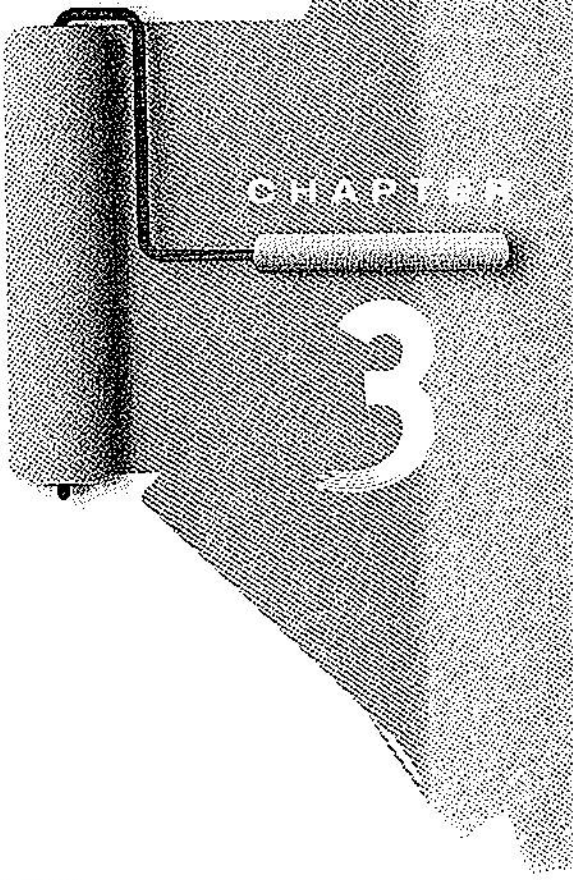


Configuring Color and Text with CSS

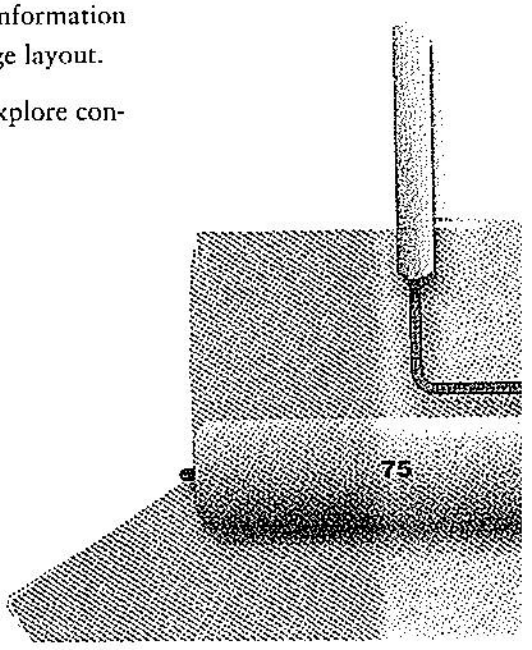
Chapter Objectives

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
- Use 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
- Create CSS `class` and `id` selectors
- Validate CSS

Now that you have been introduced to XHTML, you are ready to explore Cascading Style Sheets (CSS). CSS is not new—it was first proposed as a standard by the W3C in 1996. However, browsers and other user agents have only supported this technology consistently for the past few years. Now that there is fairly steady support, Web developers have begun to use CSS to separate the presentation style of a Web page from the information on the Web page itself. CSS is used to configure text, color, and page layout.

This chapter introduces you to the use of CSS on the Web as you explore configuring color and text.



3.1 Overview of Cascading Style Sheets

For years, style sheets have been used in desktop publishing to apply typographical styles and spacing instructions to printed media. CSS provides this functionality (and much more) for Web developers. It 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 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. Its description of CSS can be found at http://www.w3.org/Style/. Be aware that CSS, even though it has been in use for many years, is still considered an emerging technology and the two most popular browsers do not support it in exactly the same way. There are CSS reference pages at http://web.archive.org/web/20040202153928/http://devedge.netscape.com/library/xref/2003/css-support/css1/mastergrid.html and http://www.westciv.com/style_master/academy/browser_support that list the way styles are supported by various browsers and platforms. This chapter concentrates on those aspects of CSS that are well supported by popular browsers.

Advantages of Cascading Style Sheets

There are several advantages to using CSS:

- **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 XHTML remains intact. This means that if your client decides to change the background color from red to white you only need to change one file that contains the styles, instead of 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 it's possible to complete the modifications by changing the style sheet only.

Do you see that there might be advantages to using CSS? You may be wondering if there are any disadvantages to using CSS. In fact, there is one large disadvantage—CSS technology is not yet uniformly supported by browsers. This disadvantage will be less of an issue in the future as browsers comply with standards.

Types of Cascading Style Sheets

There are four methods used to incorporate CSS technology in a Web site: inline, embedded, external, and imported.

- **Inline styles** are coded in the body of the Web page as an attribute of an XHTML tag. The style only applies to the specific element that contains it as an attribute.
- **Embedded styles** are defined in the header 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 using a `<link />` element in the header 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. The @import directive is only supported by version 5.0 and later browsers. This chapter concentrates on the other three uses of CSS.

Introduction to CSS Syntax

Style sheets are composed of rules that describe the styling to be applied. Each rule contains a selector and a declaration. The selector can be an XHTML element, a class name (that you create yourself), or an id name (that you create yourself). This example concentrates on applying styles to XHTML elements. The declaration is the property you are setting (such as color or typeface) and the value you are assigning to it.

For example, the CSS rule shown in Figure 3.1 would set the color of the text used on a Web page to blue. The selector is the XHTML body tag and the declaration sets the **`color` property** to the value of blue.

Figure 3.1
Using CSS to set the text color to blue

If you wanted the background color of the Web page to be yellow, the CSS rule could be expanded as follows:

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

This could also be written using hexadecimal color values as follows:

```
body { color: #0000FF; background-color: #FFFF00; }
```

Notice that both the `background-color` and text `color` properties were configured in the previous example. To avoid surprising results caused by default browser colors, the W3C recommends that the **`background-color` property** is set when the text color is configured.

Have you ever wondered why some text-based links are not underlined? This can be accomplished with a style applied to the anchor tag. The following style rule selects the anchor tag (denoted by a) and sets the **`text-decoration` property** (the underline) to `none`:

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

You might be asking yourself how you would know what properties and values are allowed to be used. This chapter introduces you to some of the CSS properties commonly used to configure color and text. Table 3.1 presents a summary of the CSS properties used in this chapter. Appendix C, CSS Property Reference, contains a more detailed list. In the next sections, we'll take a look at how color is used on Web pages and we'll explore how to use CSS to configure color.

Table 3.1 CSS properties used in this chapter

Property	Description	Values
`background-color`	Background color on the Web page	Any valid color
`color`	Text color	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`	The size of the text font	This varies; `pt` (standard font point sizes), `px` (pixels), the unit `em` (which corresponds to the width of the capital M of the current font), or percentages; the text values `xx-small`, `small`, `medium`, `large`, `x-large`, and `xx-large` are also valid
`font-style`	The style of the font	`normal`, `italic`, `oblique`
`font-weight`	The "boldness" or weight of the font	This varies; the text values `normal`, `bold`, `bolder`, and `lighter` can be used; the numeric values `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, and `900` can be used
`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 be double-spaced
`margin`	Shorthand notation to configure the margin surrounding an element	A numeric value (`px` or `em`); for example, `body { margin: 0px}` will set the page margins in the document to zero
`margin-left`	Configures the space in the left margin of the element	A numeric value (`px` or `em`) or `auto`
`margin-right`	Configures the space in the right margin of the element	A numeric value (`px` or `em`) or `auto`
`text-align`	The alignment of text	`center`, `justify`, `left`, `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
`width`	The width of an element	A numeric value (`px` or `em`), numeric percentage, or `auto` (default)

3.2 Using Color on Web Pages

Monitors display color as a combination of different intensities of red, green, and blue, also known as **RGB color**. RGB intensity values are numerical from 0 to 255. Each

RGB color will have three values, one each for red, green, and blue. These are always listed in that order (red, green, blue) and specify the numerical value of each color used. For example, the RGB values for red are (255,0,0)—all red, no green and no blue. The RGB values for blue are (0,0,255)—no red, no green, and all blue. These colors can also be specified using hexadecimal 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. When a hexadecimal value is used to specify RGB color, the numeric value pairs range from 00 to FF (0 to 255 in Base 10). The hexadecimal value contains three numeric value pairs written sequentially as one number. Each pair is associated with the amount of red, green, and blue displayed. Using this notation, the color red would be specified as `#FF0000` and the color blue as `#0000FF`. The `#` symbol signifies that the value is in 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 3.2 (shown also in the color insert section) for an excerpt from the color chart at http://webdevfoundations.net/color.

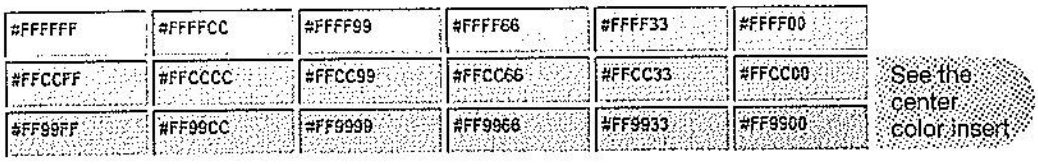


Figure 3.2
Partial color chart

Take a few moments to examine the color chart. You will observe a display of colors and their associated hexadecimal RGB values in hexadecimal. You may notice that there is a pattern to the hexadecimal numbers (pairs of 00, 33, 66, 99, CC, or FF). This pattern signifies a color on the Web Safe Color Palette (more on this later). As you examine the color chart further, you will see a list of colors using color names. Some Web developers find it easier to use the color names. However, the names are not uniformly supported by all versions of all browsers, so the W3C recommends using numeric color values instead of color names.

Web Color Palette

A Web developer usually has no way of knowing what type of computer or browser the Web site visitors will be using. The various operating systems and browsers display colors differently, and sometimes not at all. The Web Safe Color Palette, also known as the Web Color Palette, is a collection of 216 colors that display the same on both the Windows and Mac OS platforms. It is easy to tell if a color is on the Web Color Palette when you consider the individual hexadecimal RGB value pairs. The values of 00, 33, 66, 99, CC, and FF are the only values for hexadecimal RGB value pairs on the Web Color Palette. Take another look at the color chart at http://webdevfoundations.net/color and note that all the colors listed by RGB follow this numbering scheme—they comprise the Web Color Palette. See Figure 3.3 (shown also in the color insert section) for a comparison of a Web Safe Color, `#CC0000`, and a non-Web safe color, `#880000`. Both are a shade of red; however, the Web safe color will display predictably across

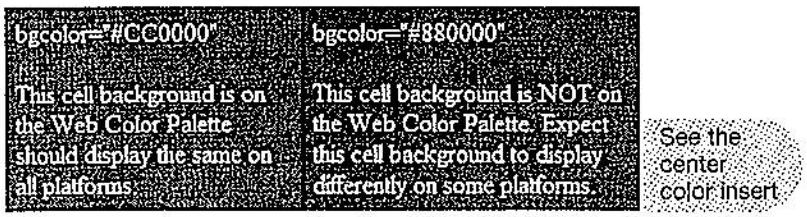


Figure 3.3
Web safe colors display predictably

Windows and Mac OS platforms and the other color will not. Using Web safe colors has become less important now that most monitors display billions of colors. The Web Color Palette is rather limited and it is common for today's Web designers to choose colors creatively rather than select them from the Web Color Palette.

Making Color Choices

You may be wondering how to select colors to display on Web pages. One easy way to choose colors is to use a monochromatic color scheme—all shades or tints of the same color. Try the Color Blender at http://meyerweb.com/eric/tools/color-blend to select colors for a monochromatic color scheme. Another way to create a color scheme is to base it on a photograph or image. Visit http://www.colr.org to generate a color scheme based on an image from the Web or one that you upload. If you have a favorite color and would like to create a color scheme around it, visit one of the following sites that suggest color schemes:

- http://colorsontheweb.com/colorwizard.asp
- http://kuler.Adobe.com
- http://www.steeldolphin.com/color_scheme.html
- http://wellstyled.com/tools/colorscheme2/index-en.html
- http://www.colors4webmasters.com/safecolor/index.htm

Accessibility and Color

Focus on Accessibility

While using color can help you create a compelling Web page, keep in mind that not all your visitors will see or be able to distinguish between colors. Some visitors will use a screen reader and will not experience your colors, so your information must be clearly conveyed even if colors cannot be viewed. Other visitors may be challenged with color vision deficiency (color blindness) and will not see the colors as you intended. According to Vischeck (http://www.vischeck.com/vischeck/) about 1 out of 20 people experience some type of color deficiency. To increase Web page accessibility, choose background and text colors with a high amount of contrast. The choice of colors is important—avoid using red, green, brown, gray, or purple next to each other. White, black, and shades of blue and yellow are easier for individuals with color vision deficiencies to differentiate. Visit http://www.vischeck.com/vischeck/vischeckURL.php to simulate how a person with a color deficiency experiences the colors on a Web page. Using color on Web pages will be revisited in Chapter 5, Web Design.

3.3 Configuring Color with Inline CSS

Now that you are aware of how color on Web pages is specified and where to get ideas for color schemes on Web pages, let's start configuring color with inline styles. **Inline** styles are coded as attributes on XHTML tags.

The Style Attribute

The **`style` attribute** is used with the value of the style rule declaration you need to set. 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 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 sets the text in the heading to red and italic:

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

The following code example uses an inline style to set the background color to `green` and text color to `white`:

```
<p style="background-color:green;color:white">This paragraph is using
an inline style.</p>
```

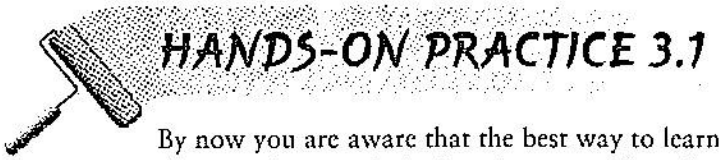
Are there different ways to configure colors using CSS?

CSS syntax allows you to configure colors in a number of ways, including hexadecimal color values, color names, and decimal color values. For example, Table 3.2 shows the syntax for setting the color of text in a paragraph to red.

The examples in this book use either hexadecimal color value or color name to configure colors using CSS. The color chart on this textbook's companion Web site at http://webdevfoundations.net/color provides examples of the color created by hexadecimal values in the Web Color Palette.

Table 3.2 Syntax for setting the color of text in a paragraph to red

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)
`p { color: rgb(255,0,0) }`	Decimal color value (RGB triplet)



HANDS-ON PRACTICE 3.1

By now you are aware that the best way to learn new coding technologies is to practice them. In this Hands-On Practice you will configure a paragraph using inline styles. The styles will specify a green background with white text. A sample is shown in Figure 3.4.

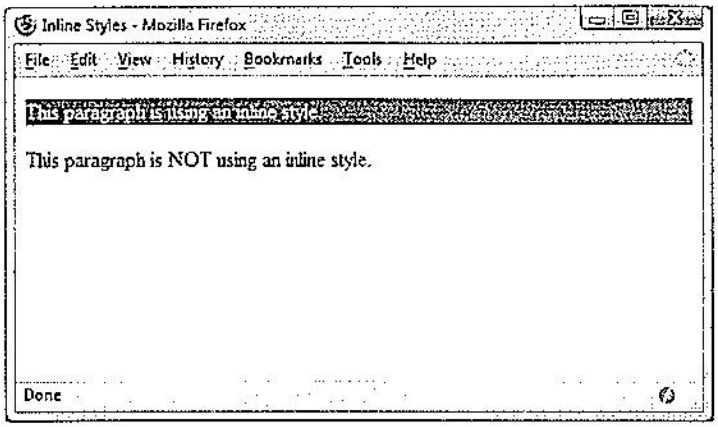


Figure 3.4
Web page using inline styles

Launch Notepad and type in the following XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Inline Styles</title>
</head>
<body>
<p style="background-color:#00FF00;color:#FFFFFF">This paragraph is
using an inline style.</p>
<p>This paragraph is NOT using an inline style.</p>
</body>
</html>
```

Save your file as inline.html. Test your page in a browser and compare it with Figure 3.4. The student files contain a sample solution at Chapter3/inline.html. Note that the paragraph that used a style has the green background and white text. The paragraph that does not use a style is displayed using default browser settings.

3.4 Configuring Color with Embedded Styles

In the previous Hands-On Practice you added inline styles for one of the paragraphs. You needed to code a style attribute on the paragraph element. But what if you needed to configure the styles for ten or twenty paragraphs instead of just one. Using inline styles, you might be doing a lot of repetitive coding! While inline styles apply to one XHTML element, embedded styles apply to an entire Web page.

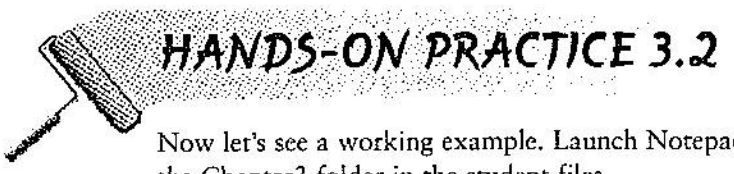
The Style Element

Embedded styles are placed within a `<style>` element located in the header section of a Web page. The opening `<style>` tag begins the embedded style rules and the closing `</style>` tag ends the area containing embedded style rules. When using the `<style>` tag, you do not need the `style` attribute. However, the `<style>` tag does use a `type` attribute that should have the value of `"text/css"`.

The following code is an example of a `<style>` tag that uses embedded styles to set the text color and background color of the page.

```
<style type="text/css">
body { background-color: #000000;
       color: #FFFFFF;
}
</style>
```

The indentation 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` selector.



Now let's see a working example. Launch Notepad 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 3.5.

Figure 3.5
The Web page without any styles

Open the file in Notepad and view the source code. Notice that the XHTML code uses the <h1>, <h2>, <p>, <ul>, and <li> elements. In this Hands-On Practice you'll code embedded styles to configure selected background and text colors. You'll use the body selector to configure the default background color (#E6E6FA) and default text color (#191970) for the entire page. You'll also use the h1 and h2 selectors to configure different background and text colors for the heading areas.

Edit the embedded.html page in Notepad and add the following code below <title> element in the head section of the Web page.

```
<style type="text/css">
body {  background-color: #E6E6FA;
        color: #191970;
}
h1 {  background-color: #191970;
      color: #E6E6FA;
}
h2 {  background-color: #AEAED4;
      color: #191970;
}
</style>
```

Save and test your file in a browser. Figure 3.6 (shown also in the color insert section) displays the Web page along with color swatches. The monochromatic color scheme was chosen using the Color Blender at http://meyerweb.com/eric/tools/color-blend. Notice how the repetition of a limited number of colors adds interest and unifies the design of the Web page.

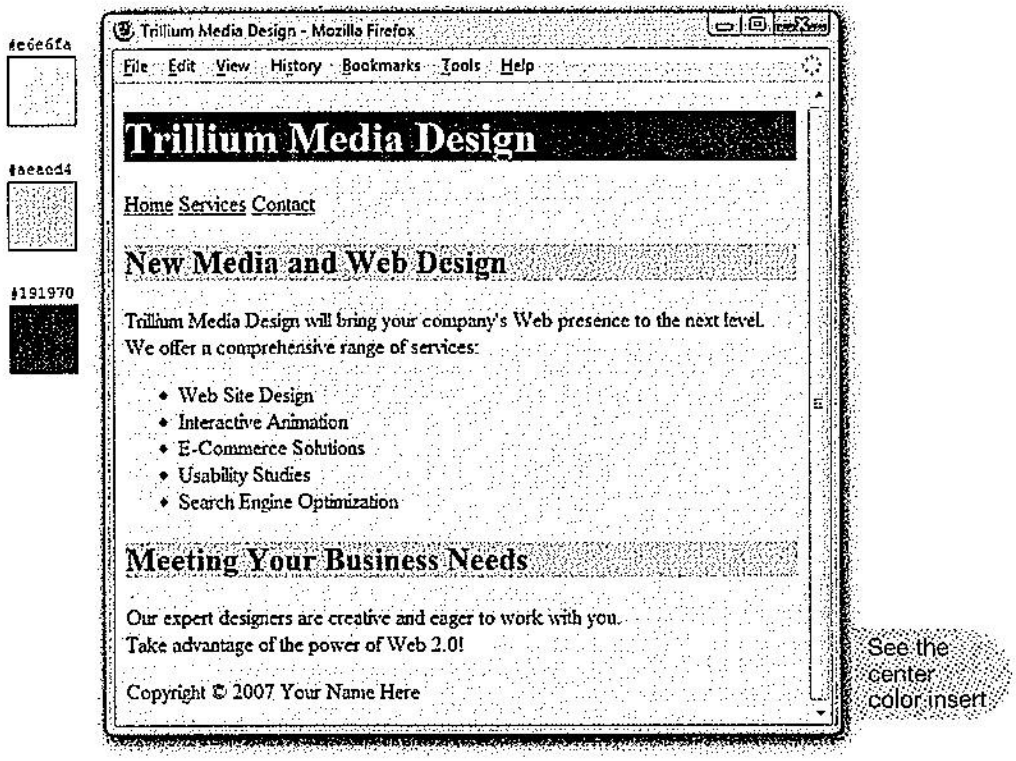


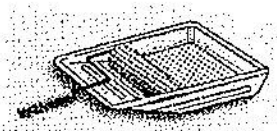
Figure 3.6 embedded.html with styles applied

View the source code for embedded.html and review the CSS and XHTML 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 only needed to code the styles for the `h2` selector once (in the head section) and *both* of the `<h2>` XHTML elements applied the `h2` style. This is more efficient than coding the same inline style on each `<h2>` element.

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 `:` and `;` symbols in the right spots—they are easy to confuse. The `:` should separate the properties from their values, the `;` should be placed between each *`property:value`* configuration.
- Check that you are not using = signs instead of `:` between each property and its value.
- Verify that the `{` and `}` symbols are properly placed around the style rules for each selector.
- Check the syntax of your selectors, their properties, and property values for correct usage.
- If part of your CSS works, and part doesn't—read through the CSS and check to determine the first rule that is not applied. Often the error is in the rule above the rule that is not applied.
- Use a program 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 3.9 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 properties: text color and 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 XHTML to configure some characteristics of text on Web pages, including logical style tags (such as the `<strong>` element) and physical style tags (such as the `<small>` element). You've also already configured text color using the CSS `color` property. In this section you'll learn how to use CSS to configure additional characteristics of text, including font typeface, font weight, font style, and font size. Using CSS to configure text is more flexible (especially when using an external style sheet as you'll discover later in the chapter) than using XHTML elements and is the method preferred by today's Web developers.

CSS and Fonts

Let's take a closer look at the CSS properties useful for configuring fonts: `font-weight`, `font-style`, `font-size`, and `font-family`.

The **`font-weight`** property configures the boldness of the text. You can use either numeric values (`100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, and `900`) or text values (including `normal` (default), `bold`, `bolder`, and `lighter`). Configuring the CSS rule `font-weight:bold` has a similar effect as the `<strong>` or `<b>` XHTML element.

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

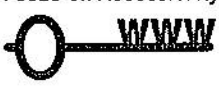
The **`font-size`** property sets the size of the font. There are a wide variety of text and numeric values. Text values for `font-size` include `xx-small`, `x-small`, `small`, `medium` (default), `large`, `x-large`, and `xx-large`. Valid numeric values include units of `px` (pixels), `pt` (standard font point sizes), percentage values, and `em`. Figure 3.7 demonstrates examples of text with various `font-size` configurations displayed in the Firefox browser on a monitor set to 1440×900 screen resolution. Compare font sizes on your own computer— launch a browser and view chapter3/fonts.html in the student files.

Figure 3.7
A sampling of CSS font-size values

Text Values	Em Units	Px Units	Pt Units	Percentage
x-small	.5 em	10 px	9 pt	50%
x-small	.60 em	11 px	8 pt	60%
small	.75 em	13 px	10 pt	75%
medium	1 em	16 px	12 pt	100%
x-large	1.5 em	24 px	18 pt	150%
xx-large	2 em	28 px	24 pt	200%

Be aware that the text values and the `pt` (point) unit size are browser dependent. For example, text configured with the CSS rule `font-size: 12pt` may look different when various browsers are used to display Web pages. The `px` (pixel) unit is monitor resolution dependent and looks different depending on the screen resolution used. The `em` unit is a relative font unit that has its roots in the print industry. Recall that printers used to 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). With this in mind, 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.

Focus on Accessibility



With all these available choices, what's the best way to configure font-sizes? The W3C recommends the use of em units or percentages in their specification for CSS2 at http://www.w3.org/TR/REC-CSS2/fonts.html. So, usually the best choice is either em units or percentages. However, the W3C's Web Accessibility Initiative WCAG 1.0 lists the use of relative font size values such as em units and percentages to be a Priority 2 level guideline (a "should" not a "must"). In addition, modern browsers such as

Firefox and Opera allow visitors to increase (or "zoom") the text size on the page easily even if nonrelative units, such as px, are used to configure font size. Expect to see more browser support of page customization and zoom features in the future. For certain graphic-dependent designs that require "pixel-perfect" rendering, px units might be more appropriate than em units or percentages— it's up to you to choose. As you work through the Hands-On Practice and Case Study exercises in this book, you'll gain experience using a variety of font-size configurations. In all cases, it is crucial to test your Web pages in a variety of client platforms (including browser and monitor resolution). This testing is part of the Web design and development process. Statistics available at http://thecounter.com indicate that at the time this was written Internet Explorer at 1028×768 or 1280×1024 resolution is most commonly used, although use of the Firefox browser continues to grow.

The **font-family property** configures font typefaces. A Web browser displays text using the fonts that have been installed on the user's computer. For example, the CSS rule `font-family: Arial` causes the browser to display text using Arial instead of the default browser font. When a font is specified that is not installed on the Web visitor's computer, the default font is substituted. Times New Roman is the default font displayed by most Web browsers. You can list multiple fonts and categories for the value of the font face attribute. The browser will attempt to use the fonts in the order listed. When processing a CSS rule such as `font-family: Arial, Verdana, sans-serif`, the browser will use Arial if it is installed. If Arial is not installed, the browser will use Verdana if it is installed. If neither Arial nor Verdana are installed, the browser will use any sans-serif font installed on the computer. Finally, if no sans-serif fonts are installed on the computer, the default font face will be used. Table 3.3 shows font family categories and some common font typefaces.

Table 3.3 Common fonts

Font-family Category	Font Typeface
serif	Times New Roman, Georgia, Times
sans-serif	Arial, Verdana, Geneva
monospace	Courier New, Lucida Console
cursive	Brush Script MT, Comic Sans MS
fantasy	Jokerman, Curlz MT

Now that you are familiar with font configuration using CSS, we'll quickly explore three other CSS properties that modify the appearance of text: `text-align`, `text-decoration`, and `line-height`.

As you already know, the default alignment of text on a Web page is at the left margin, called left alignment. The **text-align property** is used to specify the alignment of text. Values for the `text-align` property are `left` (default), `right`, and `center`.

Have you ever seen a hyperlink on a Web page that was not underlined? This is typically configured with the **text-decoration property** (`text-decoration: none` CSS rule). See Table 3.1 for additional values that are less commonly used with the `text-decoration` property.

The **line-height** property modifies the default height of a line of text. For example, code `line-height: 200%` to configure text to appear double-spaced.

HANDS-ON PRACTICE 3.3

Now that you've got a collection of new CSS properties for font and text configuration, let's try them out and modify the embedded.html page. Launch Notepad and open embedded.html. You'll code additional CSS style rules to configure the text on the page. When complete, your Web page will look similar to the one shown in Figure 3.8.

Figure 3.8
CSS configures the text on the Web 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 most text using a sans-serif font. The new font typeface style rule will apply to the entire Web page unless more specific styles rules are applied to a 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 embedded2.html and test it in a browser. Your page should look similar to the one shown in Figure 3.9. Notice that just a single line of CSS changed the font typeface of all the text on the page!

Figure 3.9
Text is displayed using a sans-serif font

Configure the `h1` Selector

You will configure the `line-height` and `font-family` CSS properties. Set the `line-height` property to 200%—this will add a bit of empty space above and below the heading text. (In Chapter 6 you'll 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 below. 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.

```
h1 { background-color: #191970;
     color: #E6E6FA;
     line-height: 200%;
     font-family: Georgia, "Times New Roman", serif;
}
```

Save and test your page in a browser. If you notice that the Trillium Media Design text seems to crowd the left margin, add a ` ` nonbreaking space special character in the body of the Web page after the opening `<h1>` tag.

Configure the `h2` Selector

Configure the CSS rule to use the same font typeface as the `h1` selector.

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

Add a New Paragraph Element Selector

Configure text in paragraphs to display just slightly smaller than the default text size. Use the `font-size` property set to `.90em`.

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

Configure the Unordered List

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

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

Save your page as embedded2.html and test it in a browser. Your page should look similar to the one shown in Figure 3.8. The student files contain a sample solution at Chapter3/embedded2.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 paragraphs to display in exactly the same way? While you could add inline styles to the Web page code, that's 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 XHTML tag or more than one class?

Yes, you can apply the same style rules to multiple selectors (such as XHTML elements, `classes`, or `ids`) by listing the selectors in front of the rule. The code sample below shows the `font-size` of `1em` being applied to both the paragraph and line item elements.

```
p, li { font-size: 1em; }
```

3.6 The `Class` and `Id` Selectors

The `Class` Selector

There are times when you'd like to apply a CSS rule to a certain class of elements on a Web page and not necessarily tie the style to a particular XHTML tag. This is when you use the **`class` selector**. For example, perhaps you would prefer if the paragraph containing the navigation area information in embedded.2html was displayed with large, bold text. When setting a style for a class, configure the class name as the selector. Place a dot or period (.) in front of the class name in the style sheet. The following code configures a class called `nav` in a style sheet.

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

The styles set in the nav class can be applied to any XHTML element you wish. You do this by using the **class attribute**, such as `class="nav"`. Do not write the dot in front of the class value in the XHTML tag where the class is being applied. The following code will apply the nav class styles to a <p> element:

```
<p class="nav">This paragraph will be displayed using the styles in
the nav class.</p>
```

Why is the class called nav?

You can choose almost any name you wish for a CSS class. 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 was changed to display the area differently; but a structural class name such as `nav`, `logo`, `footer`, `content`, or `subheading` is meaningful regardless of how the area is configured. Here are more hints for class names:

- Use short but descriptive names.
- Avoid class names that are the same as XHTML element names—they could be confusing to anyone working on the page.
- Both letters and numbers may be used.
- Avoid spaces in class names.
- Class names are not case sensitive, but consistency will make page maintenance easier.

A final tip about CSS classes is to 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.

The Id Selector

Use an **id selector** instead of a `class` selector if you plan to identify and apply a CSS rule uniquely to a single area on a Web page. For example, perhaps you would prefer if the paragraph containing the copyright information in the page footer area of embedded2.html was displayed with small italics text. While a `class` selector could be used, an `id` selector is more appropriate if your page layout contains a single footer area. For example, you can create a style for an `id` named `footer` to configure the footer area to use small, italicized text. When setting a style for an `id`, place a hash mark (`#`) in front of the `id` name in the style sheet. The following code will configure an `id` called new in a style sheet:

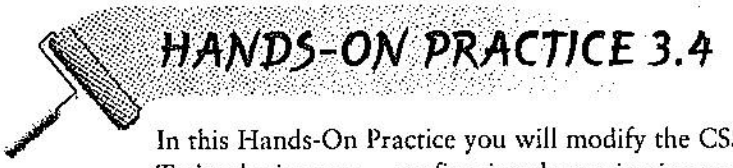
```
#footer {  font-size: .75em;
           font-style: italic;
}
```

The styles set in the `footer id` can be applied to any XHTML element you wish by using the **id attribute**, `id="footer"`. Do not write the `#` in front of the class value in the XHTML tag.

The following code will apply the `footer id` styles to a `<p>` tag:

```
<p id="footer">This paragraph will be displayed using styles
configured in the footer id.</p>
```

Using CSS with an `id` selector is similar to using CSS with a `class` selector. It's common practice to use an `id` selector to refer to a *single* XHTML element and a `class` selector to refer to multiple XHTML elements.



In this Hands-On Practice you will modify the CSS and the XHTML in the Trillium Technologies page—configuring the navigation and page footer areas. Launch Notepad and open embedded2.html.

Configure the Navigation Area

The navigation links would be more prominent if they displayed in a larger and bolder font. Create a class named nav, which sets the `font-size` and `font-weight` properties. The code follows:

```
.nav { font-weight: bold;
       font-size: 1.25em;
}
```

Modify the opening paragraph tag of the navigation area. Add a `class` attribute that associates the paragraph with the nav class as follows:

```
<p class="nav"><a href="index.html">Home</a>
<a href="services.html">Services</a>
<a href="contact.html">Contact</a></p>
```

Configure the Footer Area

Create an `id` named `footer`, which sets the `font-size` and `font-style` properties.

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

Modify the opening paragraph tag of the `footer` area. Add an `id` attribute that associates the paragraph with the `id` class.

```
<p id="footer">Copyright &copy; 2007 Your Name Here</p>
```

Save your file embedded3.html and test it in a browser. Your page should look similar to the image shown in Figure 3.10. The student files contain a sample solution at Chapter3/embedded3.html. Notice how the `class` and `id` styles are applied.

Figure 3.10
Using classes and ids

3.7 The Div and Span XHTML Elements

The and XHTML elements are used along with CSS to format page areas. The block-level element configures a section or division on a Web page with a line break above and below. Use the tag when you need to format a section that is separated from the rest of the Web page by line breaks. The element is also useful to define a section that contains block-level elements, such as <p>, <blockquote>, <ul>, <ol>, and even other elements within it. In Chapter 6 you will see how elements are used to configure a page layout with CSS.

In contrast, the element defines a section on a Web page that is *not* physically separated from other areas by line breaks. Use the tag if you need to format an area that is contained within another, such as within a <p>, <blockquote>, <li>, or element.

HANDS-ON PRACTICE 3.5

You will experiment with the and elements in this Hands-On Practice. First, you will place the navigation area within a element. Next, you will configure a new class to format the company name when displayed within the text on the page and use the element to apply this class. Open the embedded3.html file in Notepad. Your Web page will look similar to the one shown in Figure 3.11 after the changes are complete.

Configure the Navigation Area

View the source code of embedded3.html and notice that the hyperlinks (anchor elements) in the navigation area are contained within a paragraph element. While this is

valid XHTML, it isn't the best choice *semantically* since the navigation is a list of hyperlinks and not a true paragraph of text. Replace the <p> tags with tags as follows:

```
<div class="nav"><a href="index.html">Home</a>
<a href="services.html">Services</a>
<a href="contact.html">Contact</a></div>
```

Save your file as embedded4.html and test in a browser. You'll notice that the navigation area does not look any different— however, "under the hood" the code is better semantically (see Chapter 7 for more information about this topic).

Configure the Company Name

View Figure 3.11 and notice that the company name, Trillium Technologies, is displayed in bold and serif font within the first paragraph. You'll code both CSS and XHTML to accomplish this. First, create a new CSS rule that configures a class called companyname as bold, serif font, and 1.25em in size. The code follows:

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

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

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

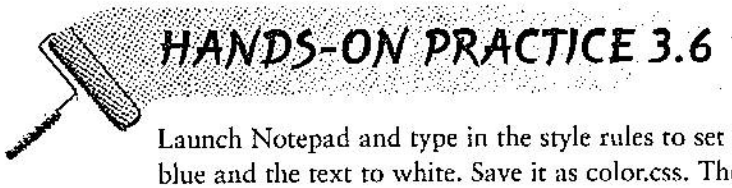
Save your file and test in a browser. Your page should look similar to the one shown in Figure 3.11. The student files contain a sample solution at Chapter3/embedded4.html.

Figure 3.11
This Web page uses and elements

3.8 Using External Style Sheets

External style sheets are contained in a text file separate from the XHTML documents. The `<link />` element is a self-contained tag used in the header section of an XHTML document to associate the style sheet with the Web page. This allows multiple Web pages to link to the same external style sheet file. The external style sheet text file is saved with the file extension .css and contains style rules only—it does not contain any XHTML tags.

The advantage of this technique 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 can save a Web developer much time and increase productivity. Let's get some practice with this useful technique.

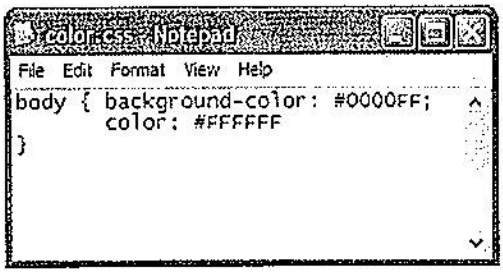


Launch Notepad and type in the style rules to set the `background-color` of a page to blue and the text to white. Save it as color.css. The code is as follows:

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

Figure 3.12 shows the external color.css style sheet displayed in Notepad. Notice that there is no XHTML in this file. `<style>` tags are not coded within an external style sheet. Only CSS rules (selectors, properties, and values) are coded in an external style sheet.

Figure 3.12
The external style sheet `color.css`



Next, associate that style to a Web page using the `<link />` element in the header section of the page. Three attributes are used with the `<link />` element to associate a Web page with an external style sheet: `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 a style sheet. The XHTML code to link color.css to a Web page is as follows:

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

Ready to try it out? Launch Notepad and type in the following XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External Styles</title>
  <link rel="stylesheet" href="color.css" type="text/css" />
</head>
<body>
  <p>This web page uses an external style sheet.</p>
</body>
</html>
```

Save the file as external.html in the same folder as color.css. Test your page in a browser. Your file should look similar to Figure 3.13.

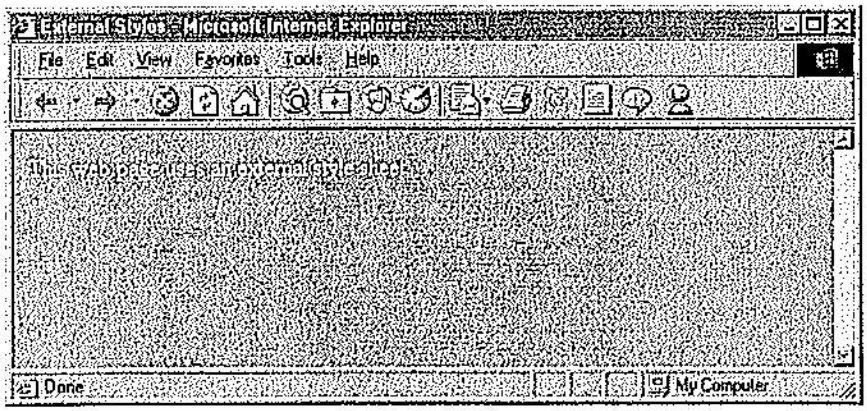


Figure 3.13
This page is associated with an external style sheet

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 only need to change a single file (color.css) instead of multiple files (all the Web pages). As mentioned earlier, this technique can boost productivity on a large site.

This is a simple example, but the advantage of having only a single file to update is significant for both small and large Web sites. In the next Hands-On Practice you'll modify the Trillium Technologies home page to use an external style sheet.

HANDS-ON PRACTICE 3.7

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

A version of the Trillium home page is in the student files. Open the embedded4.html file in a browser. The display should be the same as the Web page shown in Figure 3.11 from Hands-On Practice 3.5.

Now that you've seen what you're working with let's begin. Launch Notepad and save the file as index.html in a folder called trillium. 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 to copy the CSS code to the clipboard. You will place the CSS in a new file. Launch Notepad, paste the CSS style rules (use Edit, Paste or press the Ctrl+V keys), and save the file as trillium.css. See Figure 3.14 for a screenshot of the new trillium.css file in Notepad. Notice that there are no XHTML elements in trillium.css— not even the `<style>` element. The file contains CSS rules only.

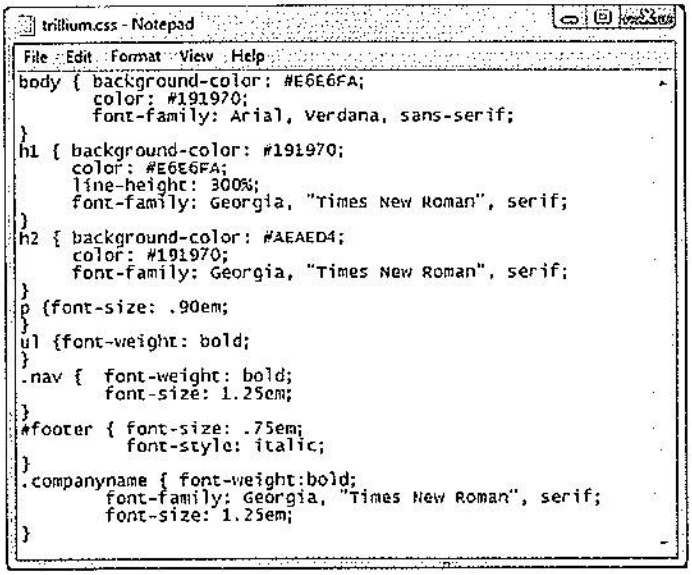


Figure 3.14
The external style sheet named trillium.css

Next, edit the index.html file in Notepad. 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>` element code follows:

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

Save the file and test in a browser. Your Web page should look just like the one shown in Figure 3.11. 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'll 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 3.15. Notice that although the structure of the page is similar to the home page, the styling of the text and colors are absent.

Launch Notepad to edit the services.html file. If you view the XHTML code you'll notice that this page is ready for our trillium.css styles—for example, the nav class and `footer` id have been coded as attributes in the corresponding navigation and page footer areas. All that's left for you to do is to code an XHTML `<link>` element to associate the services.html Web page with the trillium.css external style sheet. Place the following code in the header section above the closing `</head>` tag:

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

Save your file and test in a browser. Your page should look similar to Figure 3.16—the CSS rules have been applied!

Figure 3.15
The services.html page is not associated with a style sheet

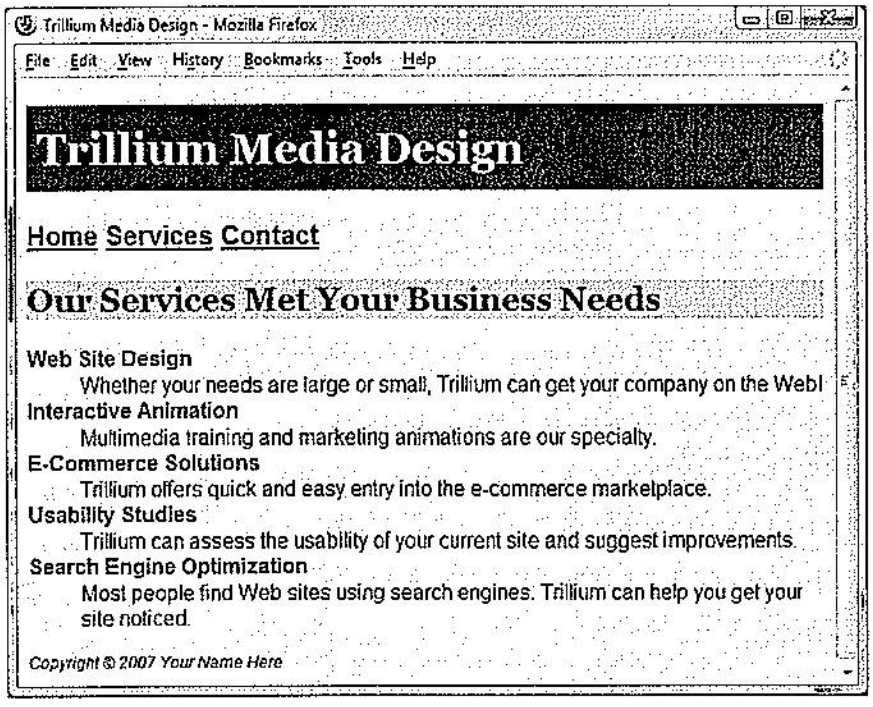
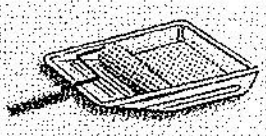


Figure 3.16
The services.html page has been associated with trillium.css

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 Chapter3/3.7 folder.

Notice that when using an external style sheet, if the use of color or fonts on the page ever needs to be changed, modifications only need to be made to the external style sheet. Think about how this 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 and other technologies such as Extensible Style Sheet Language (XSL) 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 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 Web site, 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` selector.
- Identify typical elements used for organization in the page (such as `<h1>`, `<h3>`, and so on) and declare style rules for these if different from default.
- Identify various page areas such as logo, navigation, footer, and so on—and 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 Web site.

3.9 Centering XHTML Elements with CSS

Recall that by default, XHTML elements are left-aligned— they begin at the left margin. In Chapter 2 (Hands-On Practice 2.4) you used the XHTML `align="center"` attribute to center text on a Web page. While this is valid, it is more efficient to configure the alignment using CSS. The CSS `text-align` property configures the alignment of text. The CSS code sample below configures an `<h1>` XHTML 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.

Center the Page Content

A popular page layout design that is easy to accomplish with just a few lines of CSS is to center the entire content of a Web page within a browser window. The Web page shown in Figure 3.17 uses this type of page layout.

Figure 3.17
The Web page content is centered in the browser window

Compare the left and right margins of Figure 3.17 to the Web page displayed in Figure 3.11. It's easy to configure this centered layout. Create a `<div>` to contain, or wrap, the page content and then configure it with the CSS **`margin-left` property**, **`margin-right` property**, and **`width` property**. 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. 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-level element. The CSS code sample below sets the width of an `id` named `wrapper` to 700 pixels and centered (using `margin-left:auto` and `margin-right:auto`).

```
#wrapper { width: 700px;
           margin-left: auto;
           margin-right: auto;
}
```

The XHTML code follows:

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

HANDS-ON PRACTICE 3.8

In this Hands-On Practice you will code CSS properties to configure a centered page layout. We'll use the files from Hands-On Practice 3.7 as a starting point. Create a new folder called trillium2. Locate the Chapter3/3.7 folder in the student files. Copy the index.html, services.html, and trillium.css files to your trillium2 folder. Open the trillium.css file in a text editor. Create an id named `container`. Add the `margin-left`, `margin-right`, and `width` style properties to the style rules as follows:

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

Save the file.

Open the index.html file in a text editor. Add the XHTML code to configure a `<div>` assigned to the id `container` 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 one shown in Figure 3.17. The student files contain a sample solution in the Chapter3/3.8 folder.

3.10 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 slower than otherwise.

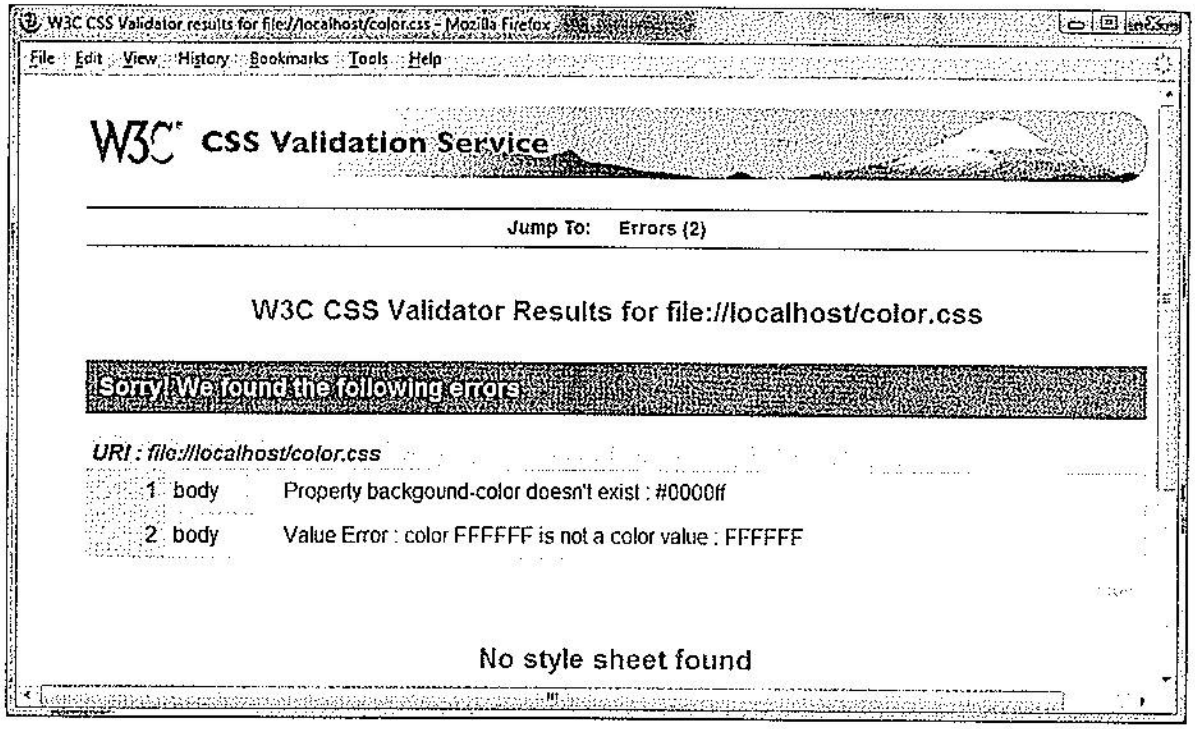
HANDS-ON PRACTICE 3.9

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 3.6 (student files Chapter3/color.css). Locate color.css and open it in Notepad.

We will add an error to the color.css file. Find the body selector style rule and delete the first "r" in the `background-color` property. Remove 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 3.18. Notice that two errors were found. The selector is listed followed by the reason an error was noted.

Figure 3.18
The validation results indicate errors



Notice that the first message 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 error. Test and revalidate your page. Your browser should now look similar to the one shown in Figure 3.19 and report only one error.

The error reminds you that `FFFFFF` is not a color value and expects you to already know that you need to add a "`#`" character to code a valid color value, `#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 3.20. 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's a good practice to validate your CSS style rules. The CSS validator can help you to identify code that needs to be corrected quickly and indicate which style rules a browser is likely to consider valid. Validating CSS is one of the many productivity techniques that Web developers commonly use.

Figure 3.19
The valid CSS is displayed below the errors (and warnings, if any)

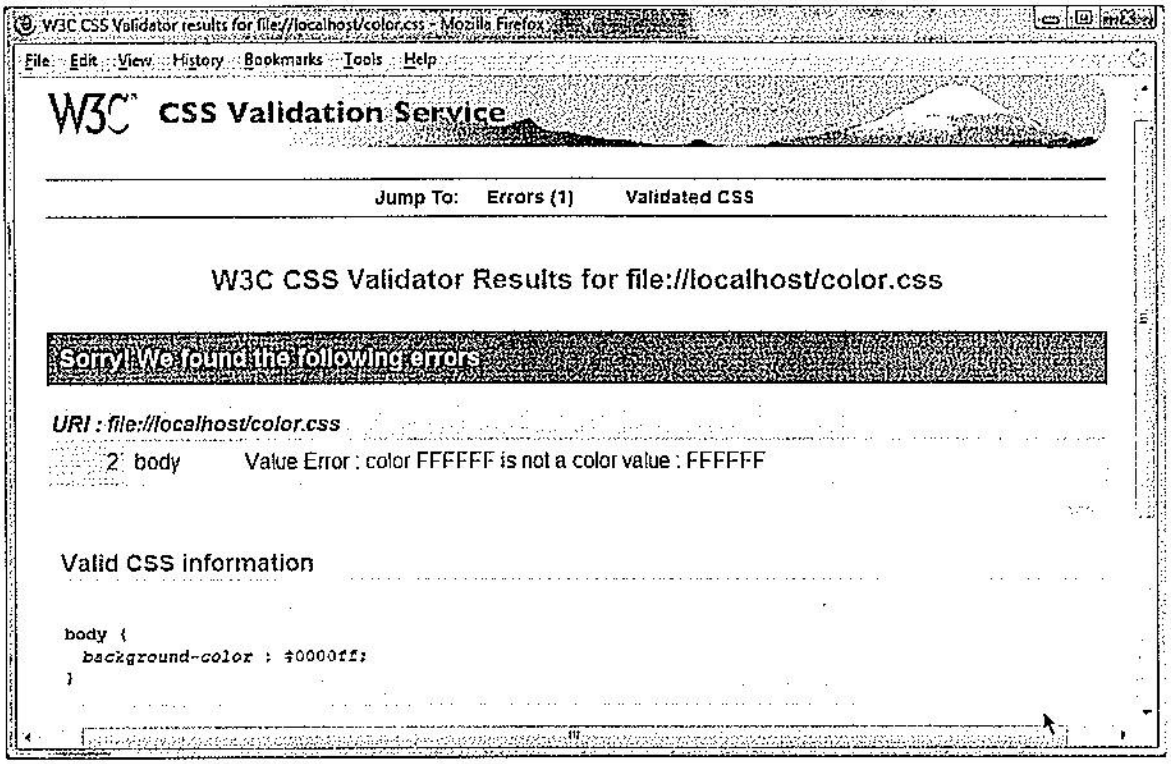
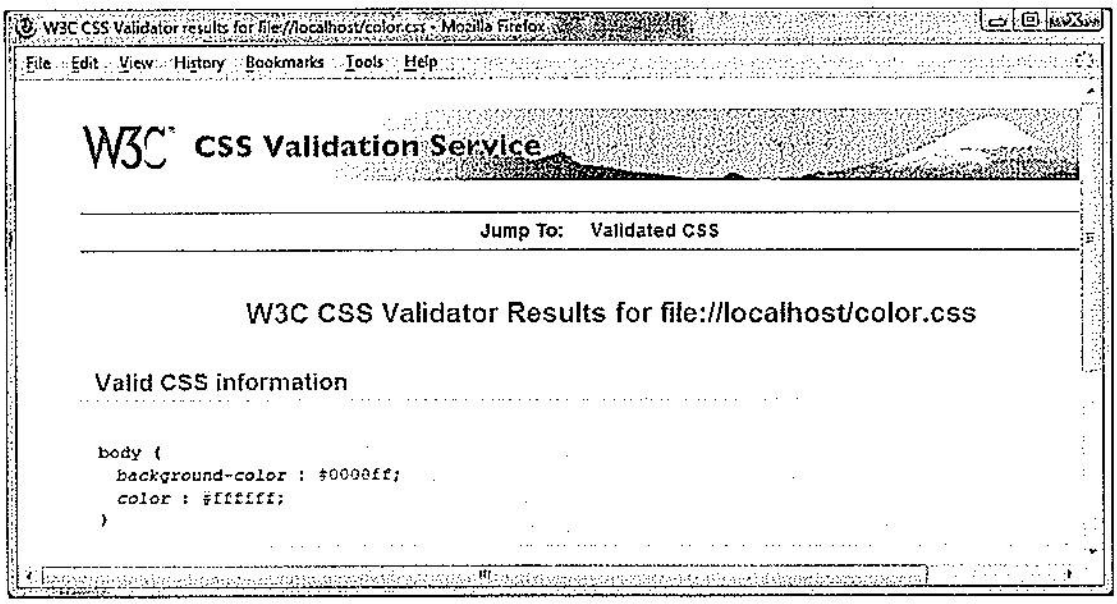
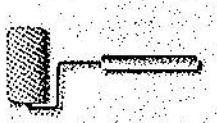


Figure 3.20
The CSS is valid!



CHAPTER SUMMARY



This chapter introduced Cascading Style Sheet rules associated with color and text on Web pages. There is much more that you can do with CSS: positioning, hiding and showing page areas, formatting margins, and formatting borders. As you continue your study of Web development with this textbook, you will study these additional uses. To learn more about CSS, check out the tutorials at http://echoecho.com/css.htm and http://www.mako4css.com, or visit the W3C site for official specifications.

Visit the textbook Web site at http://www.webdevfoundations.net for examples, the links listed in this chapter, and updated information.

Key Terms

`<div>`
`<link />`
`<span>`
`<style>`
`background-color` property
Cascading Style Sheets (CSS)
`class` attribute
`class` selector
`color` property
CSS validation
declaration
em unit
embedded styles
external styles
`font-family` property
`font-size` property
`font-style` property
`font-weight` property
hexadecimal color values
`id` attribute
`id` selector
imported styles
inline styles
`line-height` property
`margin-left` property
`margin-right` property
pixels
point
property
`rel` attribute
RGB color
rule
selector
`style` attribute
`text-align` property
`text-decoration` property
`type` attribute
Web Safe Color Palette
`width` property

Review Questions

Multiple Choice

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

2. 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

3. Which of the following can be a CSS selector?
   a. an XHTML element
   b. a class name
   c. an id name
   d. all of the above

4. Which of the following is the declaration property used to set the background color of a Web page?
   a. `bgcolor`
   b. `background-color`
   c. `color`
   d. none of the above

5. Which of the following do you configure to apply a style to a certain group of elements on a Web page?
   a. group
   b. id
   c. class
   d. none of the above

6. 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`

7. Which of the following is the file extension for an external style sheet?
   a. ess
   b. css
   c. htm
   d. no file extension is necessary

8. Which of the following is the element used to associate a Web page with an external style sheet?
   a. `<target>`
   b. `<a>`
   c. `<include>`
   d. `<link />`

9. Which of the following configures a background color of `#FFF8DC` for a Web page using CSS?
   a. `body { background-color: #FFF8DC; }`
   b. `document { background: #FFF8DC; }`
   c. `body { background: #FFF8DC'}`
   d. none of the above

10. Which of the following configures a class called special with red text, 24px, and Arial or a sans-serif font using CSS?

```
a. special {  color: red;
              font-size: 24px;
              font-family: Arial,
                sans-serif;}
b. .special { color: red;
              font-size: 24px;
              font-family: Arial,
                sans-serif;}
c. .special { text: red;
              font-size: 24px;
              font-family: Arial,
                sans-serif;}
d. .#special {text: red;
              font-size: 24px;
              font-family: Arial,
                sans-serif;}
```

Fill in the Blank

11. The _____ element is useful for creating logical areas on a Web page that are embedded within paragraphs or other block formatting elements.

12. CSS is a technology that is _____ supported by browsers.

13. The _____ CSS property can be used to center text on a Web page.

14. The _____ element is useful for creating areas on a Web page that are physically separated from other areas.

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 XHTML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Predict the Result</title>
<style type="text/css">
```

```
body {   background-color: #000066;
         color: #CCCCCC;
         font-family: Arial,sans-serif;
         font-size: 12px;
}
h1 {   background-color: #FFFFFF;
       color: #000066;
       font-size: 20px;
}
.footer {  font-size: 10px;
           font-style: italic;
}
</style>
</head>
<body>
  <div align="center">
    <h1>Trillium Media Design</h1>
    <br />
    <p>Home <a href="about.html">About</a>
      <a href="services.html">Services</a>
    </p>
  </div>
<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 well presented Web site that works.</p>
<p> </p>
<p> </p>
  <div align="center">
  <p class="footer">Contact <a
    href="mailto:web@trilliumtechnologies.com">
    web@trilliumtechnologies.com</a><br />
  Copyright &copy; 2008 Trillium Media Design</p>
  </div>
</body>
</html>
```

2. **Fill in the Missing Code.** This Web page should be configured so that the background and text colors have good contrast. The <h2> tag should use Arial. Some CSS properties and values, indicated by "_" (underscore), are missing. Some XHTML tags, indicated by <_>, are missing. Fill in the missing code.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Trillium Media Design</title>
<style type="text/css">
body {  background-color: #0066CC;
        color: "_";
}
```

```
h2 { "_": "_"
}
<_>
<_>
<body>
  <h2>Trillium Media Design</h2>
  <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 well presented Web site that works.
  </p>
</body>
</html>
```

3. **Find the Error.** Why won't this page display properly in a browser?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "~//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Trillium Media Design</title>
<style type="text/css">
body { background-color: #000066;
       color: #CCCCCC;
       font-family: Arial,sans-serif;
       font-size: 12px
}
<style>
</head>
<body>
  <h2>Trillium Media Design</h2>
  <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 well presented Web site that works.
  </p>
</body>
</html>
```

Hands-On Exercises

1. Write the XHTML for a large heading that uses inline styles to configure the background color of red and the text color of white.
2. Write the XHTML and CSS code for an embedded style sheet that configures a background color of white and a text color of green.
3. Write the CSS code for an external style sheet that configures the text to be brown, 14 px in size, and in Arial, Verdana, or a sans-serif font.
4. Write the XHTML and CSS code for an embedded style sheet that configures a class called new, that is bold and italic.

5. Write the XHTML and CSS code for an embedded style sheet that configures links without underlines; 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`). `<h1>` elements should 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, document text color of green. Hyperlinks should have a background color of white. `<h1>` elements should 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 Web site 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. Hand in printouts of format1.css, the moviecss1.html source code (print in Notepad), and the browser display of your moviecss1.html to your instructor.
   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! Hand in printouts of format2.css, the moviecss2.html source code (print in Notepad), and the browser display of your moviecss2.html.

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 Web site. Write the XHTML 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 use the external style sheet called ex3.css. Save the page as advantage.html.
   b. Create a Web page that includes an unordered list describing at least three disadvantages of utilizing Cascading Style Sheets. The text `CSS Disadvantages` should be contained within `<h1>` tags. This page should include a hyperlink to the W3C Web site. Write the XHTML code so that one of the disadvantages is

configured to be a class called news. Place an e-mail link to yourself on the Web page. The Web page should use the external style sheet called ex3.css. Save the page as disadvantage.html.

c. Create an external style sheet (call it ex3.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.

d. Launch a browser and test your work. Display the advantage.html page. It should use the formatting configured in ex3.css. Modify the Web page and/or the css file until your page displays as requested. Display the disadvantage.html page. It should also use the formatting configured in the ex3.css file. Create printouts of ex3.css, advantage.html, disadvantage.html source code (print in Notepad), the browser display of advantage.html, and the browser display of disadvantage.html. Label these printouts Exercise 8d.

e. Change the configuration of the external style sheet (ex3.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 and disadvantage.html pages. Notice how they each pick up the styles from the external style sheet. Create printouts of the advantage.html and disadvantage.html browser display and label them Exercise 8e.

f. 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. Print the browser display of advantage.html and label it Exercise 8f.

9. **Practice Validating CSS.** Choose a CSS external style sheet file to validate—perhaps you have created one for your own Web site. Otherwise, use an external style sheet file that you worked with in this chapter. Use the W3C free CSS validator. Visit 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. Answer the following questions. Was it 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 this to other students? Why or why not?

Web Research

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

   - http://www.w3.org/Style/CSS/
   - http://positioniseverything.net
   - http://www.dezwozhere.com/links.html
   - http://www.westciv.com/style_master/academy/browser_support

Create a Web page that provides a list of at least five CSS resources on the Web. For each CSS resource provide the URL, Web site name, and a brief description. Your Web page should contain a table and use color. Place your name in the e-mail address at the bottom of the Web page. Print the source code (from Notepad) and the browser view of your 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.mako4css.com
   - http://www.htmlgoodies.com/beyond/css.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 Web site, and a description of the new technique you discovered. Place your name in the e-mail address at the bottom of the Web page. Print the external style sheet (if you used one), the Web page source code (from Notepad), and the browser view of your Web page.

Focus on Web Design

In this chapter you 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://coe.sdsu.edu/eet/Articles/wadecolor/start.htm
- http://iit.bloomu.edu/vthc/Design/psychology.htm
- http://www.my-photoshop.com/bydesign/id-tutorials/color-psychology.html

Color Theory

- http://www.colormatters.com/colortheory.html
- http://colortheory.liquisoft.com/
- http://www.digital-web.com/articles/color_theory_for_the_colorblind/

Color Scheme Generators

- http://meyerweb.com/eric/tools/color-blend
- http://colorschemer.com/schemes/
- http://www.colr.org
- http://colorsontheweb.com/colorwizard.asp
- http://www.steeldolphin.com/color_scheme.html
- http://wellstyled.com/tools/colorscheme2/index-en.html

You have the following tasks:

a. Design a color scheme. List three hexadecimal color values (in addition to 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 Web site would they be appropriate for? List the URLs of any resources you used.
c. Create an external CSS file name color1.css that configures font properties, text color, and background color selections for the document, `h1` selector, `p` 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.

Open your files in Notepad and print the source code for color1.css and color1.html. Display your page in a browser; print the page. Hand in all printouts to your instructor.

WEB SITE CASE STUDY: Implementing CSS

Each of the following case studies continues throughout most of the text. This chapter implements CSS in the Web sites.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House Case Study. Figure 2.26 shows a site map for the JavaJam Web site. The Home page and Menu page were created in Chapter 2. You will use the existing Web site as a start while you create a new version that uses an external style sheet to configure text and color.

You have the following tasks:

1. Create an external style sheet named javajam.css that configures the color and text for the JavaJam Web site.
2. 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 3.21 (also shown in the color insert section).
3. Modify the Menu page to be consistent with the new Home page.
4. Configure centered page layout.

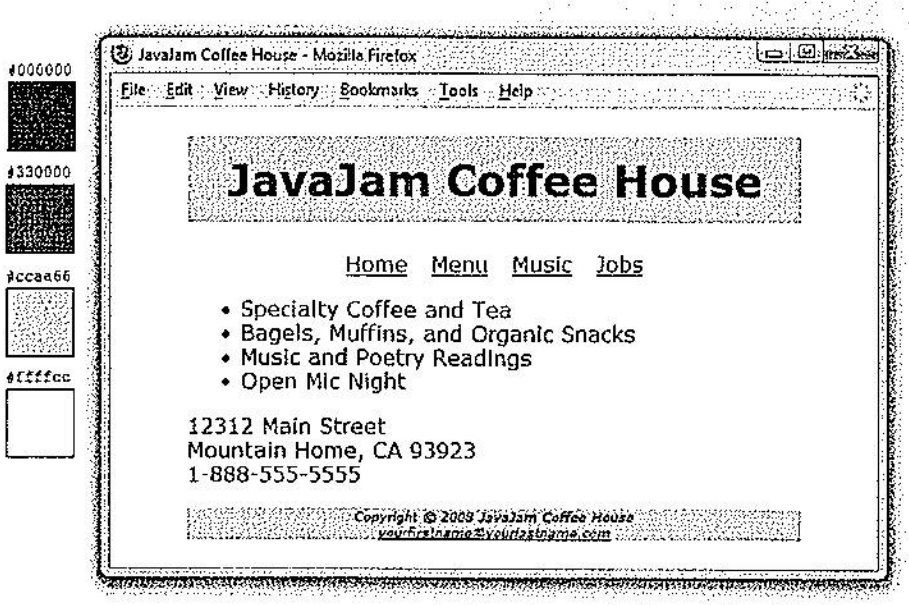


Figure 3.21
New JavaJam index.html

See the center color insert

Hands-On Practice Case

Create a folder called javajamcss. Copy all the files from your javajam folder into the javajamcss folder.

1. **The External Style Sheet.** Launch Notepad. You will create an external style sheet named javajam.css. Code the CSS to configure the following:
   - Global styles for the document background color (`#ffffcc`), text color (`#330000`), and Verdana, Arial, or any sans-serif font
   - Style rules for the `h1` selector that configure background color (`#ccaa66`), text color (`#000000`), 200% line height, and centered text
   - Style rules for the centered navigation area (use an id named nav)
   - Style rules for the page footer area (use an id named `footer`) for background color (`#ccaa66`), text color (`#000000`), small font size (.60em), italics, and centered

   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.
2. **The Home Page.** Launch Notepad and open the index.html file. You will modify this file to apply styles from the javajam.css external style sheet as follows:
   - Add a `<link />` element to associate the Web page with the javajam.css external style sheet file. Save and test your index.html page in a browser and you'll notice that the styles configured with the body and `h1` selectors are already applied!
   - Configure the navigation area. Since the navigation is not semantically a "paragraph" (a collection of sentences about a central topic), replace the `<p>` element with a `<div>` element. Assign this `<div>` to the id named `nav`.
   - Configure the page footer area. Replace the `<p>` elements with `<div>` elements. Remove the `<small>` and `<em>` elements because the `font-size` and `font-style` are configured as part of the `footer` id. Assign this `<div>` to the id named `footer`.

- Save the index.html file and test in a browser. Your page should look similar to the one shown in Figure 3.21 except that your page content will be left-aligned instead of centered. Don't worry—you'll center your page layout in Step 4 of this case study.

3. **The Menu Page.** Launch Notepad and open the menu.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 menu.html page. It should look similar to the one shown in Figure 3.22 except for the alignment.

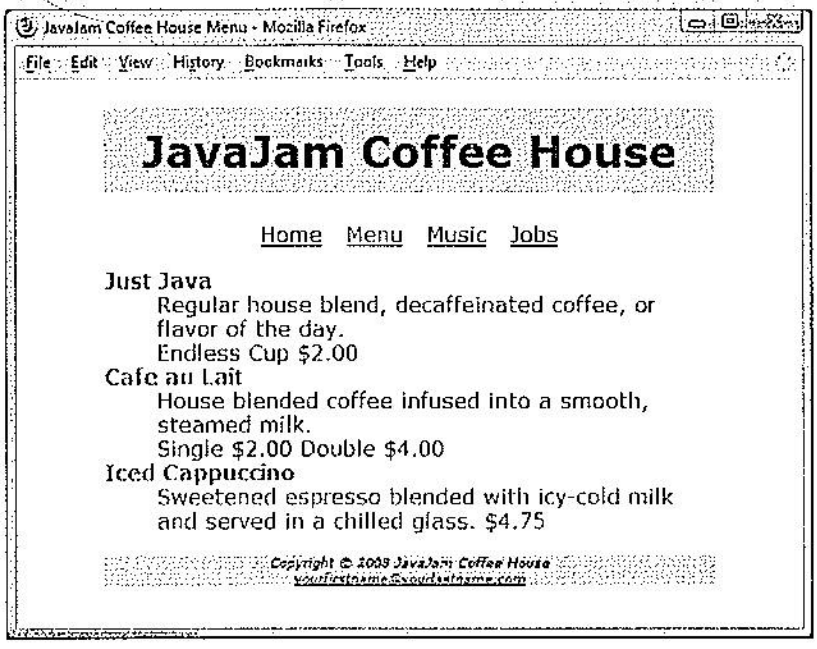


Figure 3.22
New menu.html page

4. **Centered Page Layout with CSS.** Modify javajam.css, index.html, and menu.html to configure page content that is centered with width set to 80%. Refer to Hands-On Practice 3.8 if necessary. Save your files and retest your pages. The index.html and menu.html pages should closely match the samples shown in Figures 3.21 and 3.22.

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?

Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital Case Study. Figure 2.30 shows a site map for the Fish Creek Web site. The Home page and Services page were created in Chapter 2. You will use the existing Web site as a start while you create a new version that uses an external style sheet to configure text and color.

You have the following tasks:

1. Create an external style sheet named fishcreek.css that configures the color and text for the Fish Creek Web site.

2. Modify the Home page to use an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.23 (shown also in the color insert section).

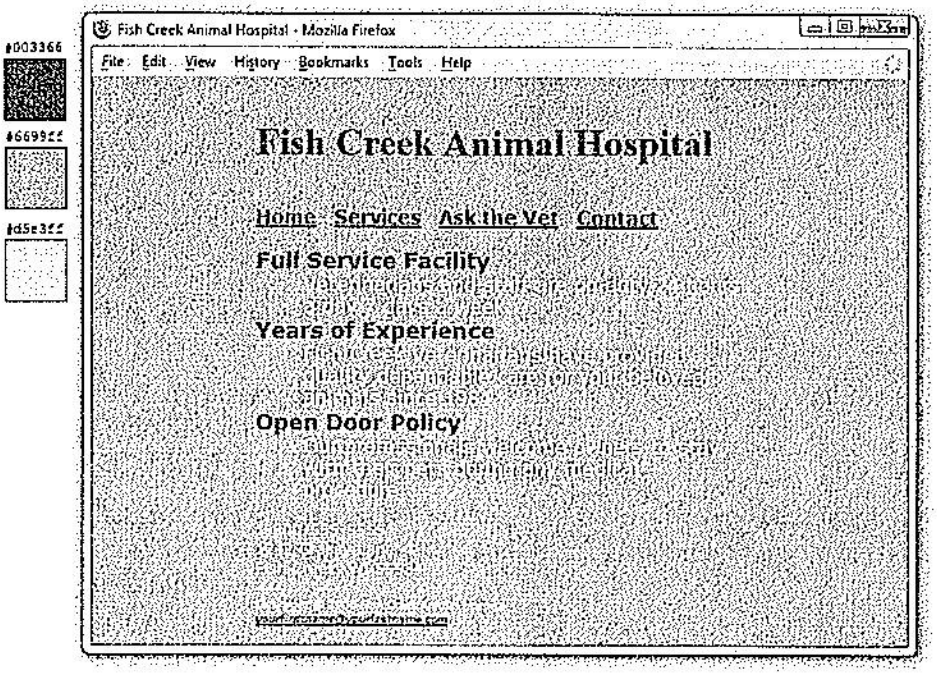


Figure 3.23
New Fish Creek index.html

See the center color insert

3. Modify the Services page to be consistent with the new Home page. Configure centered page layout.

Hands-On Practice Case

Create a folder called fishcreekcss. Copy all the files from your fishcreek folder into the fishcreekcss folder.

1. **The External Style Sheet.** Launch Notepad. You will create an external style sheet named fishcreek.css. Code the CSS to configure the following:
   - Global styles for the document background color (`#6699ff`), text color (`#d5e3ff`), and Verdana, Arial, or any sans-serif font
   - Style rules for the `h1` selector that configure background color (`#6699ff`), text color (`#003366`), and 200% line height
   - Style rules for a navigation area (use an id named `nav`) that displays text in bold
   - Style rules for a class named category with bold font, background-color (`#6699ff`), text color (`#003366`), and larger font size (1.1em)
   - Style rules for the page footer area (use an id named `footer`) 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.
2. **The Home Page.** Launch Notepad and open the index.html file. You will modify this file to apply styles from the fishcreek.css external style sheet.

- Add a `<link />` element to associate the Web page with the fishcreek.css external style sheet file. Save and test your index.html page in a browser and you'll notice that the styles configured with the `body` and `h1` selectors are already applied!
- Configure the navigation area. Since the navigation is not semantically a "paragraph" (a collection of sentences about a central topic), replace the `<p>` element with a `<div>` element. Assign this `<div>` to the id named `nav`. Remove the `<strong>` element from this area.
- Configure each `<dt>` element to apply the `category` class. Remove the `<strong>` elements from this area.
- Configure the page footer area. Replace the `<p>` elements with `<div>` elements. Assign this `<div>` to the id named `footer`. Remove the `<small>` and `<em>` elements because the `font-size` and `font-style` are configured as part of the `footer` id.
- Save the index.html file and test in a browser. Your page should look similar to the one shown in Figure 3.23 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 Step 4 of this case study.

3. **The Services Page.** Launch Notepad and open the services.html file. You will modify this file in a similar manner—add the `<link />` element, configure the navigation area, configure the `category` classes (use `<span>` elements), and configure the page footer area. Save and test your new services.html page. It should look similar to the one shown in Figure 3.24 except for the alignment.

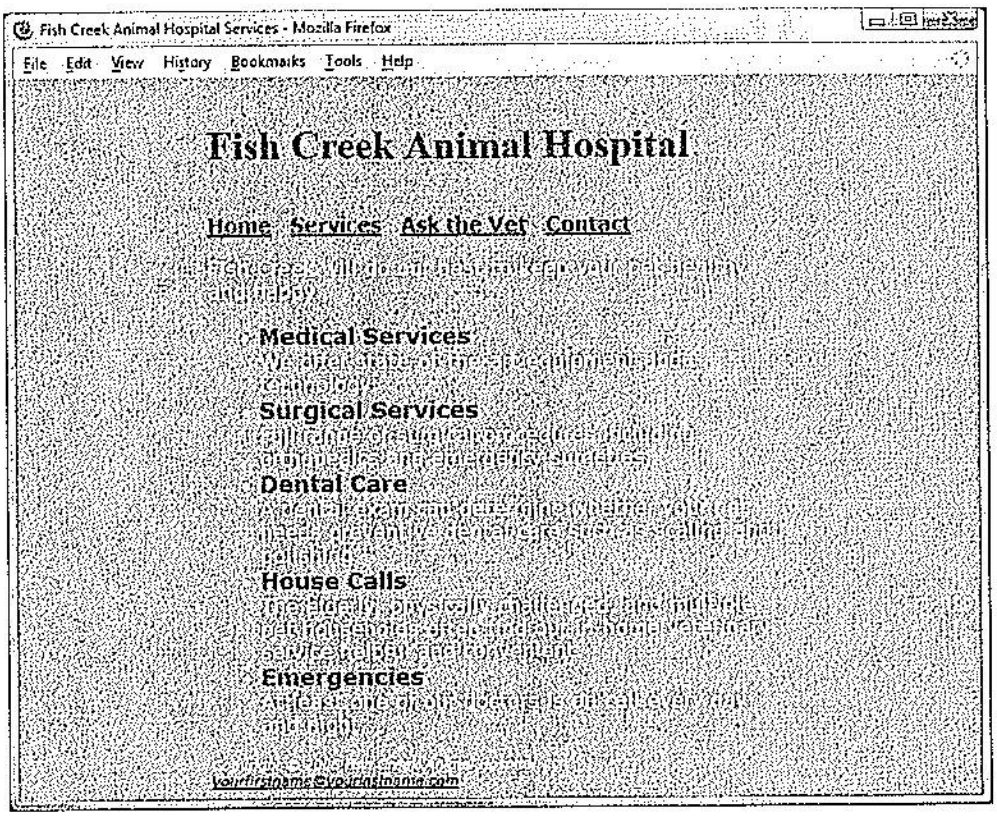


Figure 3.24
New services.html page

4. **Centered Page Layout with CSS.** Modify fishcreek.css, index.html, and services.html to configure page content that is centered with width set to 80%. Refer to Hands-On Practice 3.8 if necessary. Save your file and retest your pages. The index.html and services.html pages should closely match the samples shown in Figures 3.23 and 3.24.

Experiment with modifying the fishcreek.css file. Change the page background color, the font family, font color, and so on. Test your pages in a browser. Notice that multiple pages display differently because they link to the single file (fishcreek.css) that configures their formatting.

Pete the Painter

See Chapter 2 for an introduction to the Pete the Painter Case Study. Figure 2.34 shows a site map for the Pete the Painter Web site. The Home page and Services page were created in Chapter 2. You will use the existing Web site as a start while you create a new version of this Web site that uses an external style sheet to configure text and color.

You have the following tasks:

1. Create an external style sheet named painter.css that configures the color and text for the Pete the Painter Web site.
2. 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 3.25 (shown also in the color insert section).

Figure 3.25
New Pete the Painter index.html

See the center color insert

3. Modify the Services page to be consistent with the new Home page.

Hands-On Practice Case

Create a folder called paintercss. Copy all the files from your painter folder into the paintercss folder.

1. **The External Style Sheet.** Launch Notepad. You will create an external style sheet named painter.css. Code the CSS to configure the following:
   - Global styles for the document background color (`#ffffff`), text color (`#000000`), and Verdana, Arial, or any sans-serif font
   - Style rules for the logo class that configure background color (`#ffffff`), text color (`#336633`), and Georgia, Times New Roman, or any serif font
   - Style rules for a navigation area (use an id named `nav`) that displays text in bold
   - Style rules for a class named category with a bold font, background-color (`#ffffff`), text color (`#336633`), and a larger font size (1.2em)
   - Style rules for the page footer area (use an id named `footer`) with a small font size (.60em) and italic text

   Save the file as painter.css in the paintercss folder. Check your syntax with the CSS validator (http://jigsaw.w3.org/css-validator). Correct and retest if necessary.
2. **The Home Page.** Launch Notepad and open the index.html file. You will modify this file to apply styles from the painter.css external style sheet.
   - Add a `<link />` element to associate the Web page with the painter.css external style sheet file. Save and test your index.html page in a browser and you'll notice that the styles configured with the `body` selector are already applied!
   - Configure the logo area. Code a `<div>` element that contains the `<h1>` and `<h3>` elements in the logo area. Assign the `<div>` to the logo class.
   - Configure the navigation area. Since the navigation is not semantically a "paragraph" (a collection of sentences about a central topic), replace the `<p>` element with a `<div>` element. Assign this `<div>` to the id named `nav`. Remove the `<strong>` element from this area.
   - Configure the `<ul>` to apply the `category` class.
   - Configure the page footer area. Replace the `<p>` elements with `<div>` elements. Assign this `<div>` to the id named `footer`. Remove the `<small>` and `<em>` elements because the `font-size` and `font-style` are configured as part of the `footer` id.
   - Save the index.html file and test in a browser. Your page should look similar to Figure 3.25.
3. **The Services Page.** Launch Notepad and open the services.html file. You will modify this file in a similar manner— add the `<link />` element, configure the logo area, configure the navigation area, configure the `category` classes (use `<span>` elements and remove the `<strong>` element from this area), and configure the page footer area. Save and test your new services.html page. It should look similar to the one shown in Figure 3.26.

Experiment with modifying the painter.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.

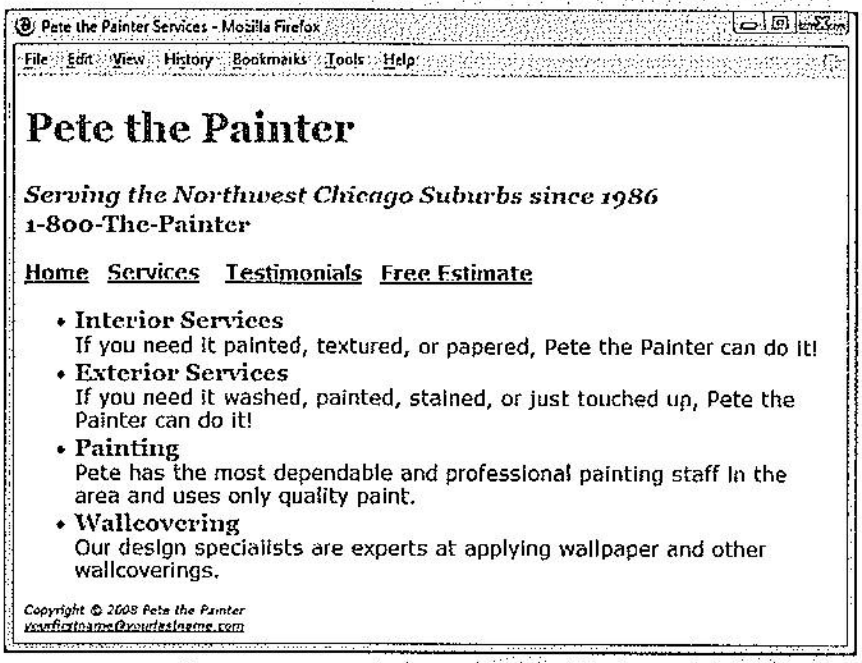


Figure 3.26
New services.html page

Prime Properties

See Chapter 2 for an introduction to the Prime Properties Case Study. Figure 2.38 shows a site map for the Prime Properties Web site. The Home page and Financing page were created in Chapter 2. You will use the existing Web site as a start while you create a new version of this Web site that uses an external style sheet to configure text and color.

You have the following tasks:

1. Create an external style sheet named prime.css that configures the color and text for the Prime Properties Web site.
2. Modify the Home page to use an external style sheet to configure colors and fonts. The new Home page and color swatches are shown in Figure 3.27 (shown also in the Color Insert Section).
3. Modify the Financing page to be consistent with the new Home page.

Hands-On Practice Case

Create a folder called primecss. Copy all the files from your prime folder into the primecss folder.

1. **The External Style Sheet.** Launch Notepad. You will create an external style sheet named painter.css. Code the CSS to configure the following:
   - Global styles for the document background color (`#ffffcc`), text color (`#003300`), and Arial, Helvetica, or any sans-serif font
   - Style rules for the `h2` selector that configure background color (`#ffffcc`) and text color (`#003366`)
   - Style rules for the `h4` selector that configure background color (`#ffffcc`) and text color (`#006600`)

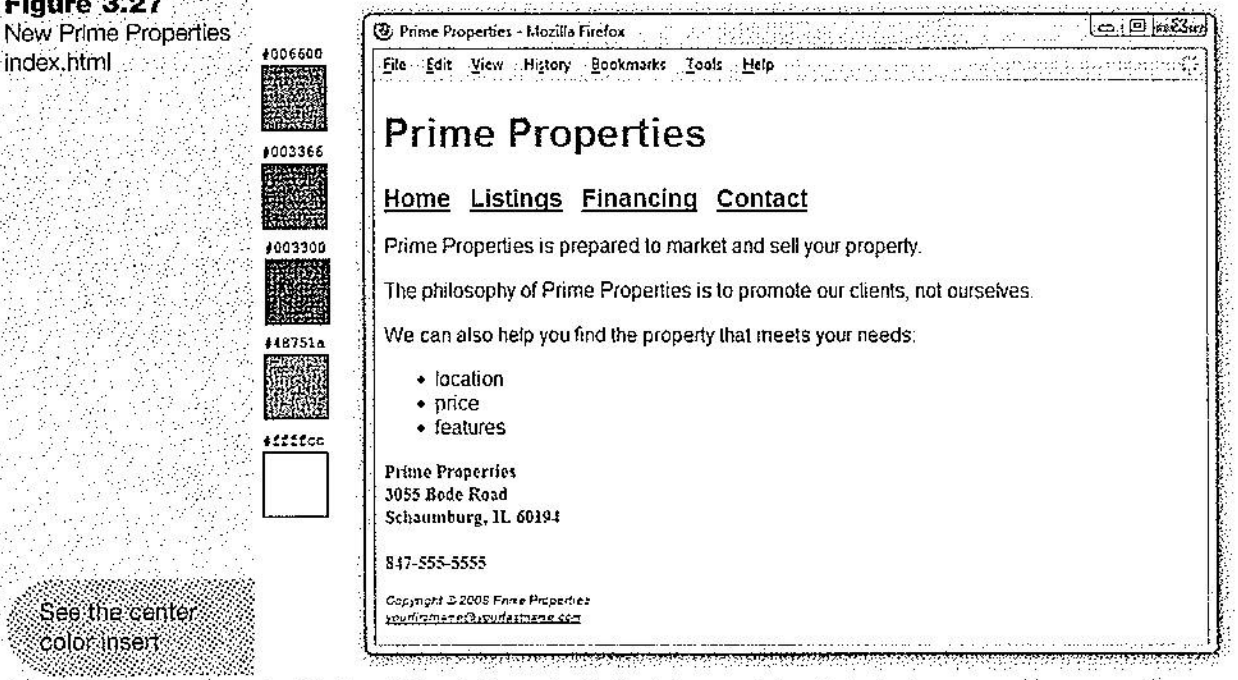


Figure 3.27
New Prime Properties index.html

See the center color insert

- Style rules for the `dd` selector that configure italic, smaller than the default (.90em), with 200% line height
- Style rules for the `logo` class that configure background color (`#ffffcc`) and text color (`#48751A`)
- Style rules for a navigation area (use an id named `nav`) that displays text in bold, large (1.2em) font
- Style rules for a class named `contact` with bold, smaller than the default (.90em) using the Times New Roman or any serif font
- Style rules for the page footer area (use an id named `footer`) with small font size (.60em) and italic text

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

2. **The Home Page.** Launch Notepad and open the index.html file. You will modify this file to apply styles from the prime.css external style sheet.
   - Add a `<link />` element to associate the Web page with the painter.css external style sheet file. Save and test your index.html page in a browser and you'll notice that the styles configured with the `body` selector are already applied!
   - Configure the logo area. Assign the `<h1>` element to the class named `logo`.
   - Configure the navigation area. Since the navigation is not semantically a "paragraph" (a collection of sentences about a central topic), replace the `<p>` element with a `<div>` element. Assign this `<div>` to the `id` named `nav`. Remove the `<strong>` element from this area.
   - Configure the paragraph containing the address and phone information. Assign this area to the class named `contact`. Remove the `<small>` element from this area.

- Configure the page footer area. Replace the `<p>` elements with `<div>` elements. Assign this `<div>` to the `id` named `footer`. Remove the `<small>` and `<em>` elements because the `font-size` and `font-style` are configured as part of the `footer id`.
- Save the index.html file and test in a browser. Your page should look similar to the one shown in Figure 3.27.

3. **The Financing Page.** Launch Notepad and open the financing.html file. You will modify this file in a similar manner—add the `<link />` element, configure the logo area, configure the navigation area, and configure the page footer area. Save and test your new financing.html page. It should look similar to the one shown in Figure 3.28.

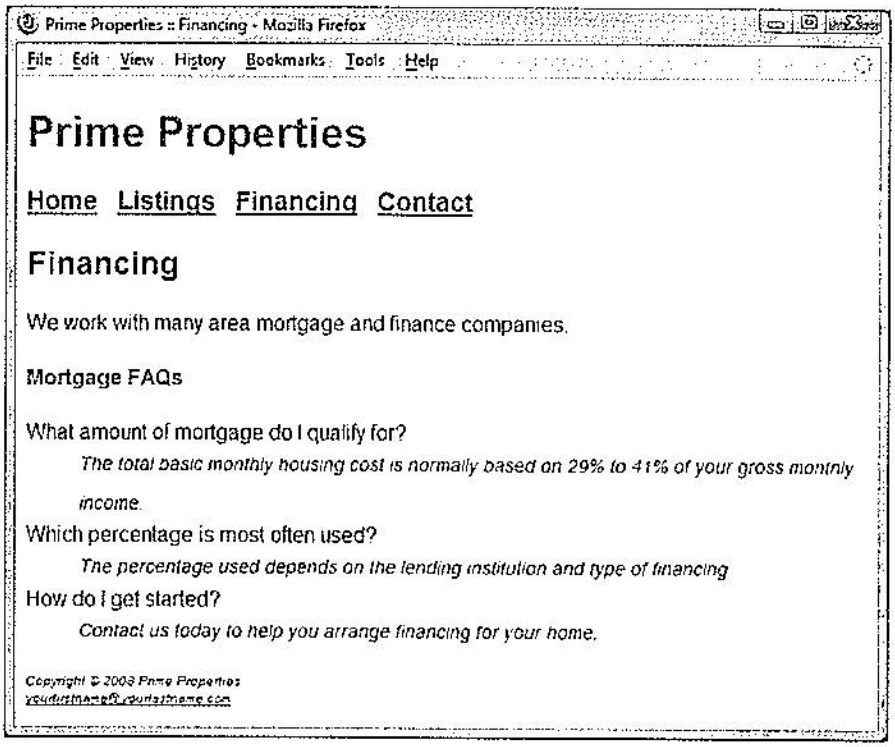


Figure 3.28 New financing.html page

Experiment with modifying the prime.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.