
</head>

<body>
   <header>
      <img src="ct_logo.png" alt="Curbside Thai" />
   </header>
   <nav>
      <a href="ct_about.html"><img src="ct_about.png" alt="About Us" /></a>
      <a href="ct_locations.html"><img src="ct_locations.png" alt="Locations" /></a>
      <a href="ct_menu.html"><img src="ct_menu.png" alt="Menu" /></a>
      <a href="ct_reviews.html"><img src="ct_reviews.png" alt="Reviews" /></a>
      <a href="ct_catering.html"><img src="ct_catering.png" alt="Catering" /></a>
      <a href="ct_contact.html"><img src="ct_contact.png" alt="Contact Us" /></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 for displaying the content. For HTML 5, 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 always put your HTML 5 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. The following code marks a paragraph element enclosed within the `<p>` and `</p>` tags:

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

The `<p></p>` tags 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>
```

The following `br` element indicates the presence of a line break in the text, and thus does not contain any content:

```
<br />
```

Note that, while this code could also be entered as `<br>`, the ending slash `/>` form is required in XHTML documents as well as other markup languages. While HTML 5 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. We'll follow the `/>` convention in the code in this book.

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

Proper syntax:

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

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 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 examine at how they're organized 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 marking the entire document. Within the `html` element is the `head` element enclosing information about the document itself and the `body` element enclosing the content of 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 placed within 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.

Introducing Element Attributes

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.

Elements often contain one or more **element attributes** providing additional information 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>
```

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

For attributes that do not require a value, HTML supports **attribute minimization** by removing the attribute value completely. 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 HTML 5 differs from other markup languages such as XHTML in which minimization is not allowed and all attributes must have attribute values.

REFERENCE

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

An HTML file 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. The browser reading the HTML code 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 the same:

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

and

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

The browser also collapses consecutive occurrences of white-space characters into a single occurrence, so that the text of the paragraph in the following code is still treated as "Welcome to Curbside Thai", ignoring the extra white spaces between "Curbside" and "Thai".

```
<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.