3

Configuring Color and Text with CSS

Chapter Objectives

In this chapter, you will learn how to . . .

- Describe the evolution of style sheets from print media to the Web
- List advantages of using Cascading Style Sheets
- Configure background and text color on web pages
- Create style sheets that configure common color and text properties
- Apply inline styles
- Use embedded style sheets
- Use external style sheets
- Configure element, class, id, and descendant selectors
- Utilize the "cascade" in CSS
- Validate CSS

Now that you have been introduced to HTML, let's explore **Cascading Style Sheets (CSS)**. Web designers use CSS to separate the presentation style of a web page from the information on the web page. CSS is used to configure text, color, and page layout. CSS is not new—it was first proposed as a standard by the W3C in 1996. In 1998, additional properties for positioning web page elements were introduced to the language with CSS level 2 (CSS2), which was used for over a decade before reaching official "recommendation" status in 2011. CSS level 3 (CSS3) properties support features such as embedding fonts, rounded corners, and transparency. The W3C continues to evolve CSS, with proposals for CSS level 4 (CSS4) currently in draft form. This chapter introduces you to the use of CSS on the Web as you explore how to configure color and text.

3.1 Overview of Cascading Style Sheets

For years, style sheets have been used in desktop publishing to apply typographic styles and spacing instructions to printed media. CSS provides this functionality (and much more) for web developers. CSS allows web developers to apply typographic styles (typeface, font size, and so on) and page layout instructions to a web page. The CSS Zen Garden, http://www.csszengarden.com, exemplifies the power and flexibility of CSS. Visit this site for an example of CSS in action. Notice how the content looks dramatically different depending on the design (CSS style rules) you select. Although the designs on the CSS Zen Garden are created by CSS masters, at some point these designers were just like you—starting out with CSS basics.

CSS is a flexible, cross-platform, standards-based language developed by the W3C. The W3C's description of CSS can be found at http://www.w3.org/Style. Be aware that even though CSS has been in use for many years, it is still considered an emerging technology, and different browsers do not support it in exactly the same way. We concentrate on aspects of CSS that are well supported by popular browsers.

Advantages of Cascading Style Sheets

There are several advantages to using CSS (see Figure 3.1):

- **Typography and page layout can be better controlled.** These features include font size, line spacing, letter spacing, indents, margins, and element positioning.
- **Style is separate from structure.** The format of the text and colors used on the page can be configured and stored separately from the body section of the web page document.
- **Styles can be stored.** You can store styles in a separate document and associate them with the web page. When the styles are modified, the HTML remains intact. This means, for example, that if your client decides to change the background color of a set of web pages from red to white, you only need to change one file that contains the styles, instead of modifying each web page document.
- **Documents are potentially smaller.** The formatting is separate from the document; therefore, the actual documents should be smaller.
- **Site maintenance is easier.** Again, if the styles need to be changed, then it is possible to complete the modifications by changing the style sheet only.

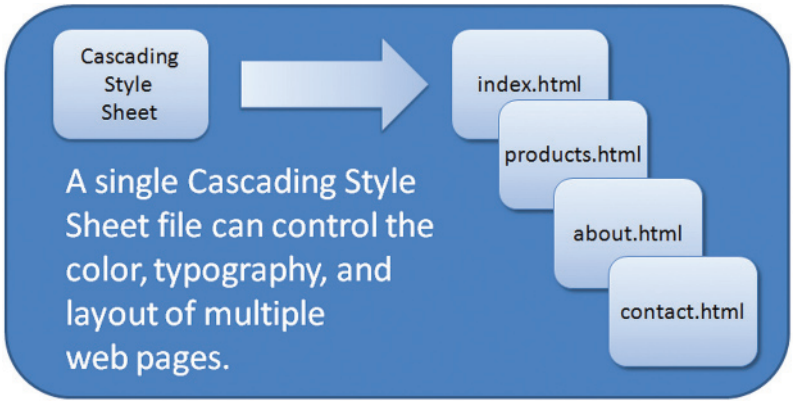

Figure 3.1 The power of a single CSS file

An issue to be aware of when using CSS is that CSS technology is still not uniformly supported by all browsers. See http://caniuse.com/#search=css3 for a list of CSS3 features supported by various browsers. This book will focus on aspects of CSS that are well supported by modern browsers and note when differences in syntax are needed.

Configuring Cascading Style Sheets

Web developers use four methods to incorporate CSS technology: inline, embedded, external, and imported.

- **Inline styles** are coded in the body of the web page as an attribute of an HTML tag. The style applies only to the specific element that contains it as an attribute.
- **Embedded styles** (also referred to as **internal styles**) are defined within a style element in the head section of a web page. These style instructions apply to the entire web page document.
- **External styles** are coded in a separate text file. This text file is associated with the web page by configuring a link element in the head section.
- **Imported styles** are similar to external styles in that they can connect styles coded in a separate text file with a web page document. An external style sheet can be imported into embedded styles or into another external style sheet by using the @import directive.

CSS Selectors and Declarations

Style sheets are composed of style rules that describe the styling to be applied. Each **rule** has two parts: a **selector** and a **declaration**:

- **CSS Style Rule Selector** The selector can be an HTML element name, a class name, or an id name. In this section, we will focus on applying styles to element name selectors. We will work with class selectors and id selectors later in this chapter.
- **CSS Style Rule Declaration** The declaration indicates the CSS **property** you are setting (such as color) and the value you are assigning to the property.

For example, the CSS rule shown in Figure 3.2 would set the color of the text used on a web page to blue. The selector is the body tag, and the declaration sets the color property to the value of blue.

Figure 3.2 Using CSS to set the text color to blue

The `background-color` Property

The CSS **`background-color` property** configures the background color of an element. The following style rule will configure the background color of a web page to be yellow. Notice how the declaration is enclosed within braces and how the colon symbol (:) separates the declaration property and the declaration value.

```
body { background-color: yellow }
```

The `color` Property

The CSS **`color property`** configures the text (foreground) color of an element. The following CSS style rule will configure the text color of a web page to be blue:

```
body { color: blue }
```

Configure Background and Text Color

Figure 3.3 displays a web page with a white text and an orchid background. To configure more than one property for a selector, use a semicolon (;) to separate the declarations as follows:

```
body { color: white; background-color: orchid; }
```

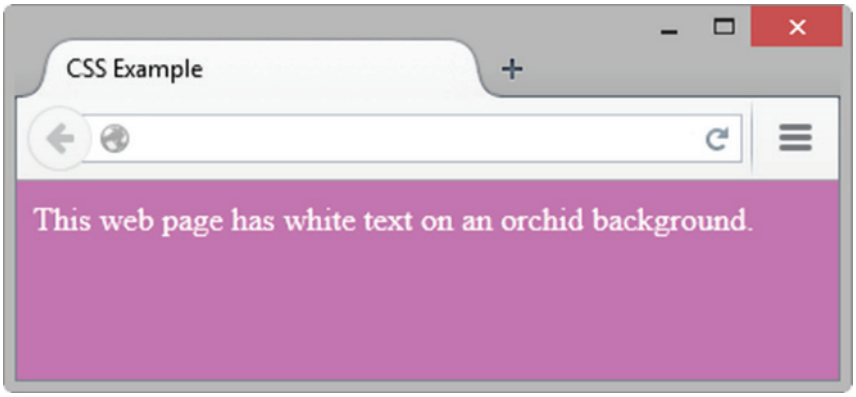

Figure 3.3
A web page with orchid background color and white text color

The spaces in these declarations are optional. The ending semicolon (;) is also optional, but useful in case you need to add additional style rules at a later time. The following code samples are also valid:

```
body {color:white;background-color:orchid}
body { color: white;
       background-color: orchid; }
body {
     color: white;
     background-color: orchid;
}
```

You might be asking yourself how you would know what properties and values can be used. See the CSS Property Reference in Appendix E for a detailed list of CSS properties. This chapter introduces you to some of the CSS properties commonly used to configure color and text, shown in Table 3.1. In the next sections, we'll take a look at how color is used on web pages.

Table 3.1 CSS properties introduced in this chapter

Property	Description	Values
`background-color`	Background color of an element	Any valid color
`color`	Foreground (text) color of an element	Any valid color
`font-family`	Name of a font or font family	Any valid font or a font family such as serif, sans-serif, fantasy, monospace, or cursive
`font-size`	Size of the font	Varies; a numeric value with pt (standard font **point** sizes) or px (**pixels**) units or the unit em (which corresponds to the width of the uppercase M of the current font); a numeric percentage; and the text values xx-small, x-small, small, medium, large, x-large, and xx-large
`font-style`	Style of the font	normal, italic, or oblique
`font-weight`	The "boldness" or weight of the font	Varies; the text values normal, bold, bolder, and lighter and the numeric values 100, 200, 300, 400, 500, 600, 700, 800, and 900
`letter-spacing`	The space between characters	A numeric value (px or em) or normal (default)
`line-height`	The spacing allowed for the line of text	It is most common to use a percentage for this value; for example, a value of 200% would correspond to double-spacing.
`margin`	Shorthand notation to configure the margin surrounding an element	A numeric value (px or em); for example, body {margin: 10px} will set the page margins in the document to 10 pixels. When eliminating the margin, do not use the px or em unit—for example, body {margin:0}
`margin-left`	Configures the space in the left margin of the element	A numeric value (px or em), auto, or 0
`margin-right`	Configures the space in the right margin of the element	A numeric value (px or em), auto, or 0
`text-align`	The alignment of text	center, justify, left, or right
`text-decoration`	Determines whether text is underlined; this style is most often applied to hyperlinks	The value "none" will cause a hyperlink not to be underlined in a browser that normally processes in this manner
`text-indent`	Configures the indentation of the first line of text	Numeric value (px or em) or percentage
`text-shadow`	Configures a drop shadow on the text displayed within an element. This CSS3 property is not supported in all browsers.	Two to four numerical values (px or em) to indicate horizontal offset, vertical offset, blur radius (optional), and spread distance (optional), and a valid color value.
`text-transform`	Configures the capitalization of text	none (default), capitalize, uppercase, or lowercase
`white-space`	Configures the display of whitespace	normal (default), nowrap, pre, pre-line, pre-wrap
`width`	The width of the content of an element	A numeric value (px or em), numeric percentage, or auto (default)
`word-spacing`	The space between words	A numeric value (px or em) or normal (default)

3.2 Using Color on Web Pages

Monitors display color as a combination of different intensities of red, green, and blue, a concept known as **RGB color**. RGB intensity values are numerical from 0 to 255. Each RGB color has three values, one each for red, green, and blue. These values are always listed in the same order (red, green, blue) and specify the numerical value of each color

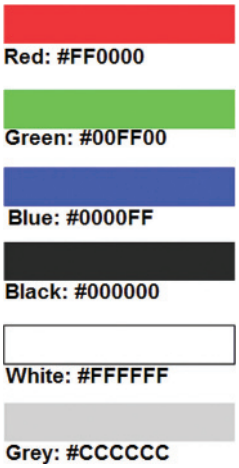

Figure 3.4 Color swatches and hexadecimal color values

used (see the examples in Figure 3.4). You will usually use hexadecimal color values to specify RGB color on web pages.

Hexadecimal Color Values

Hexadecimal is the name for the base-16 numbering system, which uses the characters `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `A`, `B`, `C`, `D`, `E`, and `F` to specify numeric values. **Hexadecimal color values** specify RGB color with numeric value pairs ranging from `00` to `FF` (`0` to `255` in base `10`). Each pair is associated with the amount of red, green, and blue displayed. Using this notation, one would specify the color red as `#FF0000` and the color blue as `#0000FF`. The `#` symbol signifies that the value is hexadecimal. You can use either uppercase or lowercase letters in hexadecimal color values; `#FF0000` and `#ff0000` both configure the color red.

Don't worry—you won't need to do calculations to work with web colors. Just become familiar with the numbering scheme. See Figure 3.5 for an excerpt from the color chart at http://webdevfoundations.net/color.

Figure 3.5 Partial color chart

Web-Safe Colors

Back in the day of 8-bit color monitors, web page color could be problematic and it was important to use one of the 216 **web-safe colors**, which display in a similar manner on both the Mac and PC platforms. The hexadecimal color values of web-safe colors use the numerals `00`, `33`, `66`, `99`, `CC`, and `FF`. The 216 web-safe colors make up the **Web-Safe Color Palette**, shown in Appendix H (also at http://webdevfoundations.net/color). Now that most monitors display millions of colors, using web-safe colors is less important. The Web-Safe Color Palette is rather limited, and it is common for today's web designers to choose colors creatively rather than select them only from the palette.

CSS Color Syntax

CSS syntax allows you to configure colors in a variety of ways:

- color name
- hexadecimal color value
- hexadecimal shorthand color value
- decimal color value (RGB triplet)
- HSL (Hue, Saturation, and Lightness) color value notation new to CSS3; see http://www.w3.org/TR/css3-color/#hsl-color

Visit http://meyerweb.com/eric/css/colors/ to view a chart with examples of configuring color values using different notations. We'll typically use hexadecimal color values in this book. Table 3.2 shows CSS syntax examples that configure a paragraph with red text.

Table 3.2 CSS color syntax examples

CSS Syntax	Color Type
`p { color: red; }`	Color name
`p { color: #FF0000; }`	Hexadecimal color value
`p { color: #F00; }`	Shorthand hexadecimal (one character for each hexadecimal pair; used only with web-safe colors)
`p { color: rgb(255,0,0); }`	Decimal color value (RGB triplet)
`p { color: hsl(0, 100%, 50%); }`	HSL color values

FAQ **Are there other methods to configure color with CSS?**

Yes, the CSS3 Color Module provides a way for web developers to configure not only color but also the transparency of the color with RGBA (Red, Green, Blue, Alpha). and HSLA (Hue, Saturation, Lightness, Alpha) color. Also new to CSS3 is the opacity property, and CSS gradient backgrounds. You'll explore these techniques in Chapter 4.

FAQ **How do I choose a color scheme for a web page?**

There's a lot to consider when you select a color scheme for a website. The colors that you choose set the tone and help to create a web presence for the company or organization that appeals to the target audience. The colors you choose for text and background need to have good contrast in order to be readable. We'll explore techniques for choosing a color scheme in Chapter 5.

3.3 Inline CSS with the Style Attribute

Recall that there are four methods for configuring CSS: inline, embedded, external, and imported. In this section, we focus on inline CSS using the style attribute.

The Style Attribute

Inline styles are coded as an attribute on an HTML tag using the **style attribute**. The value of the style attribute is set to the style rule declaration that you need to configure. Recall that a declaration consists of a property and a value. Each property is separated from its value with a colon (:). The following code will use inline styles to set the text color of an `<h1>` tag to a shade of red:

```
<h1 style="color:#cc0000">This is displayed as a red heading</h1>
```

If there is more than one property, they are separated by a semicolon (;). The following code configures the heading with a red text color and a gray background color:

```
<h1 style="color:#cc0000;background-color:#cccccc">
This is displayed as a red heading on a gray background</h1>
```

Hands-On Practice 3.1

In this Hands-On Practice, you will configure a web page with inline styles. The inline styles will specify the following:

- Global body tag styles for an off-white background with teal text. These styles will be inherited by other elements by default. For example:

```
<body style="background-color:#F5F5F5;color:#008080;">
```

- Styles for an h1 element with a teal background with off-white text. This style will override the global styles configured on the body element. For example:

```
<h1 style="background-color:#008080;color:#F5F5F5;">
```

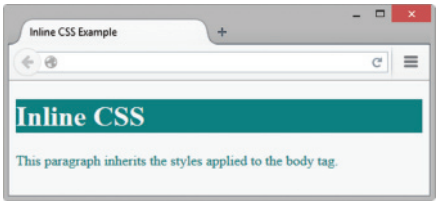

Figure 3.6 Web page using inline styles

A sample is shown in Figure 3.6. Launch a text editor. Select File > Open to edit the template file located at chapter2/template.html in the student files. Modify the title element, and add heading tags, paragraph tags, style attributes, and text to the body section as indicated by the following highlighted code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Inline CSS Example</title>
<meta charset="utf-8">
</head>
<body style="background-color:#F5F5F5;color:#008080;">
<h1 style="background-color:#008080;color:#F5F5F5;">Inline CSS</h1>
<p>This paragraph inherits the styles applied to the body tag.</p>
</body>
</html>
```

Save the document as inline.html on your hard drive or flash drive. Launch a browser to test your page. It should look similar to the page shown in Figure 3.6. Note that the inline styles applied to the body tag are inherited by other elements on the page (such as the paragraph) unless more-specific styles are specified (such as those coded on the `<h1>` tag). You can compare your work with the solution found in the student files (chapter3/inline.html).

Let's continue and add another paragraph, with the text color configured to be dark gray:

```
<p style="color:#333333">This paragraph overrides the text color style
applied to the body tag.</p>
```

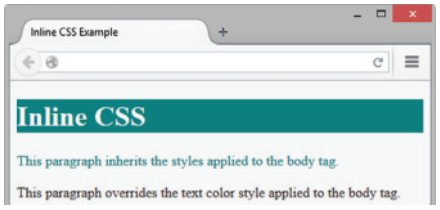

Figure 3.7 The second paragraph's inline styles override the global styles configured on the body tag

Save the document as inlinep.html. It should look similar to the page shown in Figure 3.7. You can compare your work with the solution found in the student files (chapter3/inlinep.html). Note that the inline styles applied to the second paragraph override the global styles applied to the body of the web page.

FAQ **Are inline styles recommended?**

While inline styles can sometimes be useful, you'll find that you won't use this technique much in practice—it's inefficient, adds extra code to the web page document, and is inconvenient to maintain. However, inline styles can be quite handy in some circumstances, such as when you post an article to a content management system or blog and need to tweak the sitewide styles a bit to help get your point across.

3.4 Embedded CSS with the Style Element

In the previous Hands-On Practice, you added inline styles for one of the paragraphs. To do so, you coded a style attribute on the paragraph element. But what if you needed to configure the styles for 10 or 20 paragraphs instead of just one? Using inline styles, you might be doing a lot of repetitive coding! While inline styles apply to one HTML element, embedded styles apply to an entire web page.

Style Element

Embedded styles are placed within a **style element** located in the head section of a web page. The opening `<style>` tag and the closing `</style>` tag contain the list of embedded style rules. When using XHTML syntax, the `<style>` tag requires a **type attribute** with the value of `"text/css"` to indicate the MIME type. HTML5 syntax does not require the type attribute.

The web page in Figure 3.8 uses embedded styles to set the text color and background color of the web page document with the body element selector. See the example in the student files at chapter3/embed.html. The code follows:

Figure 3.8 Web page with embedded styles

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Embedded Styles</title>
<meta charset="utf-8">
<style>
body { background-color: #E6E6FA;
       color: #191970;
}
</style>
</head>
<body>
<h1>Embedded CSS</h1>
<p>This page uses embedded styles.</p>
</body>
</html>
```

Notice the way the style rules were coded, with each rule on its own line. This formatting is not required for the styles to work, but it makes the styles more readable and easier to maintain than one long row of text. The styles are in effect for the entire web page document because they were applied to the `<body>` tag using the body element selector.

Hands-On Practice 3.2

Launch a text editor, and open the starter.html file from the chapter3 folder in the student files. Save your page as embedded.html, and test it in a browser. Your page should look similar to the one shown in Figure 3.9.

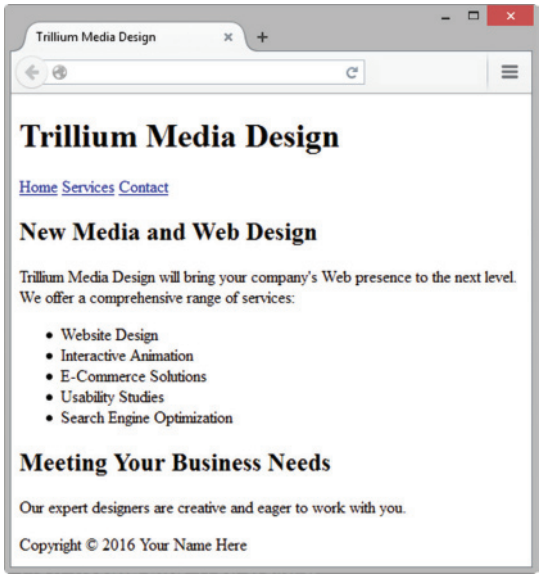

Figure 3.9 The web page without any styles

Open the file in a text editor, and view the source code. Notice that the web page code uses the `<header>`, `<nav>`, `<main>`, `<footer>`, `<h1>`, `<h2>`, `<p>`, `<ul>`, and `<li>` elements. In this Hands-On Practice, you will code embedded styles to configure selected background and text colors. You will use the body element selector to configure the default background color (#e6e6fa) and default text color (#191970) for the entire page. You will also use the h1 and h2 element selectors to configure different background and

text colors for the heading areas. Edit the embedded.html file in a text editor, and add the following code below the <title> element in the head section of the web page:

```
<style>
body { background-color: #e6e6fa; color: #191970; }
h1 { background-color: #191970; color: #e6e6fa; }
h2 { background-color: #aeaed4; color: #191970; }
</style>
```

Save your file, and test it in a browser. Figure 3.10 displays the web page along with corresponding color swatches. A monochromatic color scheme was chosen. Notice how the repetition of a limited number of colors unifies the design of the web page. View the source code for your page, and review the CSS and HTML code. An example of this web page is in the student files at chapter3/3.2/embedded.html. Note that all the styles were located in a single place on the web page. Since embedded styles are coded in a specific location, they are easier to maintain over time than inline styles. Also, notice that you coded the styles for the h2 element selector only once (in the head section), and *both* of the <h2> elements applied the h2 style. This approach is more efficient than coding the same inline style on each <h2> element. However, it is uncommon for a website to have only one page. Repeating the CSS in the head section of each web page file is inefficient and difficult to maintain. In the next section, you'll use a more productive approach—configuring an external style sheet.

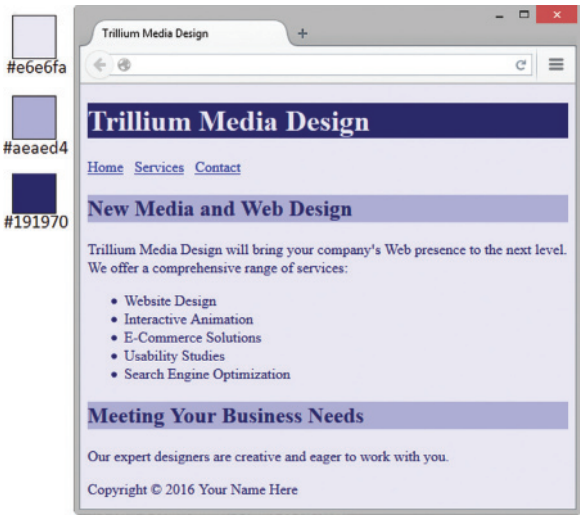

Figure 3.10 The web page after embedded styles are configured

FAQ My CSS doesn't work; what can I do?

Coding CSS is a detail-oriented process. There are several common errors that can cause the browser not to apply CSS correctly to a web page. With a careful review of your code and the following tips, you should get your CSS working:

- Verify that you are using the colon (:) and semicolon (;) symbols in the right spots—they are easy to confuse. The colon should separate the properties from their values, while the semicolon should be placed between each `property:value` configuration.
- Check that you are not using equal (=) signs instead of colons (:) between each property and its value.

- Verify that curly braces ({ and }) are properly placed around the style rules for each selector.
- Check the syntax of your selectors, the selectors' properties, and the property values for correct usage.
- If part of your CSS works and part doesn't, read through the CSS and determine the first rule that is not applied. Often, the error is in the rule above the rule that is not applied.
- Use a validation application to check your CSS code. The W3C has a free CSS code validator at http://jigsaw.w3.org/css-validator. The W3C's CSS validator can help you find syntax errors. See Section 3.11 for an overview of how to use this tool to validate your CSS.

Checkpoint 3.1

1. List three reasons to use CSS on a web page.
2. When designing a page that uses colors other than the default colors for text and background, explain why it is a good reason to configure both the text color and the background color.
3. Describe one advantage to using embedded styles instead of inline styles.

3.5 Configuring Text with CSS

In Chapter 2, you discovered how to use HTML to configure some characteristics of text on web pages, including phrase elements such as the `<strong>` element. You have also already configured text color using the CSS color property. In this section, you will learn to use CSS to configure font typeface. Using CSS to configure text is more flexible (especially when using an external style sheet, as you will discover later in the chapter) than using HTML elements and is the method preferred by modern web developers.

The `font-family` Property

The **`font-family` property** configures font typeface. A web browser displays text using the fonts that have been installed on the user's computer. When a font is specified that is not installed on your web visitor's computer, the default font is substituted. Times New Roman is the default font displayed by most web browsers. Figure 3.11 shows font family categories.

The Verdana, Tahoma, and Georgia font typefaces were specifically designed to display well on computer monitors. A common practice is to use a serif font (such as Georgia or Times New Roman) for headings and a sans-serif font (such as Verdana or Arial) for detailed text content. Not every computer has the same fonts installed. See http://www.ampsoft.net/webdesign-l/WindowsMacFonts.html for a list of web-safe fonts. Create a built-in backup plan by listing multiple fonts and categories for the value of the font-family property. The browser will attempt to use the fonts in the order listed. The following CSS configures the p element selector to display text in Arial (if installed), Helvetica (if installed), or the default installed sans-serif font:

```
p { font-family: Arial, Helvetica, sans-serif; }
```

Font Family Category	Font Family Description	Font Typeface Examples
serif	Serif fonts have small embellishments on the end of letter strokes; often used for headings.	Times New Roman, Georgia, Palatino
sans-serif	Sans-serif fonts do not have serifs; often used for web page text.	Arial, Tahoma, Helvetica, Verdana
monospace	Fixed-width font; often used for code samples.	Courier New, Lucida Console
cursive	Hand-written style; use with caution; may be difficult to read on a web page.	Lucida Handwriting, Brush Script, Comic Sans MS
fantasy	Exaggerated style; use with caution; sometimes used for headings; may be difficult to read on a web page.	Jokerman, Impact, Papyrus

Figure 3.11
Common fonts

Hands-On Practice 3.3

Launch a text editor, and open the starter2.html file from the chapter3 folder in the student files.

Locate the style tags in the head section, and code embedded CSS to style the following:

1. Configure the body element selector to set global styles to use a sans-serif font typeface, such as Verdana or Arial. An example is

```
body { font-family: Verdana, Arial, sans-serif; }
```

2. Configure h2 and h3 element selectors to use a serif font typeface, such as Georgia or Times New Roman. You can configure more than one selector in a styles rule by placing a comma before each new selector. Notice that "Times New Roman" is enclosed within quotation marks because the font name is more than a single word. Code the following style rule:

```
h2, h3 { font-family: Georgia, "Times New Roman", serif; }
```

Save your page as index.html in a folder named kayak3. Launch a browser and test your page. It should look similar to the one shown in Figure 3.12. A sample solution is in the chapter3/3.3 folder.

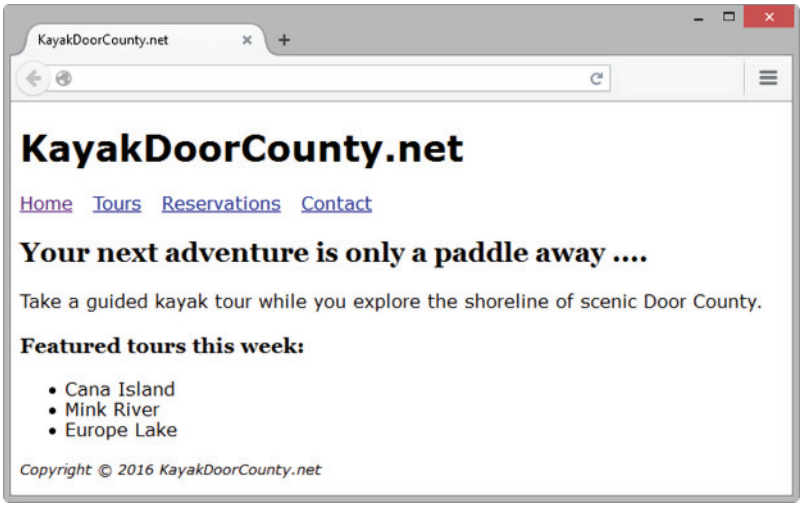

Figure 3.12 The new home page

FAQ I've heard about "embedding" fonts in order to use special fonts on a web page—what's that all about?

For many years, web designers have been limited to a set of common fonts for text on web pages. CSS3 introduced `@font-face`, which can be used to "embed" other fonts within web pages although you actually provide the location of the font and the browser downloads it. For example, if you own the rights to freely distribute the font named MyAwesomeFont and it is stored in a file myawesomefont.woff in the same folder as your web page, the following CSS will make it available to your web page visitors:

```
font-face { font-family: MyAwesomeFont;
            src: url(myawesomefont.woff) format("woff"); }
```

After you code the `@font-face` rule, you can apply that font to a selector in the usual way, such as in the following example that configures h1 elements:

```
h1 { font-family: MyAwesomeFont, Georgia, serif; }
```

Current browsers support `@font-face` but there can be copyright issues. When you purchase a font for use on your own computer you do not necessarily purchase the right to freely distribute it. Visit http://www.fontsquirrel.com to browse a selection of commercial-use free fonts available for download and use.

Google Web Fonts provides a collection of free hosted embeddable web fonts. Explore the fonts available at http://www.google.com/webfonts. Once you choose a font, all you need to do is:

1. Copy and paste the link tag provided by Google in your web page document. (The link tag associates your web page with a CSS file that contains the appropriate @ font-face rule.)
2. Configure your CSS font-family property with the Google web font name.

See the Getting Started guide at for more information https://developers.google.com/fonts/docs/getting_started for more information. Use web fonts judiciously to conserve bandwidth and avoid applying multiple web fonts to a web page. It's a good idea to use just one web font on a web page along with your typical fonts. This can provide you a way to use an uncommon font typeface in page headings and/or navigation without the need to create graphics for these page areas.

More CSS Text Properties

CSS provides you with lots of options for configuring the text on your web pages. In this section, you will explore the `font-size`, `font-weight`, `font-style`, `line-height`, `text-align`, `text-decoration`, `text-indent`, `letter-spacing`, `word-spacing`, `text-shadow`, and `text-transform` properties.

The `font-size` Property

The **`font-size` property** sets the size of the font. Table 3.3 lists a wide variety of text and numeric values—there are almost too many choices available. See the notes in Table 3.3 for recommended use.

Table 3.3 Configuring font size

Value Category	Values	Notes
Text Value	xx-small, x-small, small, medium (default), large, x-large, xx-large	Scales well when text is resized in browser; limited options for text size
Pixel Unit (px)	Numeric value with unit, such as 10 px	Pixel-perfect display depends on screen resolution; may not scale in every browser when text is resized
Point Unit (pt)	Numeric value with unit, such as 10 pt	Use to configure print version of web page (see Chapter 7); may not scale in every browser when text is resized
Em Unit (em)	Numeric value with unit, such as .75 em	Recommended by W3C; scales well when text is resized in browser; many options for text size
Percentage Value	Numeric value with percentage, such as 75%	Recommended by W3C; scales well when text is resized in browser; many options for text size

The **em unit** is a relative font unit that has its roots in the print industry, dating back to the day when printers set type manually with blocks of characters. An em unit is the width of a square block of type (typically the uppercase M) for a particular font and type size. On web pages, an em unit corresponds to the width of the font and size used in the parent element (typically the body element). So, the size of an em unit is relative to the font typeface and default size. Percentage values work in a similar manner to em units. For example, `font-size: 100%` and `font-size: 1em` should render the same in a browser. To compare font sizes on your computer, launch a browser and view chapter3/fonts.html in the student files.

The `font-weight` Property

The **`font-weight` property** configures the boldness of the text. Configuring the CSS rule `font-weight: bold;` has a similar effect as the `<strong>` or `<b>` HTML element.

The `font-style` Property

The **`font-style` property** typically is used to configure text displayed in italics. Valid values for font-style are normal (the default), italic, and oblique. The CSS font-style: italic; has the same visual effect in the browser as an `<i>` or `<em>` HTML element.

The `line-height` Property

The **`line-height` property** modifies the default height of a line of text and is often configured with a percentage value. For example, code `line-height: 200%;` to configure text to appear double spaced.

The `text-align` Property

HTML elements are left-aligned by default; They begin at the left margin. The CSS **`text-align` property** configures the alignment of text and inline elements within block display elements such as headings, paragraphs, and divs. The values for the text-align

property are left (default), right, and center. The following CSS code sample configures an h1 element to have centered text:

```
h1 { text-align: center; }
```

While it can be quite effective to center the text displayed in web page headings, be careful about centering text in paragraphs. According to WebAIM (http://www.webaim.org/techniques/textlayout), studies have shown that centered text is more difficult to read than left-aligned text.

The `text-indent` Property

The CSS **`text-indent` property** configures the indentation of the first line of text within an element. The value can be numeric (such as a px, pt, or em unit) or a percentage. The following CSS code sample configures the first line of all paragraphs to be indented:

```
p { text-indent: 5em; }
```

The `text-decoration` Property

The purpose of the CSS **`text-decoration` property** is to modify the display of text. Commonly used values for the text-decoration property include none, underline, overline, and line-through. Did you ever wonder why some hyperlinks are not underlined? Although hyperlinks are underlined by default, you can remove the underline with the text-decoration property. The following code sample removes the underline on a hyperlink:

```
a { text-decoration: none; }
```

The `text-transform` Property

The **`text-transform` property** configures the capitalization of text. Valid values for text-transform are none (default), capitalize, uppercase, or lowercase. The following code sample causes all the text within an h3 element to be displayed in uppercase:

```
h3 { text-transform: uppercase; }
```

The `letter-spacing` Property

The **`letter-spacing` property** configures the space between text characters. Valid values for letter-spacing are normal (default) and a numeric pixel or em unit. The following code sample configures extra spacing between characters within an `h3` element:

```
h3 { letter-spacing: 3px; }
```

The `word-spacing` Property

The **`word-spacing` property** configures the space between words. Valid values for word-spacing are normal (default) and a numeric pixel or em unit. The following code sample configures extra spacing between words within an `h3` element:

```
h3 { word-spacing: 2em; }
```

The white-space Property

The **white-space property** specifies the way that whitespace (such as a space character, or line feed within code) is displayed by the browser. The default browser behavior is to collapse adjacent whitespace to a single space character. Commonly used values for white-space are normal (default), nowrap (text will not wrap to the next line), and pre (preserves all whitespace in the browser display).

CSS3 text-shadow Property

The CSS3 **text-shadow property** adds depth and dimension to text displayed on web pages. Current versions of modern browsers, including Internet Explorer (version 10 and later) support the text-shadow property. Configure a text shadow by coding values for the shadow's horizontal offset, vertical offset, blur radius (optional), and color:

- **Horizontal offset.** Use a numeric pixel value. Positive value configures a shadow on the right. Negative value configures a shadow on the left.
- **Vertical offset.** Use a numeric pixel value. Positive value configures a shadow below. Negative value configures a shadow above.
- **Blur radius (optional).** Configure a numeric pixel value. If omitted, defaults to the value 0 which configures a sharp shadow. Higher values configure more blur.
- **Color value.** Configure a valid color value for the shadow.

Here's an example that configures a dark gray shadow with 3px horizontal offset, 2px vertical offset, and 5px blur radius.:

```
text-shadow: 3px 2px 5px #666;
```

FAQ Why didn't the text-shadow property work in Internet Explorer 9?

Not all browsers and browser versions support the new CSS3 properties like text-shadow. Browser support changes with each new browser version. There is no substitute for thoroughly testing your web pages. Browser support lists are available at: http://www.findmebyip.com/litmus and http://www.quirksmode.org/css/contents.html.

Hands-On Practice 3.4

Now that you've got a collection of new CSS properties for font and text configuration, let's try them out. You will use the file from Hands-On Practice 3.2 (see the student files chapter3/3.2/index.html) as a starting point. Launch a text editor, and open the file. You will now code additional CSS styles to configure the text on the page.

Set Default Font Properties for the Page

As you have already seen, CSS rules applied to the body selector apply to the entire page. Modify the CSS for the body selector to display text using a sans-serif font. The new font typeface style declaration shown in the following code will apply to the entire

web page unless more specific style rules are applied to an element selector (such as h1 or p), a class, or an id (more on classes and ids later):

```
body { background-color: #E6E6FA;
       color: #191970;
       font-family: Arial, Verdana, sans-serif;  }
```

Save your page as embedded1.html, and test it in a browser. Your page should look similar to the one shown in Figure 3.13. Notice that just a single line of CSS changed the font typeface of all the text on the page!

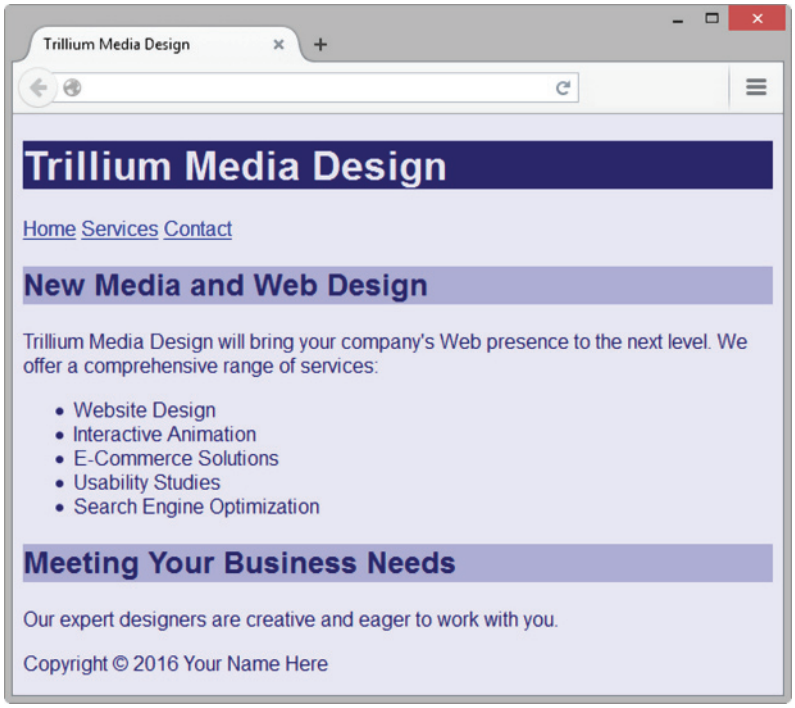

Figure 3.13 CSS configures the font on the web page

Configure the h1 Selector

Now you will configure the `line-height`, `font-family`, `text-indent`, and `text-shadow` CSS properties. Set the `line-height` property to 200%; this will add a bit of empty space above and below the heading text. (In Chapters 4 and 6, you will explore other CSS properties, such as the margin, border, and padding, that are more commonly used to configure space surrounding an element.) Next, modify the h1 selector to use a serif font. When a font name contains spaces, type quotes as indicated in the code that follows. While it is generally recognized that blocks of text using sans-serif fonts are easier to read, it is common to use a serif font to configure page or section headings. Indent the text 1em unit. Configure a gray (#CCCCCC) text shadow with a 3 pixel vertical offset, 3 pixel horizontal offset, and 5 pixel blur radius.

```
h1 { background-color: #191970;
     color: #E6E6FA;
     line-height: 200%;
     font-family: Georgia, "Times New Roman", serif;
     text-indent: 1em;
     text-shadow: 3px 3px 5px #CCCCCC;  }
```

Save your page, and test it in a browser.

Configure the h2 Selector

Configure the CSS rule to use the same font typeface as the h1 selector and to display centered text.

```
h2 { background-color: #AEAED4;
     color: #191970;
     font-family: Georgia, "Times New Roman", serif;
     text-align: center;  }
```

Configure the Navigation Area

The navigation links would be more prominent if they were displayed in a larger and bolder font. Code a selector for the nav element that sets the font-size, font-weight, and word-spacing properties.

```
nav { font-weight: bold;
      font-size: 1.25em;
      word-spacing: 1em;}
```

Configure the Paragraphs

Edit the HTML, and remove the line break tag that is after the first sentence of each paragraph; these line breaks look a bit awkward. Next, configure text in paragraphs to display just slightly smaller than the default text size. Use the `font-size` property set to .90em. Configure the first line of each paragraph to be indented. Use the `text-indent` property to configure a 3em indent.

```
p { font-size: .90em;
    text-indent: 3em; }
```

Configure the Unordered List

Configure the text displayed in the unordered list to be bold.

```
ul { font-weight: bold; }
```

Save your page as index.html in a folder named trillium3. Test it in a browser. Your page should look similar to the one shown in Figure 3.14. The student files contain a sample solution at chapter3/3.4/index.html. CSS is quite powerful—just a few lines of code significantly changed the appearance of the web page. You may be wondering if even more customization is possible. For example, what if you did not want all the

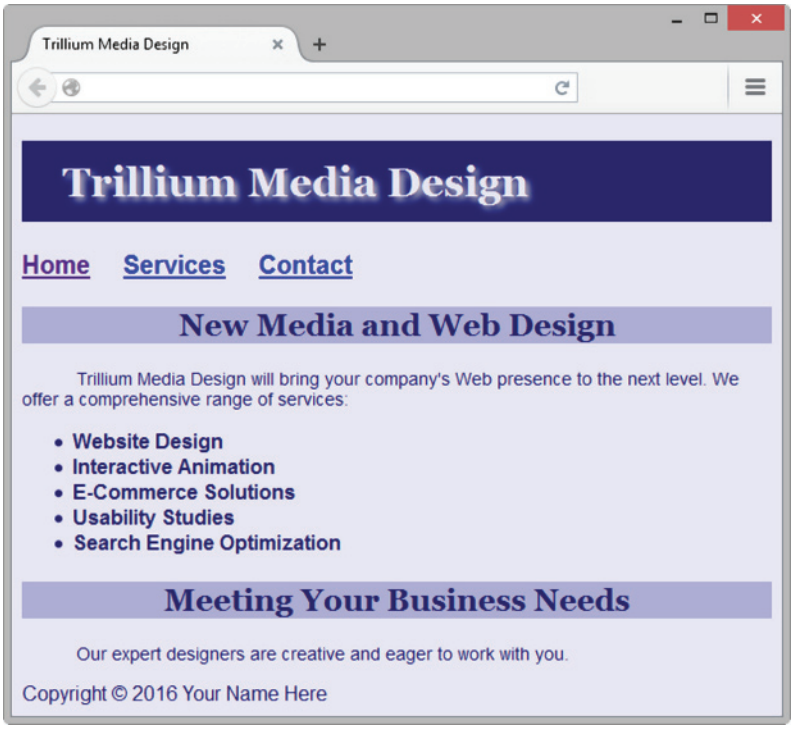

Figure 3.14 CSS configures color and text properties on the web page

paragraphs to display in exactly the same way? While you could add inline styles to the web page code, that is usually not the most efficient technique. The next section introduces the CSS class and id selectors, which are widely utilized to configure specific page elements.

FAQ **Is there a quick way to apply the same styles to more than one HTML tag or more than one class?**

Yes, you can apply the same style rules to multiple selectors (such as HTML elements, classes, or ids) by listing the selectors in front of the style rule. Place a comma between each selector. The following code sample shows the `font-size` of 2em being applied to both the paragraph and list item elements:

```
p, li { font-size: 2em; }
```

3.6 CSS Class, Id, and Descendant Selectors

The Class Selector

Use a CSS **class selector** when you need to apply a CSS declaration to certain elements on a web page and not necessarily tie the style to a particular HTML element. See Figure 3.15, and notice that the last two items in the unordered list are displayed in a different color than the others; this is an example of using a class. When setting a style for a class, configure the class name as the selector. Place a dot, or period (.), in front of the

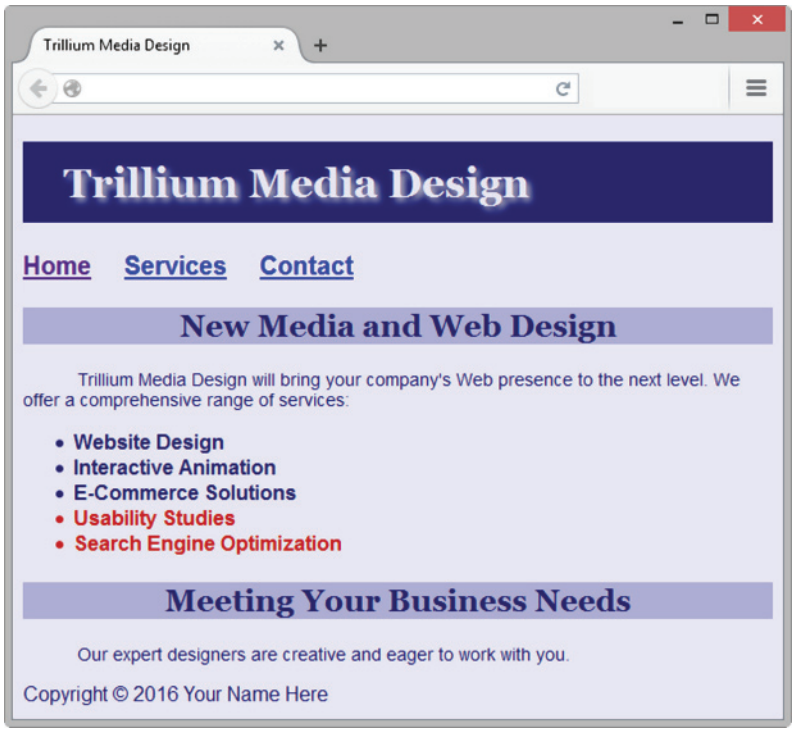

Figure 3.15 Using a class selector

class name in the style sheet. The following code configures a class called `feature` in a style sheet with a foreground (text) color set to a medium red:

```
.feature { color: #C70000; }
```

The styles set in the new class can be applied to any element you wish. You do this by using the **class attribute**, such as `class="feature"`. Do not type the dot in front of the class value in the opening tag where the class is being applied. The following code will apply the `feature` class styles to two `<li>` elements:

```
<li class="feature">Usability Studies</li>
<li class="feature">Search Engine Optimization</li>
```

The Id Selector

Use a CSS **id selector** to identify and apply a CSS rule uniquely to a *single* area on a web page. Unlike a class selector which can be applied multiple times on a web page, an id may only be applied once per web page. When setting a style for an id, place a hash mark (#) in front of the id name in the style sheet. An id name may contain letters, numbers, hyphens, and underscores. Id names may not contain spaces. The following code will configure an id called `feature` in a style sheet:

```
#feature { color: #333333; }
```

The styles set in the feature id can be applied to any element you wish by using the **id attribute,** `id="feature"`. Do not type the # in front of the id value in the opening tag. The following code will apply the `feature` id styles to a div tag:

```
<div id="feature">This sentence will be displayed using styles
configured in the feature id.</div>
```

Using CSS with an id selector is similar to using CSS with a class selector. Use an id selector to configure a single element on a web page. Use a class selector to configure one or more elements on a web page.

The Descendant Selector

Use a CSS **descendant selector** when you want to specify an element within the context of its container (parent) element. Using descendant selectors can help you to reduce the number of different classes and ids but still allows you to configure CSS for specific areas on the web page. To configure a descendant selector, list the container selector (which can be an element selector, class, or id) followed by the specific selector you are styling. For example, to specify a green text color for paragraphs located *within* the main element, code the following style rule:

```
main p { color: #00ff00; }
```

Hands-On Practice 3.5

In this Hands-On Practice, you will use the Trillium Media Design file from Hands-On Practice 3.4 (see the student files chapter3/3.4/index.html) as a starting point and modify the CSS and the HTML in the page to configure the navigation hyperlinks, content area, and page footer area. Launch a text editor, and open the index.html file.

Configure the Navigation Hyperlinks

Navigation hyperlinks are often displayed on web pages without the default underline by configuring the text-decoration property and applying that property to a descendant selector that targets only the anchor tags in the navigation area. Configure embedded CSS before the closing style tag. Code a descendant selector that specifies the hyperlinks with the nav element and set the text-decoration property.

```
nav a { text-decoration: none; }
```

Configure the Content Area

Trillium Media Design would like to draw attention to its new usability and search optimization services. Configure embedded CSS before the closing style tag. Create a class named `feature` that configures the text color to be a medium dark red (#C70000).

```
.feature { color: #C70000; }
```

Modify the last two items in the unordered list. Add a class attribute to each opening li tag that associates the list item with the `feature` class as follows:

```
<li class="feature">Usability Studies</li>
<li class="feature">Search Engine Optimization</li>
```

Configure the Footer Area

Configure embedded CSS before the closing style tag. Code a selector for the footer element that sets the text color, font-size, and font-style properties.

```
footer { color: #333333;
         font-size: .75em;
         font-style: italic; }
```

Modify the HTML and code a yourfirstname@yourlastname.com e-mail hyperlink (refer to Chapter 2) within the footer element on a line below the copyright information.

Save your file and test it in a browser. Your page should look similar to the image shown in Figure 3.16. See chapter3/3.5/index.html in the student files for a sample solution. Notice how the footer element, class, and nav hyperlinks styles are applied. Although the hyperlinks in the navigation area do not display with an underline, the e-mail hyperlink in the footer area displays with the default underline.

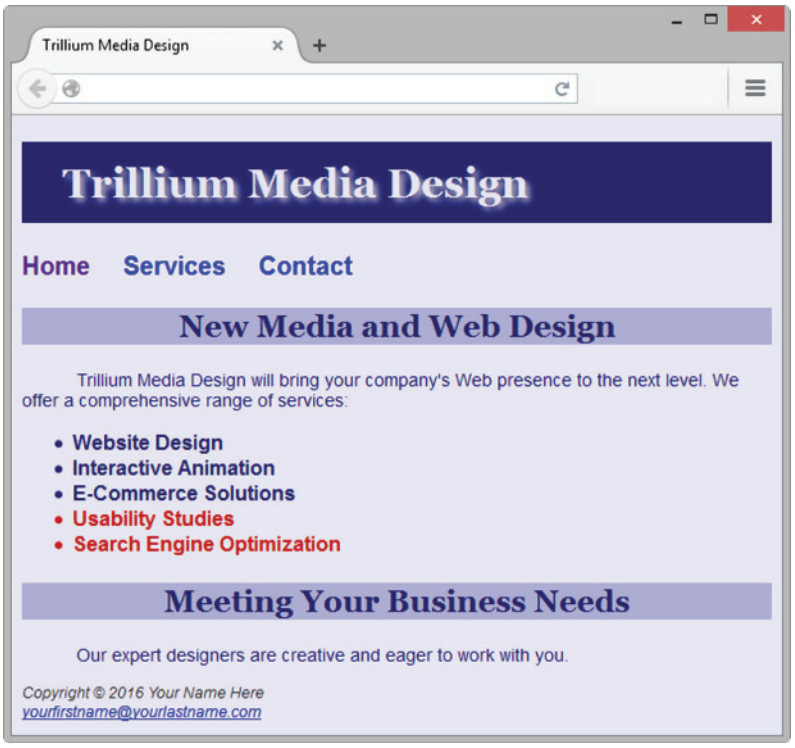

Figure 3.16 The new home page

FAQ How do you choose class and id names?

You can choose almost any name you wish for a CSS class or id. However, CSS class names are more flexible and easier to maintain over time if they are descriptive of the structure rather than of specific formatting. For example, a class name of largeBold would no longer be meaningful if the design were changed to display the area differently; however, a structural class name such as item, content, or subheading is meaningful regardless of how the area is configured. Here are more hints for class names:

- Use short, but descriptive, names.
- Always begin with a letter.

- Avoid spaces in class names.
- Feel free to use numerals, the dash character, and the underscore character in addition to letters.

Be wary of "classitis"—that is, creating a brand new class each time you need to configure text a bit differently. Decide ahead of time how you will configure page areas, code your classes, and apply them. The result will be a more cohesive and better organized web page.

3.7 Span Element

Recall from Chapter 2 that the div element configures a section or division on a web page with empty space above and below. The div element is useful when you need to format a section that is physically separated from the rest of the web page, referred to as a block display. In contrast, the **span element** defines a section on a web page that is *not* physically separated from other areas; this formatting is referred to as inline display. Use the **** tag if you need to format an area that is contained within another, such as within a <p>, <blockquote>, <li>, or tag.

Hands-On Practice 3.6

You will experiment with the span element in this Hands-On Practice by configuring a new class to format the company name when it is displayed within the text on the page and using the span element to apply this class. Use the file from Hands-On Practice 3.5 (see the student files chapter3/3.5/index.html) as a starting point. Open the file in a text editor. Your web page will look similar to the one shown in Figure 3.17 after the changes are complete.

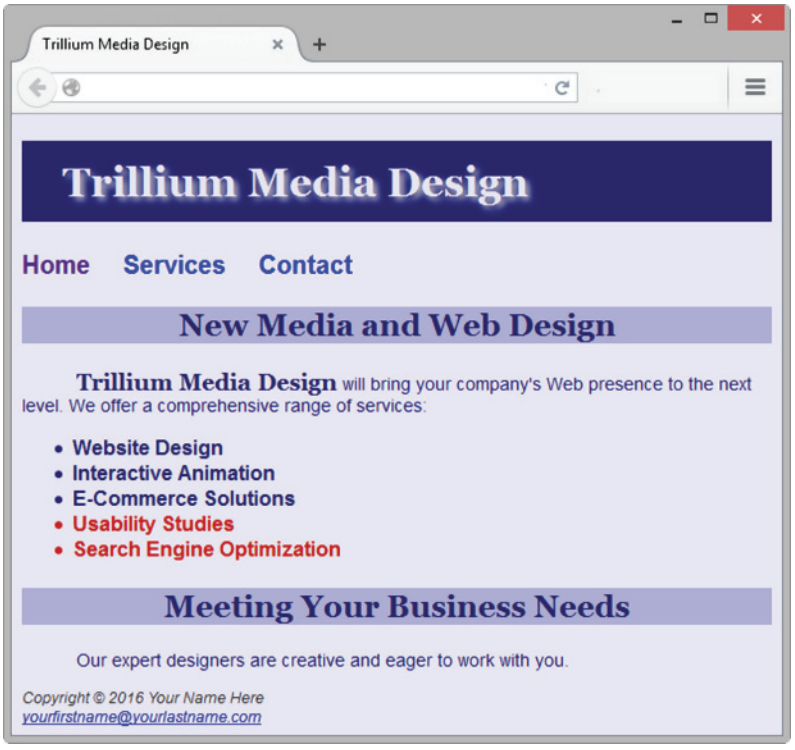

Figure 3.17 This web page uses the span element

Configure the Company Name

View Figure 3.17, and notice that the company name, Trillium Media Design, is displayed in bold and serif font within the first paragraph. You will code both CSS and HTML to configure this formatting. First, create a new CSS rule above the closing style tag that configures a class called `company` in bold, serif font, and 1.25em in size. The code follows:

```
.company { font-weight: bold;
           font-family: Georgia, "Times New Roman", serif;
           font-size: 1.25em; }
```

Next, modify the beginning of the first paragraph of HTML to use the span element to apply the class as follows:

```
<p><span class="company">Trillium Media Design</span> will bring
```

Save your file, and test it in a browser. Your page should look similar to the one shown in Figure 3.17. A sample solution (chapter3/3.6/index.html) is in the student files. View the source code for your page, and review the CSS and HTML code. Note that all the styles were located in a single place on the web page. Since embedded styles are coded in a specific location, they are easier to maintain over time than inline styles. Also notice that you needed to code the styles for the `h2` element selector only once (in the head section), and *both* of the `<h2>` elements applied the `h2` style. This approach is more efficient than coding the same inline style on each `<h2>` element. However, it is uncommon for a website to have only one page. Repeating the CSS in the head section of each web page file is inefficient and difficult to maintain. In the next section, you will use a more efficient approach: configuring an external style sheet.

3.8 Using External Style Sheets

VideoNote
External Style Sheets

The flexibility and power of CSS are best utilized when the CSS is external to the web page document. An external style sheet is a text file with a .css file extension that contains CSS style rules. The external style sheet file is associated with a web page by using the link element. This approach provides a way for multiple web pages to be associated with the same external style sheet file. The external style sheet file does not contain any HTML tags; it contains only CSS style rules.

The advantage of external CSS is that styles are configured in a single file. This means that when styles need to be modified, only one file needs to be changed, instead of multiple web pages. On large sites, this approach can save a web developer much time and increase productivity. Let's get some practice with this useful technique.

Link Element

The **link element** associates an external style sheet with a web page. It is placed in the head section of the page. The link element is a stand-alone, void tag. Three attributes are used with the link element: rel, href, and type.

- The value of the **rel attribute** is `"stylesheet"`.
- The value of the **href attribute** is the name of the style sheet file.
- The value of the **type attribute** is `"text/css"`, which is the MIME type for CSS. The type attribute is optional in HTML5 and required in XHTML.

Code the following in the head section of a web page to associate the document with the external style sheet named color.css:

```
<link rel="stylesheet" href="color.css">
```

Hands-On Practice 3.7

Let's practice using external styles. First, you will create an external style sheet. Next, you will configure a web page to be associated with the external style sheet.

Create an External Style Sheet

Launch a text editor, and type in the following style rules to set the background color of a page to blue and the text color to white. Save the file as color.css.

```
body { background-color: #0000FF;
       color: #FFFFFF; }
```

Figure 3.18 shows the external color.css style sheet displayed in Notepad. Notice that there is no HTML in this file. HTML tags are not coded within an external style sheet. Only CSS rules (selectors, properties, and values) are coded in an external style sheet.

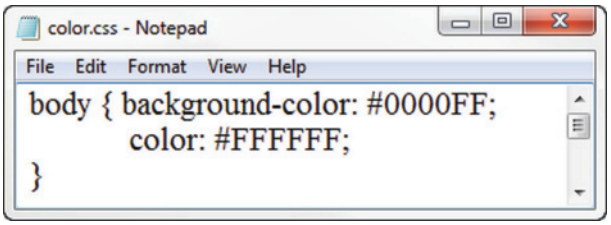

Figure 3.18 The external style sheet color.css. Screenshot from Microsoft® Notepad®. Used by permission of Microsoft Corporation

Configure the Web Page

To create the web page shown in Figure 3.19, launch a text editor and open the template file located at chapter2/template.html in the student files. Modify the title element, add a link tag to the head section, and add a paragraph to the body section as indicated by the following highlighted code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>External Styles</title>
<meta charset="utf-8">
<link rel="stylesheet" href="color.css">
</head>
<body>
 <p>This web page uses an external style sheet.</p>
</body>
</html>
```

Save your file as external.html. Launch a browser, and test your page. It should look similar to the page shown in Figure 3.19. You can compare your work with the solution in the student files (chapter3/3.7/external.html). The color.css style sheet can be associated with any number of web pages. If you ever need to change the style of formatting, you need to change only a single file (color.css) instead of multiple files (all of the web pages). As mentioned earlier, this technique can boost productivity on a large site.

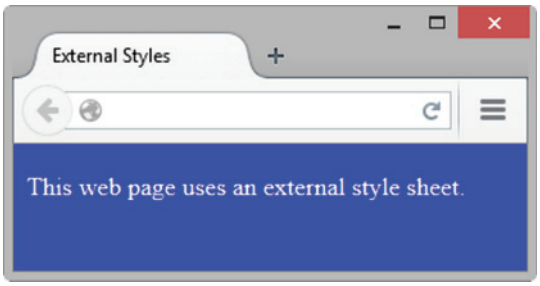

Figure 3.19 This page is associated with an external style sheet

The advantage of having only a single file to update is significant for both small and large websites. In the next Hands-On Practice, you will modify the Trillium home page to use an external style sheet.

Hands-On Practice 3.8

In this Hands-On Practice, you will continue to gain experience using external style sheets as you create the external style sheet file named trillium.css, modify the Trillium home page to use external styles instead of embedded styles, and associate a second web page with the trillium.css style sheet.

Use the file from Hands-On Practice 3.6 (see the student files chapter3/3.6/index.html) as a starting point. Open the file in a browser. The display should be the same as the web page shown in Figure 3.17 from Hands-On Practice 3.6.

Now that you've seen what you're working with, let's begin. Launch a text editor, and save the file as index.html in a folder called trilliumext. You are ready to convert the embedded CSS to external CSS. Select the CSS rules (all the lines of code between, but not including, the opening and closing `<style>` tags). Use Edit > Copy, or press the Ctrl + C keys (Cmd + C keys on a Mac), to copy the CSS code to the clipboard. Now you will place the CSS in a new file. Launch a text editor and create a new file. Use Edit > Paste, or press the Ctrl + V keys (Cmd + V keys on a Mac), to paste the CSS style rules. Save the file as trillium.css. See Figure 3.20 for a screenshot of the new trillium.css file in Notepad. Notice that there are no HTML elements in trillium.css— not even the `<style>` element. The file contains CSS rules only.

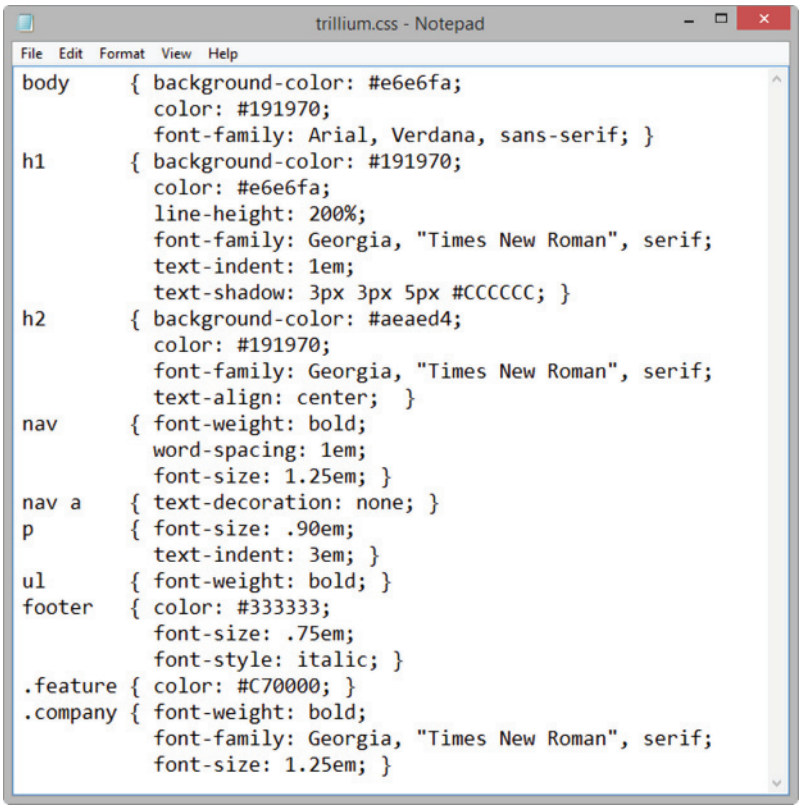

Figure 3.20 The external style sheet named trillium.css. Screenshot from Microsoft® Notepad®. Used by permission of Microsoft Corporation.

Next, edit the index.html file in a text editor. Delete the CSS code you just copied. Delete the closing `</style>` tag. Replace the opening `<style>` tag with a link element to associate the style sheet named trillium.css. The **`<link>`** tag code is as follows:

```
<link href="trillium.css" rel="stylesheet">
```

Save the file, and test it in a browser. Your web page should look just like the one shown in Figure 3.17. Although it looks the same, the difference is in the code: The page now uses external, instead of embedded, CSS.

Now, for the fun part—you will associate a second page with the style sheet. The student files contain a services.html page for Trillium at chapter3/services.html. When you display this page in a browser, it should look similar to the one shown in Figure 3.21. Notice that although the structure of the page is similar to the home page, the styling of the text and colors is absent.

Copy the services.html file to your trilliumext folder. Launch a text editor to edit your services.html file. Code a `<link>` element to associate the services.html web page with the trillium.css external style sheet. Place the following code in the head section above the closing `</head>` tag:

```
<link href="trillium.css" rel="stylesheet">
```

Save your file, and test it in a browser. Your page should look similar to Figure 3.22—the CSS rules have been applied!

If you click the Home and Services hyperlinks, you can move back and forth between the index.html and services.html pages in the browser. The student files contain a sample solution in the chapter3/3.8 folder.

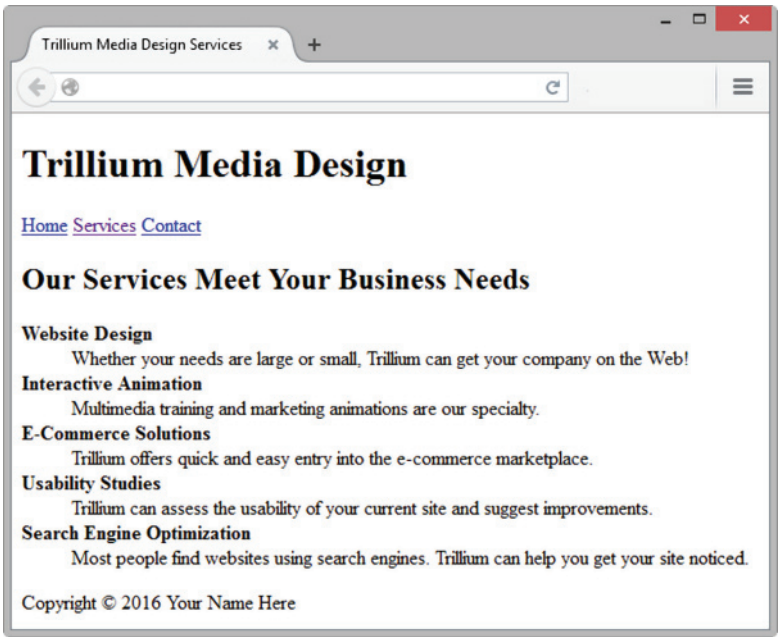

Figure 3.21 The services.html page is not associated with a style sheet

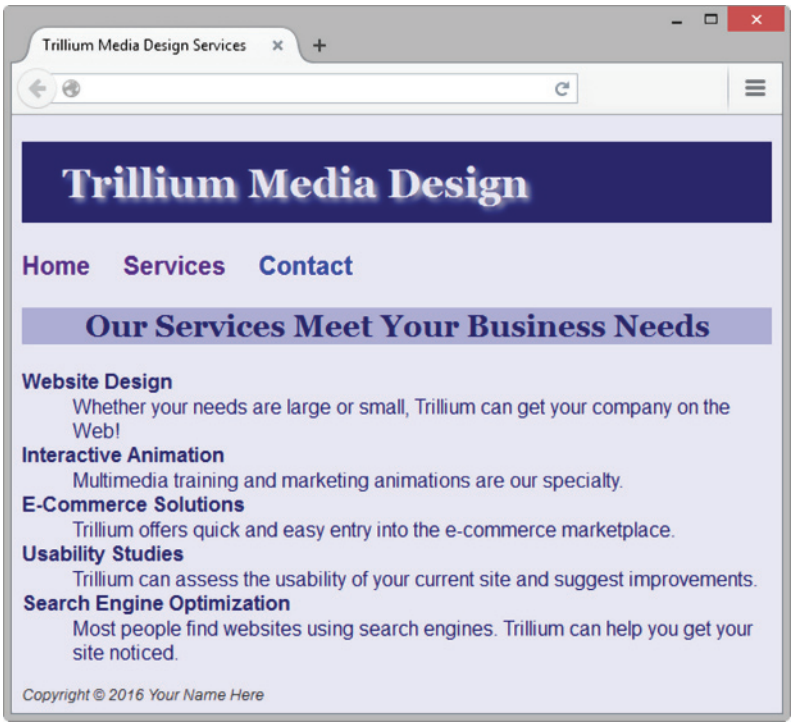

Figure 3.22 The services.html page has been associated with trillium.css

Notice that when using an external style sheet, if the use of color or fonts on the page ever needs to be changed, modifications are made only to the external style sheet. Think about how this approach can improve productivity on a site with many pages. Instead of modifying hundreds of pages to make a color or font change, only a single file—the CSS external style sheet— needs to be updated. Becoming comfortable with CSS will be important as you develop your skills and increase your technical expertise.

Checkpoint 3.2

1. Describe a reason to use embedded styles. Explain where embedded styles are placed on a web page.
2. Describe a reason to use external styles. Explain where external styles are placed and how web pages indicate that they are using external styles.
3. Write the code to configure a web page to associate with an external style sheet called mystyles.css.

FAQ When designing a new web page or website, how do I begin to work with CSS?

The following guidelines can be helpful when configuring a page using CSS:

- Review the design of the page. Check if common fonts are used. Define global properties (the default for the entire page) for characteristics such as fonts and colors attached to the body element selector.
- Identify typical elements used for organization in the page (such as `<h1>`, `<h2>`, and so on), and declare style rules for these elements if different from the default.
- Identify various page areas such as the header, navigation, footer, and so on. List any special configurations needed for these areas. You may decide to configure classes or ids in your CSS to configure these areas.
- Create one prototype page that contains most of the elements you plan to use and test. Revise your CSS as needed.
- Plan and test. These are important activities when designing a website.

3.9 Center HTML Elements with CSS

You learned how to center text on a web page earlier in this chapter—but what about centering the entire web page itself? A popular page layout design that is easy to accomplish with just a few lines of CSS is to center the entire contents of a web page within a browser viewport. The key is to configure a div element that contains, or "wraps," the entire page content. The HTML follows:

```
<body>
<div id="wrapper">
...  page content goes here  ...
</div>
</body>
```

Next, configure CSS style rules for this container. As will be discussed further in Chapter 6, the **margin** is the empty space surrounding an element. In the case of the body element, the margin is the empty space between the page content and the edges of the browser window. As you might expect, the **`margin-left`** and **`margin-right`** **properties** configure the space in the left and right margins, respectively. The margins can be set to 0, pixel units, em units, percentages, or auto. When margin-left and margin-right are both set to auto, the browser calculates the amount of space available and divides it evenly

between the left and right margins. The **`width` property** configures the width of a block display element. The following CSS code sample sets the width of an id named `wrapper` to 960 pixels and centers it:

```
#wrapper { width: 960px;
           margin-left: auto;
           margin-right: auto; }
```

You'll practice this technique in the next Hands-On Practice.

Hands-On Practice 3.9

In this Hands-On Practice, you will code CSS properties to configure a centered page layout, using the files from Hands-On Practice 3.8 as a starting point. Create a new folder called trilliumcenter. Locate the chapter3/3.8 folder in the student files. Copy the index.html, services.html, and trillium.css files to your trilliumcenter folder. Open the trillium.css file in a text editor. Create an id named `wrapper`. Add the margin-left, margin-right, and width style properties to the style rules as follows:

```
#wrapper { margin-left: auto;
           margin-right: auto;
           width: 80%; }
```

Save the file.

Open the index.html file in a text editor. Add the HTML code to configure a div element assigned to the id wrapper that "wraps," or contains, the code within the body section. Save the file. When you test your index.html file in a browser, it should look similar to the page shown in Figure 3.23. The student files contain a sample solution in the chapter3/3.9 folder.

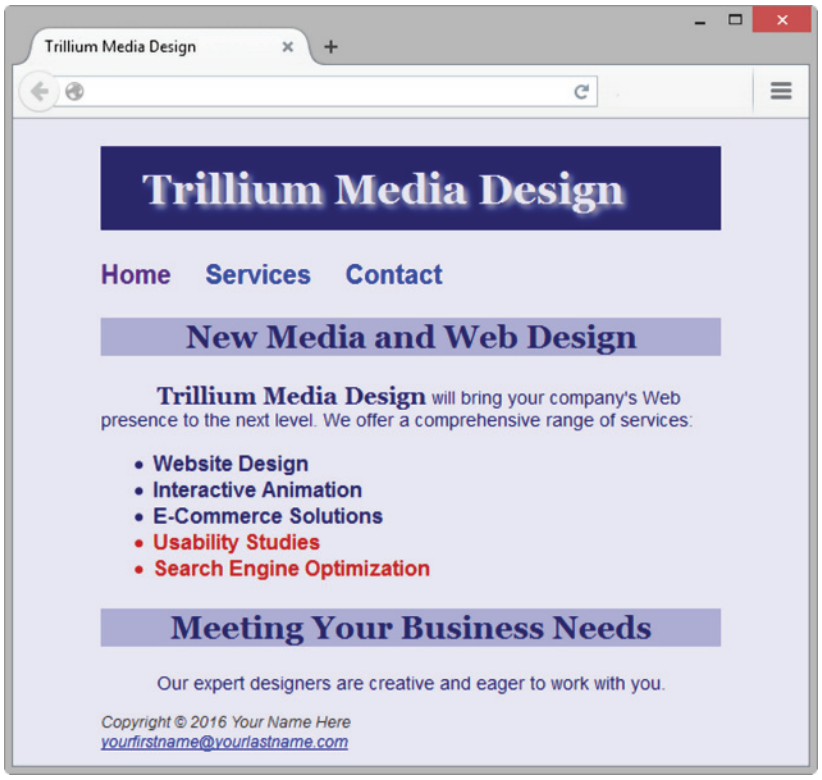

Figure 3.23 The page content is centered within the browser viewport

FAQ **Is there an easy way to add a comment for documentation purposes in CSS?**

Yes. An easy way to add a comment to CSS is to type `"/*"` before your comment and `"*/"` after your comment, as shown in the following example:

```
/* Configure Footer */
footer { font-size: .80em; font-style: italic; text-align: center; }
```

3.10 The "Cascade"

Figure 3.24 shows the "cascade" (**rules of precedence**) that applies the styles in order from outermost (external styles) to innermost (HTML attributes coded on the page). This set of rules allows the sitewide styles to be configured, but overridden when needed by more granular page-specific styles (such as embedded or inline styles).

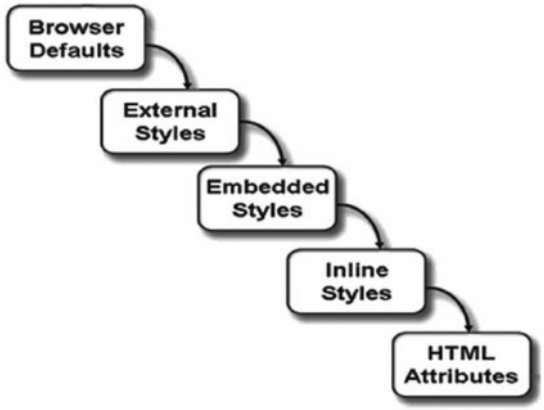

Figure 3.24 The "cascade" of Cascading Style Sheets

External styles can apply to multiple pages. If a web page contains both a link to an external style sheet followed by embedded styles, the external styles will be applied first, and then the embedded styles will be applied. This approach allows a web developer to override global external styles on selected pages.

If a web page contains both embedded styles and inline styles, the embedded styles are applied first, and then the inline styles are applied. This approach allows a web developer to override pagewide styles for particular HTML tags or classes.

Any HTML tag or attribute will override styles. For example, a `<strong>` tag will override corresponding font-related styles configured for an element. If no attribute or style is applied to an element, the browser's default is applied. However, the appearance of the browser's default may vary by browser, and you might be disappointed with the result. Use CSS to specify the properties of your text and web page elements. Avoid depending on the browser's default.

In addition to the general cascade of CSS types described previously, the style rules themselves follow rules of precedence. Style rules applied to more local elements (such as a paragraph) take precedence over those applied to more global elements (such as a `<div>` that contains the paragraph).

Let's look at an example of the cascade. Consider the following CSS code:

```
.special { font-family: Arial, sans-serif; }
p { font-family: "Times New Roman", serif; }
```

The CSS has two style rules: a rule creating a class named `special` that configures text using the Arial (or generic sans-serif) font family, and a rule configuring all paragraphs to use the Times New Roman (or generic serif) font family. The HTML on the page contains a `<div>` with multiple elements, such as headings and paragraphs, as shown in the following code:

```
<div class="special">
<h2>Heading</h2>
<p>This is a paragraph. Notice how the paragraph is contained in the
div.</p>
</div>
```

Here is how the browser would render the code:

1. The text contained in the heading is displayed with Arial font because it is part of the `<div>` assigned to the `special` class. It inherits the properties from its parent (`<div>`) class. This is an example of **inheritance**, in which certain CSS properties are passed down to elements nested within a container element, such as a `<div>` or `<body>` element. Text-related properties (font-family, color, etc.) are generally inherited, but box-related properties (margin, padding, width, etc.) are not. See http://www.w3.org/TR/CSS21/propidx.html for a detailed list of CSS properties and their inheritance status.
2. The text contained in the paragraph is displayed with Times New Roman font because the browser applies the styles associated with the most local element (the paragraph). Even though the paragraph is contained in (and is considered a child of) the `special` class, the local paragraph style rules takes precedence and are applied by the browser.

Don't worry if CSS and rules of precedence seem a bit overwhelming at this point. CSS definitely becomes easier with practice. You will get a chance to practice with the "cascade" as you complete the next Hands-On Practice.

Hands-On Practice 3.10

You will experiment with the "cascade" in this Hands-On Practice as you work with a web page that uses external, embedded, and inline styles.

1. Create a new folder named mycascade.
2. Launch a text editor. Open a new file. Save the file as site.css in the mycascade folder. You will create an external style sheet that sets the `background-color` of the web page to a shade of yellow (`#FFFFCC`) and the text color to black (`#000000`). The code follows:

   ```
   body { background-color: #FFFFCC;
          color: #000000; }
   ```

   Save and close the site.css file.

3. Open a new file in the text editor, and save it as mypage1.html in the mycascade folder. The web page will be associated with the external style sheet site.css, use embedded styles to set the global text color to blue, and use inline styles to configure the text color of the second paragraph. The file mypage1.html will contain two paragraphs of text. The code follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>The Cascade in Action</title>
<meta charset="utf-8">
<link rel="stylesheet" href="site.css">
<style>
  body { color: #0000FF; }
</style>
</head>
<body>
<p>This paragraph applies the external and embedded styles &mdash;
note how the blue text color that is configured in the embedded
styles takes precedence over the black text color configured in
the external stylesheet.</p>
<p style="color: #FF0000">Inline styles configure this paragraph
to have red text and take precedence over the embedded and
external styles.</p>
</body>
</html>
```

Save mypage1.html, and display it in a browser. Your page should look similar to the sample shown in Figure 3.25. The student files contain a sample solution at chapter3/3.10/mypage1.html.

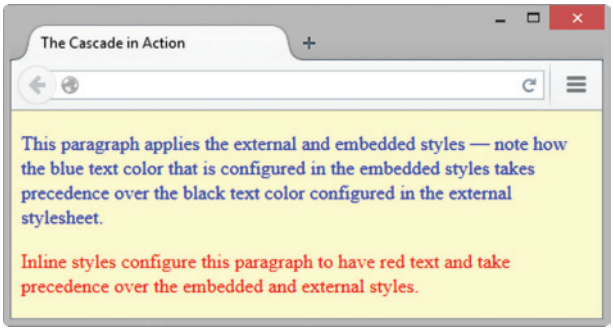

Figure 3.25 Mixing external, embedded, and inline styles

Take a moment to examine the mypage1.html web page and compare it with its source code. The web page picked up the yellow background from the external style sheet. The embedded style configured the text to be the color blue, which overrides the black text color in the external style sheet. The first paragraph in the web page does not contain any inline styles, so it inherits the style rules in the external and embedded style sheets. The second paragraph contains an inline style of red text color; this setting overrides the corresponding external and embedded styles.

3.11 CSS Validation

VideoNote
CSS Validation

The W3C has a free Markup Validation Service (http://jigsaw.w3.org/css-validator) that will validate your CSS code and check it for syntax errors. **CSS validation** provides students with quick self-assessment—you can prove that your code uses correct syntax. In the working world, CSS validation serves as a quality assurance tool. Invalid code may cause browsers to render the pages more slowly than otherwise.

Hands-On Practice 3.11

In this Hands-On Practice, you will use the W3C CSS Validation Service to validate an external CSS style sheet. This example uses the color.css file completed in Hands-On Practice 3.7 (student files chapter3/3.7/color.css). Locate color.css, and open it in a text editor. Now add an error to the color.css file by finding the body element selector style rule and deleting the first "r" in the `background-color` property. Add another error by removing the # from the `color` property value. Save the file.

Next, attempt to validate the color.css file. Visit the W3C CSS Validation Service page at http://jigsaw.w3.org/css-validator and select the "by File Upload" tab. Click the Browse button and select the color.css file from your computer. Click the Check button. Your display should be similar to that shown in Figure 3.26. Notice that two errors were found. The selector is listed, followed by the reason an error was noted.

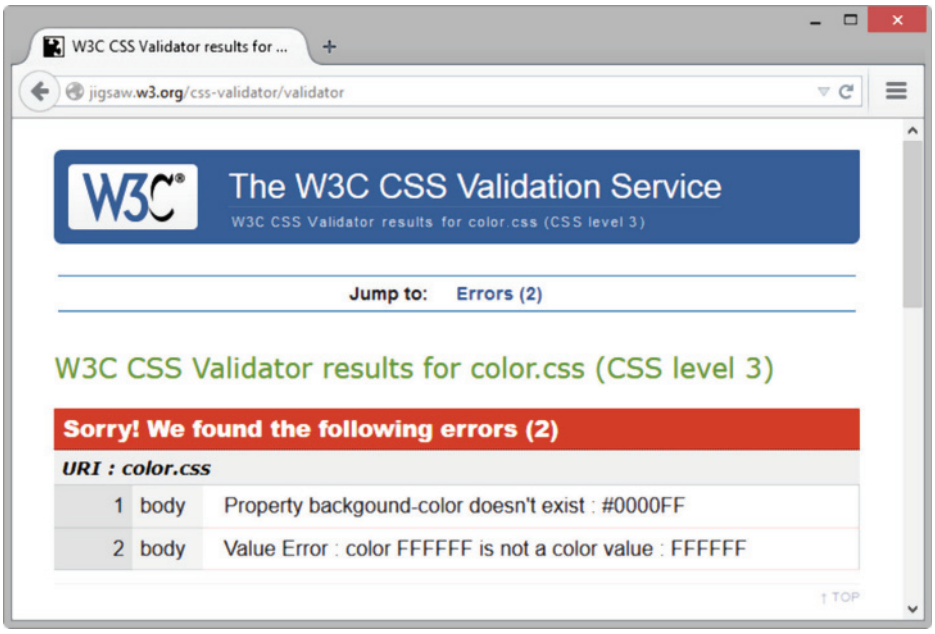

Figure 3.26 The validation results indicate errors. Screenshots of W3C. Courtesy of W3C (World Wide Web Consortium)

Notice that the first message in Figure 3.26 indicates that the "`backgound-color`" property does not exist. This is a clue to check the syntax of the property name. Edit color.css and correct the first error. Test and revalidate your page. Your browser should now look similar to the one shown in Figure 3.27 and should be reporting only one error.

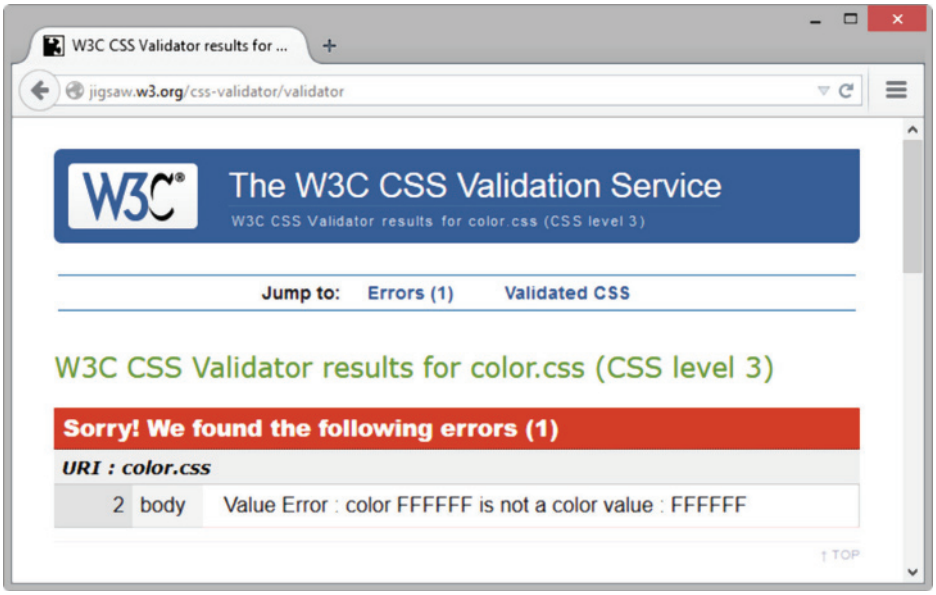

Figure 3.27 The valid CSS code is displayed below the errors (and warnings, if any). Screenshots of W3C. Courtesy of W3C (World Wide Web Consortium)

The error reminds you that `FFFFFF` is not a color value; you are expected to already know that you need to add a `"#"` character to code a valid color value, as in `#FFFFFF`. Notice how any valid CSS rules are displayed below the error messages. Correct the color value, save the file, and test again.

Your results should look similar to those shown in Figure 3.28. There are no errors listed. The Valid CSS Information contains all the CSS style rules in color.css. This means your file passed the CSS validation test. Congratulations, your color.css file is valid CSS syntax! It is good practice to validate your CSS style rules. The CSS validator can help you to quickly identify code that needs to be corrected and indicate which style rules a browser is likely to consider valid. Validating CSS code is one of the many productivity techniques that web developers commonly use.

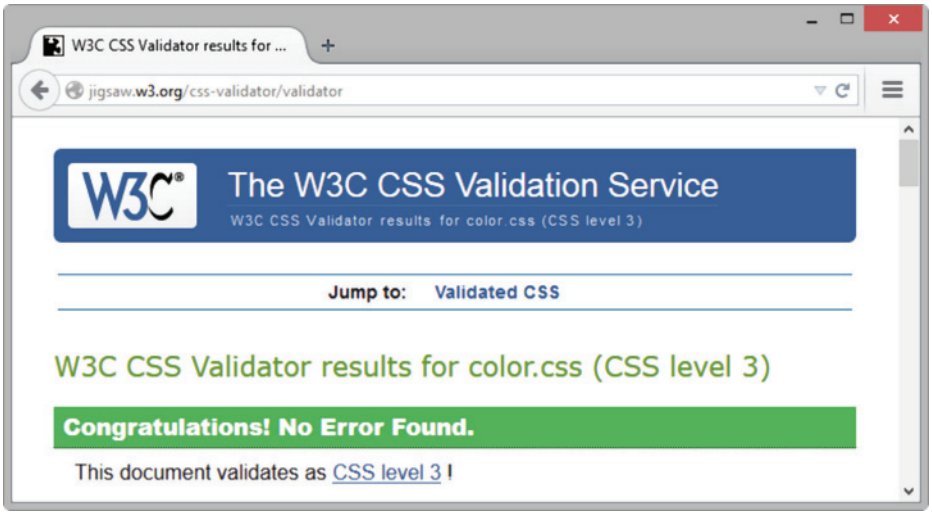

Figure 3.28 The CSS is valid! Screenshots of W3C. Courtesy of W3C (World Wide Web Consortium)

Chapter Summary

This chapter has introduced Cascading Style Sheet rules associated with color and text on web pages. There is much more that you can do with CSS, including positioning, hiding and showing page areas, formatting margins, and formatting borders. As you continue your study of web development with this textbook, you will learn about these additional uses. Visit this textbook's website at http://www.webdevfoundations.net for examples, information on the resources described in this chapter, and to view updated information.

Key Terms

`<link>`
`<span>`
`<style>`
`background-color` property
Cascading Style Sheets (CSS)
class attribute
class selector
`color` property
CSS validation
descendant selector
declaration
em unit
embedded styles
external styles
`font-family` property
`font-size` property
`font-style` property
`font-weight` property
hexadecimal color values
href attribute
id attribute
id selector
imported styles
inheritance
inline styles
internal styles
`letter-spacing` property
`line-height` property
link element
`margin-left` property
`margin-right` property
pixels
point
property
`rel` attribute
RGB color
rule
rules of precedence
selector
span element
`style` attribute
style element
`text-align` property
`text-decoration` property
`text-indent` property
`text-shadow` property
`text-transform` property
`type` attribute
Web-Safe Color Palette
Web-Safe Colors
`white-space` property
`width` property
`word-spacing` property

Review Questions

Multiple Choice

1. Which type of CSS is coded in the body of the web page as an attribute of an HTML tag?
   a. embedded
   b. inline
   c. external
   d. imported

2. Which of the following is the CSS property used to set the background color of a web page?
   a. bgcolor
   b. background-color
   c. color
   d. none of the above

3. Which of the following describe two components of CSS rules?
   a. selectors and declarations
   b. properties and declarations
   c. selectors and attributes
   d. none of the above

4. Which of the following associates a web page with an external style sheet?
   a. `<style rel="external" href="style.css">`
   b. `<style src="style.css">`
   c. `<link rel="stylesheet" href="style.css">`
   d. `<link rel="stylesheet" src="style.css">`

5. Which of the following is the declaration property used to set the font typeface for an area of a web page?
   a. font-face
   b. face
   c. font-family
   d. size
6. Which of the following do you configure to apply a style to only one area on a web page?
   a. group
   b. class
   c. id
   d. none of the above
7. Which of the following can be a CSS selector?
   a. an HTML element name
   b. a class name
   c. an id name
   d. all of the above
8. Where do you place the code to associate a web page with an external style sheet?
   a. in the external style sheet
   b. in the DOCTYPE of the web page document
   c. in the body section of the web page document
   d. in the head section of the web page document
9. Which of the following configures a background color of #00CED1 for a web page using CSS?
   a. `body { background-color: #00CED1; }`
   b. `document { background: #00CED1; }`
   c. `body { bgcolor: #00CED1;}`
   d. `document { bgcolor: #00CED1; }`
10. Which of the following uses CSS to configure a class called news with red text, large font, and Arial or a sans-serif font?

```
a. news { color: red;
          font-size: large;
          font-family: Arial,
          sans-serif; }
b. .news { color: red;
           font-size: large;
           font-family: Arial,
           sans-serif; }
c. .news { text: red;
           font-size: large;
           font-family: Arial,
           sans-serif; }
d. #news { text: red;
           font-size: large;
           font-family: Arial,
           sans-serif;}
```

11. Which of the following is true if a web page contains both a link to an external style sheet and embedded styles?
    a. Embedded styles will be applied first, and then the external styles will be applied.
    b. The inline styles will be used.
    c. External styles will be applied first, and then the embedded styles will be applied.
    d. The web page will not display.

Fill in the Blank

12. The _____ element is useful for creating areas on a web page that are embedded within paragraphs or other block display elements.
13. The _____ CSS property can be used to center text within a block display element.
14. The _____ CSS property can be used to indent the first line of text.
15. CSS was first proposed as a standard by the W3C in _____.

Apply Your Knowledge

1. **Predict the Result.** Draw and write a brief description of the web page that will be created with the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Trillium Media Design</title>
<meta charset="utf-8">
```

```
<style>
  body { background-color: #000066;
         color: #CCCCCC;
         font-family: Arial,sans-serif; }
  header { background-color: #FFFFFF;
           color: #000066; }
  footer { font-size: 80%;
           font-style: italic; }
</style>
</head>
<body>
<header><h1>Trillium Media Design</h1></header>
<nav>Home <a href="about.html">About</a> <a href="services.html">
Services</a>
</nav>
<p>Our professional staff takes pride in its working relationship
with our clients by offering personalized services that listen
to their needs, develop their target areas, and incorporate these
items into a website that works.</p>
<br><br>
<footer>
Copyright &copy; 2016 Trillium Media Design
</footer>
</body>
</html>
```

2. **Fill in the Missing Code.** Consider the following code, in which some CSS properties and values, indicated by "_", and some HTML tags, indicated by <_>, are missing:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Trillium Media Design</title>
<meta charset="utf-8">
<style>
  body { background-color: #0066CC;
         color:  "_"; }
  header {  "_":  "_"  }
<_>
<_>
<body>
<header><h1>Trillium Media Design</h1></header>
   <p>Our professional staff takes pride in its working
relationship with our clients by offering personalized services
that listen to their needs, develop their target areas, and
incorporate these items into a website that works.</p>
</body>
</html>
```

The web page corresponding to the code should be configured so that the background and text colors have good contrast. The header area should use Arial. Fill in the missing code.

3. **Find the Error.** Why won't the page corresponding to the following code display properly in a browser?

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Trillium Media Design</title>
<meta charset="utf-8">
<style>
  body { background-color: #000066;
         color: #CCCCCC;
         font-family: Arial,sans-serif;
         font-size: 1.2em; }
<style>
</head>
<body>
<header><h1>Trillium Media Design</h1></header>
   <main><p>Our professional staff takes pride in its working
relationship with our clients by offering personalized services
that listen to their needs, develop their target areas, and
incorporate these items into a website that works.</p></main>
</body>
</html>
```

Hands-On Exercises

1. Write the HTML for a paragraph that uses inline styles to configure the background color of green and the text color of white.
2. Write the HTML and CSS code for an embedded style sheet that configures a background color of `#eaeaea` and a text color of `#000033`.
3. Write the CSS code for an external style sheet that configures the text to be brown, 1.2em in size, and in Arial, Verdana, or a sans-serif font.
4. Write the HTML and CSS code for an embedded style sheet that configures a class called new that is bold and italic.
5. Write the HTML and CSS code for an embedded style sheet that configures links without underlines; a background color of white; text color of black; is in Arial, Helvetica, or a sans-serif font; and has a class called new that is bold and italic.
6. Write the CSS code for an external style sheet that configures a page background color of `#FFF8DC;` has a text color of `#000099;` is in Arial, Helvetica, or a sans-serif font; and has an id called new that is bold and italic.
7. **Practice with External Style Sheets.** In this exercise, you will create two external style sheet files and a web page. You will experiment with linking the web page to the external style sheets and note how the display of the page is changed.
   a. Create an external style sheet (call it format1.css) to format as follows: document background color of white, document text color of `#000099`, and document font family of Arial, Helvetica, or sans-serif. Hyperlinks should have a background color of gray `(#CCCCCC)`. Configure the h1 selector to use the Times New Roman font with red text color.

b. Create an external style sheet (call it format2.css) to format as follows: document background color of yellow and document text color of green. Hyperlinks should have a background color of white. Configure the h1 selector to use the Times New Roman font with white background color and green text color.

c. Create a web page about your favorite movie that displays the movie name in an <h1> tag, a description of the movie in a paragraph, and an unordered (bulleted) list of the main actors and actresses in the movie. The page should also have a hyperlink to a website about the movie. Place an e-mail link to yourself on the web page. This page should be associated with the format1.css file. Save the page as moviecss1.html. Be sure to test your page in more than one browser.

d. Modify the moviecss1.html page to link to the format2.css external style sheet instead of the format1.css file. Save the page as moviecss2.html and test it in a browser. Notice how different the page looks!

8. **Practice with the Cascade.** In this exercise, you will create two web pages that link to the same external style sheet. After modifying the configuration in the external style sheet, you will test your pages again and find that they automatically pick up the new style configuration. Finally, you will add an inline style to one of the pages and find that it takes effect and overrides the external style.

a. Create a web page that includes an unordered list describing at least three advantages of using CSS. The text "CSS Advantages" should be contained within <h1> tags. This page should include a hyperlink to the W3C website. Write the HTML code so that one of the advantages is configured to be a class called news. Place an e-mail link to yourself on the web page. The web page should be associated with the external style sheet called ex8.css. Save the page as advantage.html.

b. Create an external style sheet (call it ex8.css) to format as follows: document background color of white; document text color of #000099; and document font family of Arial, Helvetica, or sans-serif. Hyperlinks should have a background color of gray (#CCCCCC). <h1> elements should use the Times New Roman font with black text color. The news class should use red italic text.

c. Launch a browser, and test your work. Display the advantage.html page. It should use the formatting configured in ex8.css. Modify the web page or the CSS file until your page displays as requested.

d. Change the configuration of the external style sheet (ex8.css) to use a document background color of black, document text color of white, and <h1> text color of gray (#CCCCCC). Save the file. Launch a browser, and test the advantage.html page. Notice how it picks up the new styles from the external style sheet.

e. Modify the advantage.html file to use an inline style. The inline style should be applied to the <h1> tag and configure it to have red text. Save the advantage.html page, and test in a browser. Notice how the <h1> text color specified in the style sheet is overridden by the inline style.

9. **Practice Validating CSS.** Choose a CSS external style sheet file to validate; perhaps you have created one for your own website. Otherwise, use an external style sheet file that you worked with in this chapter. Use the W3C CSS validator at http://jigsaw.w3.org/css-validator. If your CSS does not immediately pass the validation test, modify it and test again. Repeat this process until the W3C validates your CSS code. Write a one- or two-paragraph summary about the validation process that answers the following questions: Was the CSS validator easy to use? Did anything surprise you? Did you encounter a number of errors or just a few? How easy was it to determine how to correct the CSS file? Would you recommend the validator to other students? Why or why not?

Web Research

1. This chapter has introduced you to using CSS to configure web pages. Use a search engine to find CSS resources. The following resources can help you get started:

   - http://www.w3.org/Style/CSS
   - https://developer.mozilla.org/en-US/docs/Web/CSS
   - http://reference.sitepoint.com/css

   Create a web page that provides a list of at least five CSS resources on the Web. For each CSS resource, provide the URL, website name, and a brief description. Configure text and background colors with good contrast. Place your name in the e-mail address at the bottom of the web page.

2. There is still much for you to learn about CSS. A great place to learn about web technology is the Web itself. Use a search engine to search for CSS tutorials. The following resources can help you get started:

   - http://www.echoecho.com/css.htm
   - http://www.w3schools.com/css
   - http://www.noupe.com/css/css-typography-contrast-techniques-tutorials-and-best-practices.html

   Choose a tutorial that is easy to read. Select a section that discusses a CSS technique that was not covered in this chapter. Create a web page that uses this new technique. The web page should provide the URL of your tutorial, the name of the website, and a description of the new technique you discovered. Place your name in the e-mail address at the bottom of the web page.

Focus on Web Design

In this chapter, you have learned how to configure color and text with CSS. In this activity, you will design a color scheme, code an external CSS file for the color scheme, and code an example web page that applies the styles you configured. Use any of the following sites to help you get started with color and web design ideas:

Psychology of Color

- http://www.infoplease.com/spot/colors1.html
- http://www.empower-yourself-with-color-psychology.com/meaning-of-colors.html
- http://www.designzzz.com/infographic-psychology-color-web-designers

Color Scheme Generators

- http://meyerweb.com/eric/tools/color-blend
- http://www.colr.org
- http://colorsontheweb.com/colorwizard.asp
- https://color.adobe.com/create/color-wheel
- http://paletton.com

Complete the following tasks:

a. Design a color scheme. List three hexadecimal color values in addition to neutral colors such as white (`#FFFFFF`) or black (`#000000`) in your design.

b. Describe the process you went through as you selected the colors. Describe why you chose these colors. What type of website would they be appropriate for? List the URLs of any resources you used.

c. Create an external CSS file named color1.css that configures font properties, text color, and background color selections for the document, h1 element selector, p element selector, and footer class, using the colors you have chosen.

d. Create a web page named color1.html that shows examples of the CSS style rules.

WEBSITE CASE STUDY

Implementing CSS

Each of the case studies in this section continues throughout most of the text. This chapter implements CSS in the websites.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House Case Study. Figure 2.30 shows a site map for the JavaJam website. The Home page and Menu page were created in Chapter 2. You will develop a new version of the website that uses an external style sheet to configure text and color. Figure 2.31 depicts the wireframe page layout.

You have the following tasks:

1. Create a new folder for this JavaJam case study.
2. Create an external style sheet named javajam.css that configures the color and text for the JavaJam website.
3. Modify the Home page to utilize an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.29.

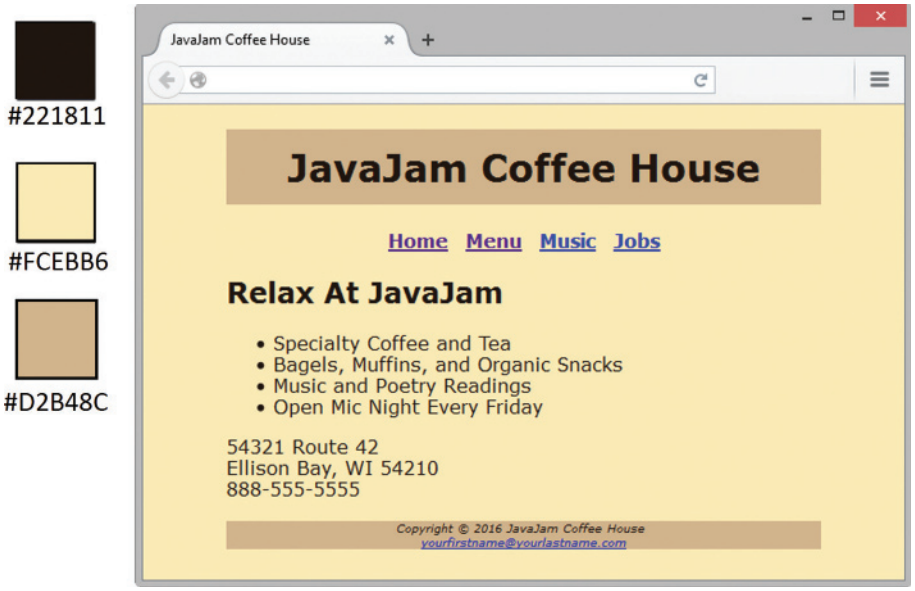

Figure 3.29 New JavaJam index.html

4. Modify the Menu page to be consistent with the new Home page.
5. Configure centered page layout.

Hands-On Practice Case

Task 1: Create a folder on your hard drive or portable storage device called javajamcss. Copy all the files from your Chapter 2 javajam folder into the javajamcss folder.

Task 2: The External Style Sheet. You will use a text editor to create an external style sheet named javajam.css. Code the CSS to configure the following:

1. Global styles for the document (use the body element selector) with background color `#FCEBB6;` text color `#221811;` and Verdana, Arial, or any sans-serif font.
2. Styles for the header element selector that configure background color `#D2B48C` centered text.
3. Styles for the h1 element selector that configure 200% line height.
4. Styles for the nav element selector that configure centered, bold text. *Hint*: Use the CSS `text-align` and `font-weight` properties.
5. Styles for the footer element selector that configure background color `#D2B48C`, small font size (`.60em`), italics, and centered text.

Save the file as javajam.css in the javajamcss folder. Check your syntax with the CSS validator (http://jigsaw.w3.org/css-validator). Correct and retest if necessary.

Task 3: The Home Page. Launch a text editor, and open the index.html file. You will modify this file to apply styles from the javajam.css external style sheet as follows:

1. Add a `<link>` element to associate the web page with the javajam.css external style sheet file.
2. Configure the navigation area. Remove the `<b>` elements which are no longer needed because you have configured bold text with CSS.
3. Configure the page footer area. Remove the `<small>` and `<i>` elements—they are no longer needed since CSS is now used to configure the text.

Save the index.html file, and test it in a browser. Your page should look similar to the one shown in Figure 3.29 except that your page content will be left-aligned instead of centered. Don't worry—you'll center your page layout in Task 5 of this case study.

Task 4: The Menu Page. Launch a text editor, and open the menu.html file. You will modify this file in a similar manner as you modified the home page: Add the `<link>` element and configure the navigation and page footer areas. Save and test your new menu.html page. It should look similar to the one shown in Figure 3.30, except for the alignment.

Task 5: Center Page Layout with CSS. Modify javajam.css, index.html, and menu.html to configure page content that is centered with 80% width (refer to Hands-On Practice 3.9 if necessary):

1. Launch a text editor, and open the javajam.css file. Add a style rule for an id named `wrapper` with `width` set to 80%, `margin-right` set to auto, and `margin-left` set to auto.
2. Launch a text editor, and open the index.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your index.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.29.

3. Launch a text editor, and open the menu.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your menu.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.30.

Experiment with modifying the javajam.css file. Change the page background color, the font family, and so on. Test your pages in a browser. Isn't it amazing how a change in a single file can affect multiple files when external style sheets are used?

Figure 3.30
New menu.html page

Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital Case Study. Figure 2.34 shows a site map for the Fish Creek website. The Home page and Services page were created in Chapter 2. You will develop a new version that uses an external style sheet to configure text and color. Figure 2.35 depicts the wireframe page layout.

You have the following tasks:

1. Create a new folder for this Fish Creek case study.
2. Create an external style sheet named fishcreek.css that configures the color and text for the Fish Creek website.
3. Modify the Home page to utilize an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.31.
4. Modify the Services page to be consistent with the new Home page.
5. Configure centered page layout.

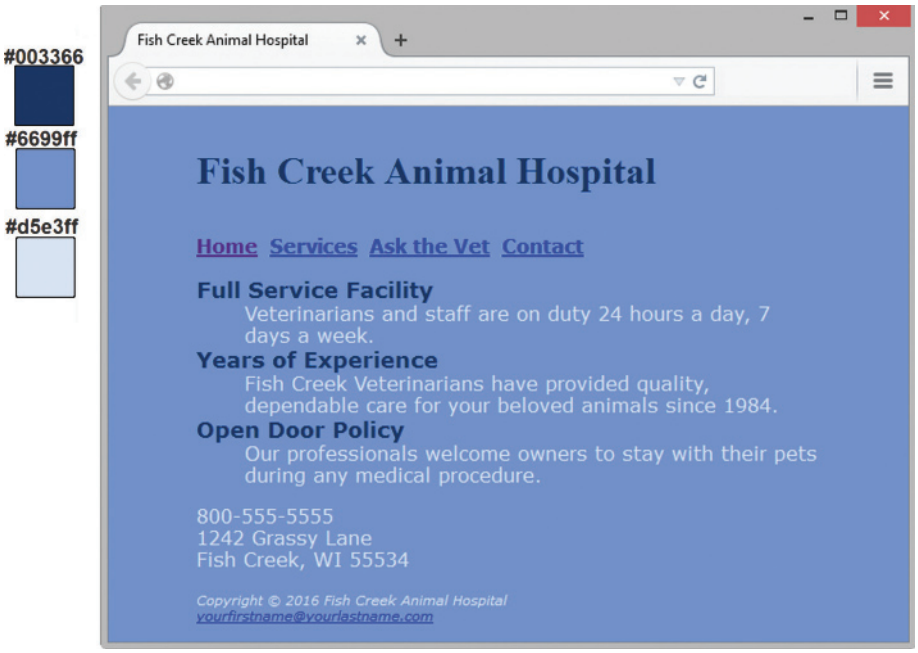

Figure 3.31 New Fish Creek index.html

Hands-On Practice Case Study

Task 1: Create a folder on your hard drive or portable storage device called fishcreekcss. Copy all the files from your Chapter 2 fishcreek folder into the fishcreekcss folder.

Task 2: The External Style Sheet. You will use a text editor to create an external style sheet named fishcreek.css. Code the CSS to configure the following:

1. Global styles for the document (use the body element selector) with background color `#6699FF`; text color `#D5E3FF`; and Verdana, Arial, or any sans-serif font.
2. Styles for the header element selector that configure background color `#6699FF`, text color `#003366`, and serif font.
3. Styles for the h1 element selector that configure 200% line height.
4. Styles for the nav element selector that display text in bold.
5. Styles for a class named `category` with bold font, background color `#6699FF`, text color `#003366`, and larger font size (`1.1em`).
6. Styles for the footer element selector with a small font size (`.70em`) and italic text.

Save the file as fishcreek.css in the fishcreekcss folder. Check your syntax with the CSS validator (http://jigsaw.w3.org/css-validator). Correct and retest if necessary.

Task 3: The Home Page. Launch a text editor, and open the index.html file. You will modify this file to apply styles from the fishcreek.css external style sheet as follows:

1. Add a `<link>` element to associate the web page with the fishcreek.css external style sheet file.
2. Configure the navigation area. Remove the `<b>` element from the navigation area, because the CSS will configure the bold font style.
3. Configure each `<dt>` element to apply the `category` class.
   Hint `<dt class="category">`. Remove the `<strong>` tags, because the CSS will configure the bold font style.

4. Configure the page footer area. Remove the `<small>` and `<i>` elements—they are no longer needed since CSS is now used to configure the text.

Save the index.html file, and test in a browser. Your page should look similar to the one shown in Figure 3.31 except that your page content will be left-aligned instead of indented from the margins. Don't worry—you'll configure your page layout in Task 5 of this case study.

Task 4: The Services Page. Launch a text editor, and open the services.html file. You will modify this file in a similar manner: Add the `<link>` element, configure the navigation area and page footer areas, configure the `category` classes (*Hint*: Use the `<span>` element to contain the name of each service offered), and remove the strong tags.) Save and test your new services.html page. It should look similar to the one shown in Figure 3.32 except for the alignment.

Task 5: Center Page Layout with CSS. Modify fishcreek.css, index.html, and services.html to configure page content that is centered with 80% width. Refer to Hands-On Practice 3.9 if necessary.

1. Launch a text editor, and open the fishcreek.css file. Add a style rule for an id named `wrapper` with `width` set to 80%, `margin-right` set to auto, and `margin-left` set to auto.
2. Launch a text editor, and open the index.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your index.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.31.
3. Launch a text editor, and open the services.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your services.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.32.

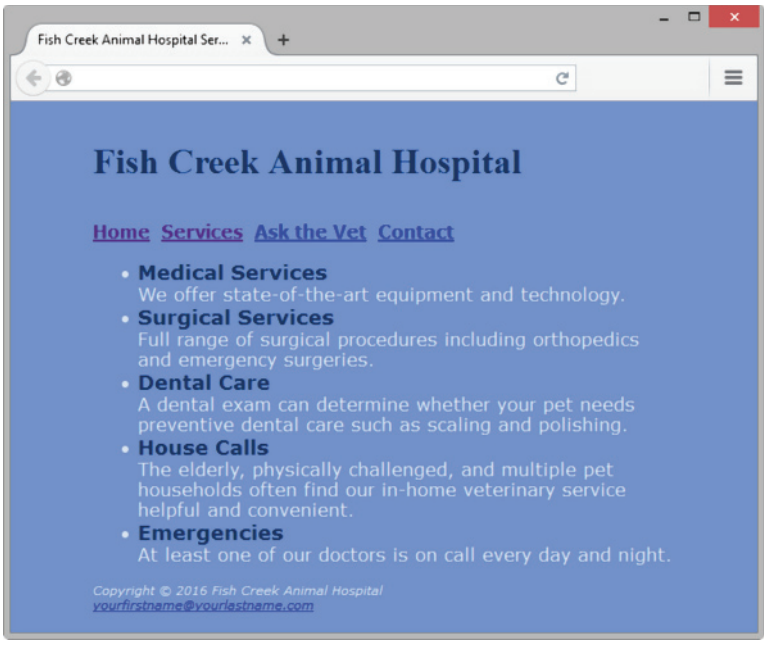

Figure 3.32 New services.html page

Experiment with modifying the fishcreek.css file. Change the page background color, the font family, and so on. Test your pages in a browser. Isn't it amazing how a change in a single file can affect multiple files when external style sheets are used?

Pacific Trails Resort

See Chapter 2 for an introduction to the Pacific Trails Resort Case Study. Figure 2.38 shows a site map for the Pacific Trails Resort website. The Home page and Yurts page were created in Chapter 2. You will develop a new version of this website that uses an external style sheet to configure text and color. Figure 2.39 depicts the wireframe page layout.

You have the following tasks:

1. Create a new folder for this Pacific Trails case study.
2. Create an external style sheet named pacific.css that configures the color and text for the Pacific Trails website.
3. Modify the Home page to utilize an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.33.
4. Modify the Yurts page to be consistent with the new Home page.
5. Configure centered page layout.

Hands-On Practice Case Study

Task 1: Create a folder on your hard drive or portable storage device called pacificcss. Copy all the files from your Chapter 2 pacific folder into the pacificcss folder.

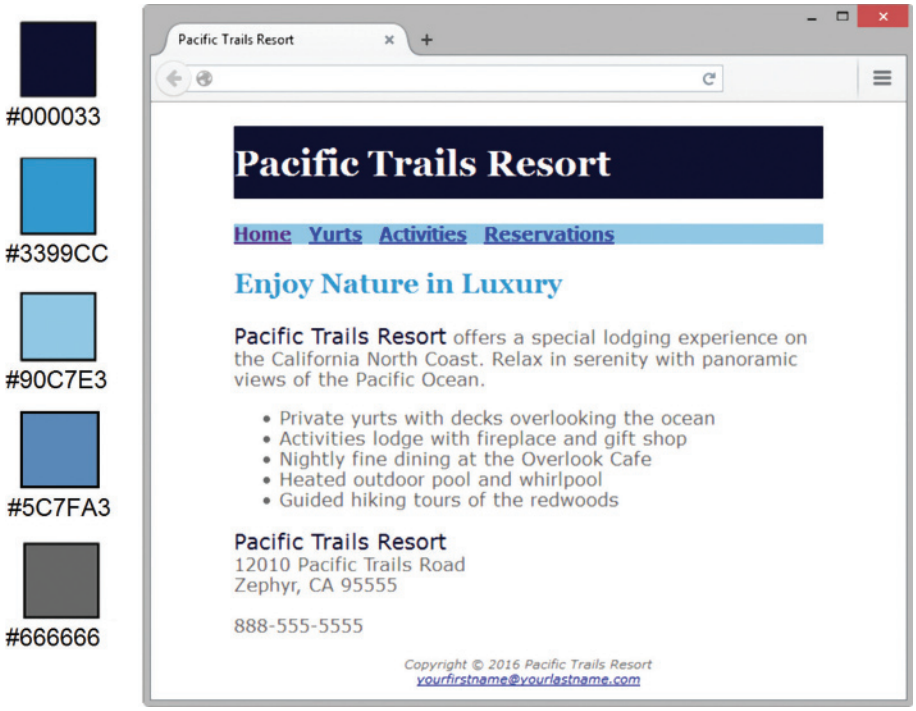

Figure 3.33 New Pacific Trails index.html

Task 2: The External Style Sheet. You will use a text editor to create an external style sheet named pacific.css. Code the CSS to configure the following:

1. Global styles for the document (use the body element selector) with background color `#FFFFFF`, text color `#666666`, and Verdana, Arial, or any sans-serif font.
2. Styles for the header element selector that configure background color `#000033`, text color `#FFFFFF`, and Georgia or any serif font.
3. Styles for the h1 element selector that configure 200% line height.
4. Styles for the nav element selector that display text in bold and has a sky-blue background color (`#90C7E3`).
5. Styles for the h2 element selector that configure medium-blue text color (`#3399CC`) and Georgia or any serif font.
6. Styles for the dt element selector that configure dark-blue text color (`#000033`) and bold font.
7. Styles for a class named `resort` that configure dark-blue text color (`#000033`) and `1.2em` font size.
8. Styles for the footer element selector with a small font size (`.70em`) and italic, centered text.

Save the file as pacific.css in the pacificcss folder. Check your syntax with the CSS validator (http://jigsaw.w3.org/css-validator). Correct and retest if necessary.

Task 3: The Home Page. Launch a text editor, and open the index.html file. You will modify this file to apply styles from the pacific.css external style sheet as follows:

1. Add a `<link>` element to associate the web page with the pacific.css external style sheet file.
2. Configure the navigation area. Remove the `<b>` element from the navigation area, because the CSS will configure the bold font weight.
3. Find the company name ("Pacific Trails Resort") in the first paragraph below the h2. Configure a span that contains this text. Assign the span element to the `resort` class.
4. Look for the company name ("Pacific Trails Resort") directly above the street address. Configure a span that contains this text. Assign the span element to the `resort` class.
5. Configure the page footer area. Remove the `<small>` and `<i>` elements—they are no longer needed since CSS is now used to configure the text.

Save the index.html file, and test in a browser. Your page should look similar to the one shown in Figure 3.33 except that your page content will be left-aligned instead of indented from the margins. Don't worry—you'll configure your page layout in Task 5 of this case study.

Task 4: The Yurts Page. Launch a text editor, and open the yurts.html file. You will modify this file in a similar manner: Add the `<link>` element, configure the navigation area, and configure the page footer area. Delete the strong tags contained within each dt element. Save and test your new yurts.html page. It should look similar to the one shown in Figure 3.34 except for the alignment.

Task 5: Center Page Layout with CSS. Modify pacific.css, index.html, and yurts.html to configure page content that is centered with 80% width. Refer to Hands-On Practice 3.9 if necessary.

1. Launch a text editor, and open the pacific.css file. Add a style rule for an id named `wrapper` with `width` set to 80%, `margin-right` set to auto, and `margin-left` set to auto.
2. Launch a text editor, and open the index.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your index.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.33.
3. Launch a text editor and open the yurts.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your yurts.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.34.

Experiment with modifying the pacific.css file. Change the page background color, the font family, and so on. Test your pages in a browser. Isn't it amazing how a change in a single file can affect multiple files when external style sheets are used?

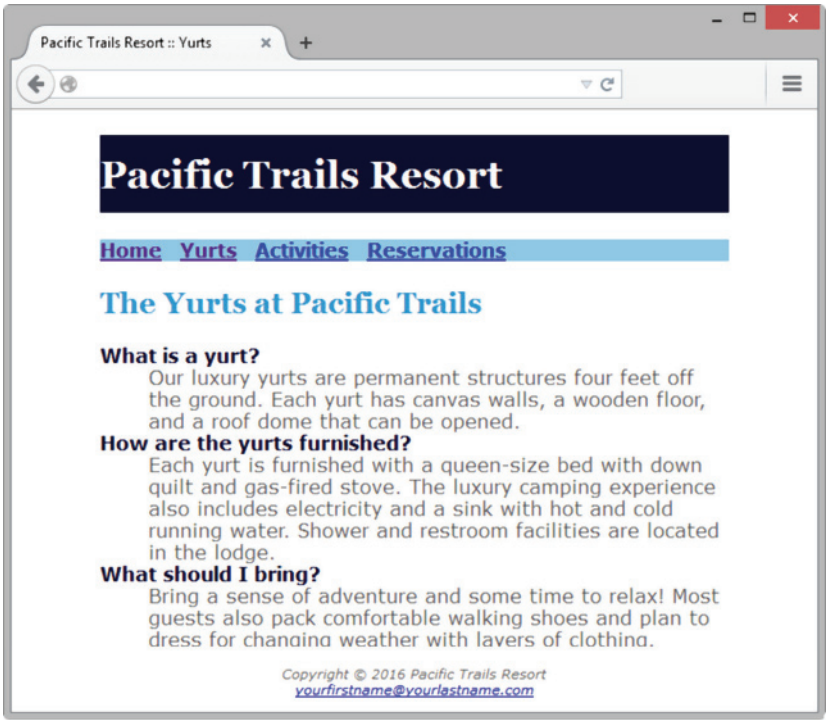

Figure 3.34 New yurts.html page

Path of Light Yoga Studio

See Chapter 2 for an introduction to the Path of Light Yoga Studio Case Study. Figure 2.42 shows a site map for the Path of Light Yoga Studio website. The Home page and Classes page were created in Chapter 2. You will develop a new version of this website that uses an external style sheet to configure text and color. Figure 2.43 depicts the wireframe page layout.

You have the following tasks:

1. Create a new folder for this Path of Light Yoga Studio case study.
2. Create an external style sheet named yoga.css that configures the color and text for the Path of Light Yoga Studio website.

3. Modify the Home page to utilize an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.35.
4. Modify the Classes page to be consistent with the new Home page.
5. Configure centered page layout.

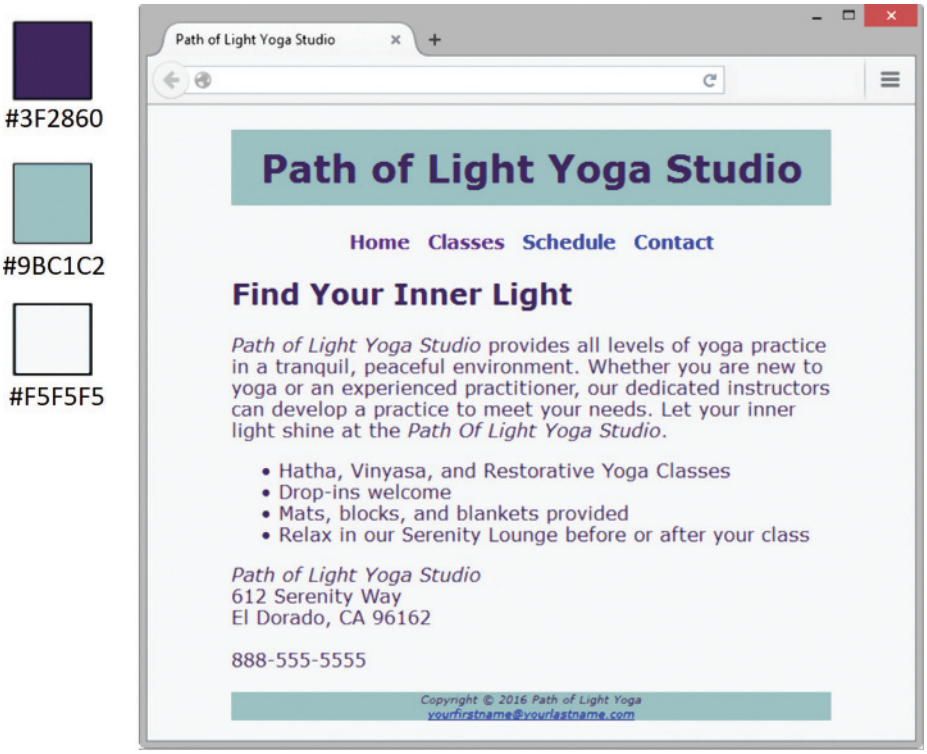

Figure 3.35 New Path of Light Yoga Studio index.html

Hands-On Practice Case Study

Task 1: Create a folder on your hard drive or portable storage device called yogacss. Copy all the files from your Chapter 2 yoga folder into the yogacss folder.

Task 2: The External Style Sheet. You will use a text editor to create an external style sheet named yoga.css. Code the CSS to configure the following:

1. Global styles for the document (use the body element selector) with background color `#F5F5F5`; text color `#3F2860`; and Verdana, Arial, or any sans-serif font.
2. Styles for the header element that configure background color `#9BC1C2` with centered text.
3. Styles for the h1 element selector that configure 200% line height.
4. Styles for the nav element selector that configure centered and bold font.
5. Styles for the anchor elements within the nav area to eliminate the default underline (Hint: use the `nav a` selector).
6. Styles for a class named `studio` that configures italic text.
7. Styles for the footer element selector with `#9BC1C2` background color, small font size (`.60em`), and italic, centered text.

Save the file as yoga.css in the yogacss folder. Check your syntax with the CSS validator (http://jigsaw.w3.org/css-validator). Correct and retest if necessary.

Task 3: The Home Page. Launch a text editor, and open the index.html file. You will modify this file to apply styles from the yoga.css external style sheet.

1. Add a `<link>` element to associate the web page with the yoga.css external style sheet file.
2. Configure the navigation area. Remove the `<b>` element from the navigation area, because the CSS will configure the bold font style.
3. Look in the main content area for the company name ("Path of Light Yoga Studio") and configure a span element to contain this text each time it appears. Assign each span element to the `studio` class.
4. Configure the page footer area. Remove the `<small>` and `<i>` elements—they are no longer needed since CSS is now used to configure the text.

Save the index.html file, and test in a browser. Your page should look similar to the one shown in Figure 3.35.

Task 4: The Classes Page. Launch a text editor, and open the classes.html file. You will modify this file in a similar manner: Add the `<link>` element, configure the navigation area, and configure the page footer area. Save and test your new classes.html page. It should look similar to the one shown in Figure 3.36.

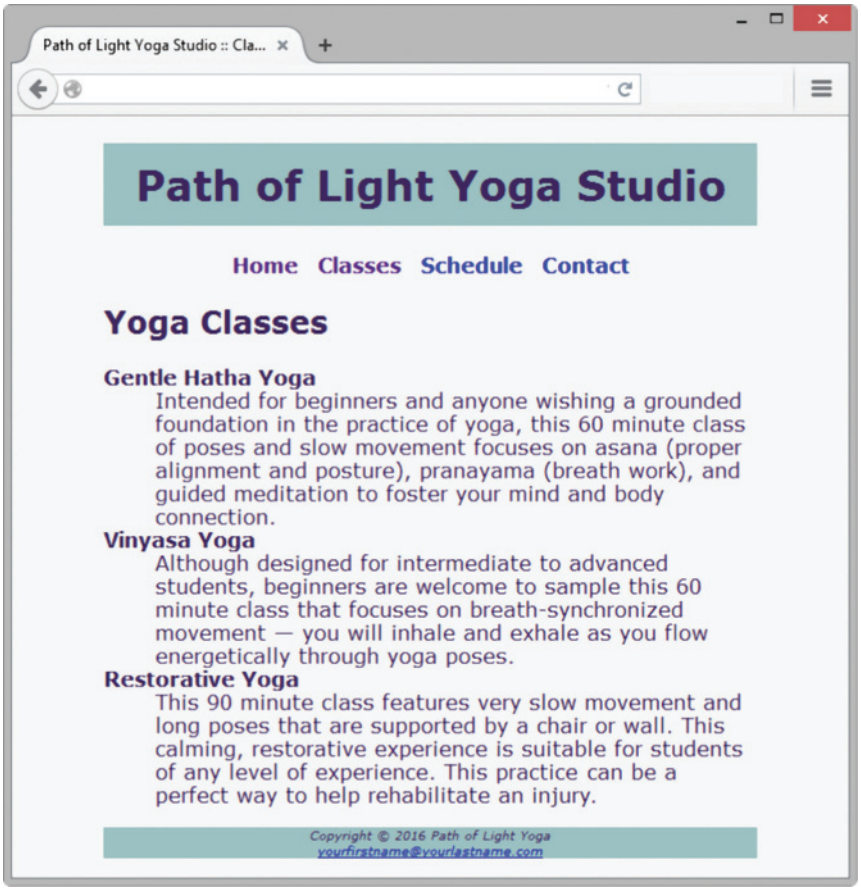

Figure 3.36 New classes.html page

Task 5: Center Page Layout with CSS. Modify yoga.css, index.html, and classes.html to configure page content that is centered with 80% width. Refer to Hands-On Practice 3.9 if necessary.

1. Launch a text editor, and open the yoga.css file. Add a style rule for an id named `wrapper` with `width` set to 80%, `margin-right` set to auto, and `margin-left` set to auto.
2. Launch a text editor, and open the index.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your index.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.35.
3. Launch a text editor, and open the classes.html file. Add the HTML code to configure a div element assigned to the id `wrapper` that "wraps," or contains, the code within the body section. Save and test your classes.html page in a browser and you'll notice that the page content is now centered within the browser viewport as shown in Figure 3.36

Experiment with modifying the yoga.css file. Change the page background color, the font family, and so on. Test your pages in a browser. Notice how a change in a single file can affect multiple files when external style sheets are used.