chapter

7 Styling Content with Cascading Style Sheets

USING HTML, DEVELOPERS LAY out content within a webpage, specifying headers, paragraphs, tables, images, and more. For years, developers would say that HTML "formats a webpage." It's preferable to say that HTML specifies the webpage elements and layout, and cascading style sheet attributes specify the actual formatting. This chapter takes a detailed look at cascading style sheets, or CSS. In Chapter 8, you will continue to drill down into additional formatting capabilities. That said, you have actually been using CSS attributes throughout this book's first six chapters! You have used attributes to specify background colors, text alignment, image appearance, and more.

Learning Objectives

This chapter examines how to use cascading style sheets to format webpages. By the time you finish this chapter, you will understand the following key concepts:

- How to use the **style** attribute within an HTML tag to apply an inline style
- How to use the **<style>** and **</style>** tag pair to define embedded style definitions
- How to create an external style sheet file and use the **<link>** tag to include it within an HTML file
- Why the word "cascading" is used in the term "cascading style sheets"

APPLYING INLINE STYLES

Throughout this book, you have made use of tag attributes to control an element's appearance. For example, the following **<h1>** tag uses the **color** property (as defined below) to display header text in red:
<h1 style="color:red">This text is red
A **style** is a collection of formatting attributes applied to an HTML tag. Using a style for text, a developer may specify a font family, font size, text color, text alignment, and so on. Developers refer to attribute settings that appear within an HTML tag as **inline styles**. To specify inline styles, place the **style** attribute within an HTML tag, and within the **style** attribute, assign values to the desired formatting properties. With a simple HTML page, using inline styles is fine. As you will learn, as the number of pages in your website increases, the use of inline styles often leads to redundancy of effort and pages that are hard to modify.

Revisiting Inline Styles

Applying inline styles is easy. The only challenge is in knowing which **style** attributes exist. To help you get started, the W3Schools HTML tutorial website provides a complete list of CSS properties at *http://www.w3schools.com/cssref/default.asp*.
The sections that follow will help you better understand formatting performed by assigning property values to element attributes.
A **property** is an attribute value within a style definition, such as **background-color** or **font-family**, to which a developer can assign a specific value.
Getting a feel for the various attributes is important because, later in this chapter, you will use the same attributes to create embedded styles as well as external CSS files. Developers use **embedded styles** to apply formatting to an entire HTML page. Developers use **external styles** to apply formatting to an entire website.

Applying Inline Styles to the <body> Tag

As you have learned, the **<body>** and **</body>** tag pair groups your entire page's HTML tags. Using inline CSS styles within the **<body>** tag, you can assign a color or image to the page background. You also can specify default settings for fonts, font colors, and so on, which the browser uses throughout the page as it displays the content.
The following HTML file, YellowBody.html, uses the **background-color** property to set the page background to yellow:

```
<!DOCTYPE html>
```

```
<html>
<body style="background-color:yellow">
<h1>Yellow Background</h1>
<hr/>
</body>
</html>
```
As you can see, to specify the inline style, you specify the **style** attribute within the tag followed by the specific property settings within quotes.
In a similar way, the following HTML file, BodyBackgroundImage.html, uses the **background-image**property to direct that the browser repeat a photo to fill the page background:
```
<!DOCTYPE html>
<html>
<body style="background-image:
url('http://www.websitedevelopmentbook.com/chapter07/puppy.jpg')">
</body>
</html>
```
When you view the file's contents, your browser will display the contents shown in FIGURE 7-1.

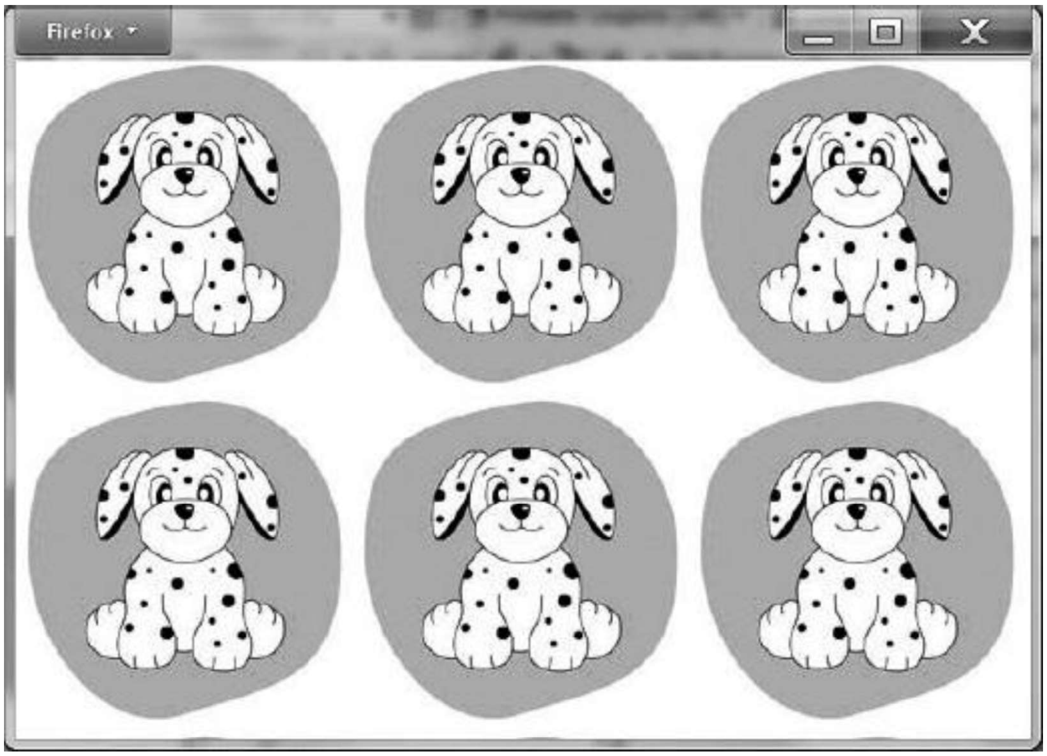


FIGURE 7-1 Using an inline CSS style to specify a page background image

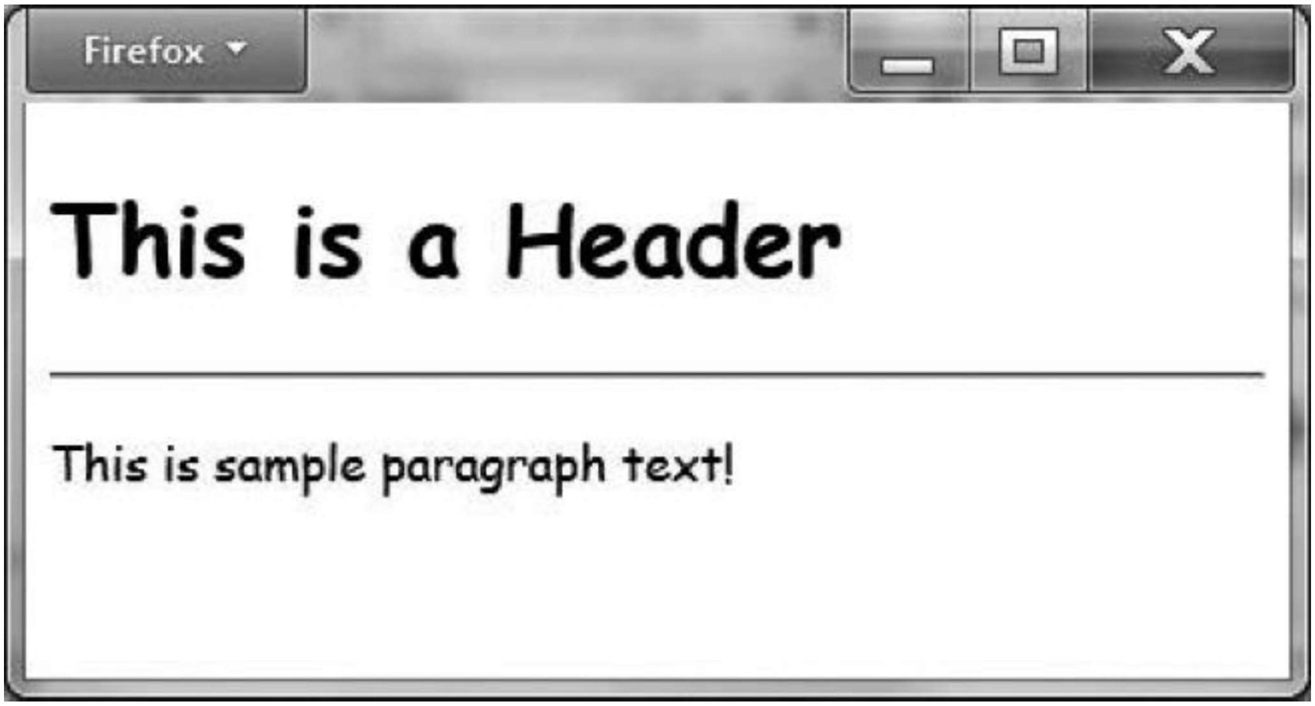
Credit: © sbego/Shutterstock

When you apply CSS property settings to an HTML tag that encloses other tags, the styles you select also apply to the internal tags. For example, the following HTML file, BodyFont.html, uses CSS inline styles to select the Comic Sans MS font for the <**body**> tag. Because the **<body>** tag encloses the other tags within your file, the font becomes the default for text within nested tags:

```
<!DOCTYPE html>
<html>
<body style="font-family:'Comic Sans MS'">
<h1>This is a Header</h1>
<hr/>
<p>This is sample paragraph text!</p>
</body>
</html>
```

When you view the file's contents, the browser displays the contents shown in **FIGURE 7-2**.

Again, by applying the **font-family** property to the **<body>** tag, you can set the default font for all elements. Those elements, in turn, can override the font setting by also using the **font-family** property within an inline style. For example, the following HTML file, FontOverride.html, sets the default font for the page within the **<body>** tag and then later, a paragraph tag uses an inline style to override the font setting for its text content:



This is a Header

This is sample paragraph text!

FIGURE 7-2 Using the **font-family** style attribute to specify the default font

```
<!DOCTYPE html>
<html>
<body style="font-family:'Comic Sans MS'">
<h1>This is a Header</h1>
<hr/>
<p style="font-family:Arial">This is sample paragraph text!</p>
</body>
</html>
```

As you examine HTML tags, you should note the various CSS properties they support. Again, the W3Schools reference is a great resource.

The following HTML file, ImageStack.html, uses inline styles to specify the x, y, and z coordinates to position images on a page:

```
<!DOCTYPE html>
<html>
<body>
<img
src="http://www.websitedevelopmentbook.com/Chapter07/cigar.jpg"
style="position:absolute; left:400px; top:300px; z-index:1;" />
<img
src="http://www.websitedevelopmentbook.com/Chapter07/wine.jpg"
style="position:absolute; z-index:0;" />
```

```
<img
src="http://www.websitedevelopmentbook.com/Chapter07/pizza.jpg"
style="position:absolute; left:500px; top:0px; z-index:2;" />
</body>
</html>
```

As you can see, each image specifies **absolute** positioning and then specifies a left-edge position, top-edge position, and a z-index. Using the z-index value, the page stacks images on top of one another. When you view the file's contents, your browser will display the contents shown in **Figure 7-3**.

As you work with inline styles to format your page, you may find that you begin to repeat the same style definitions throughout the file. For example, the following HTML file, RepeatStyles.html, has three headings, three paragraphs, and three images. The file applies the same styles to each element, meaning each **<h1>** tag uses the same attributes, each **<p>** tag uses the same attributes, and each **<img>** tag uses the same attributes:

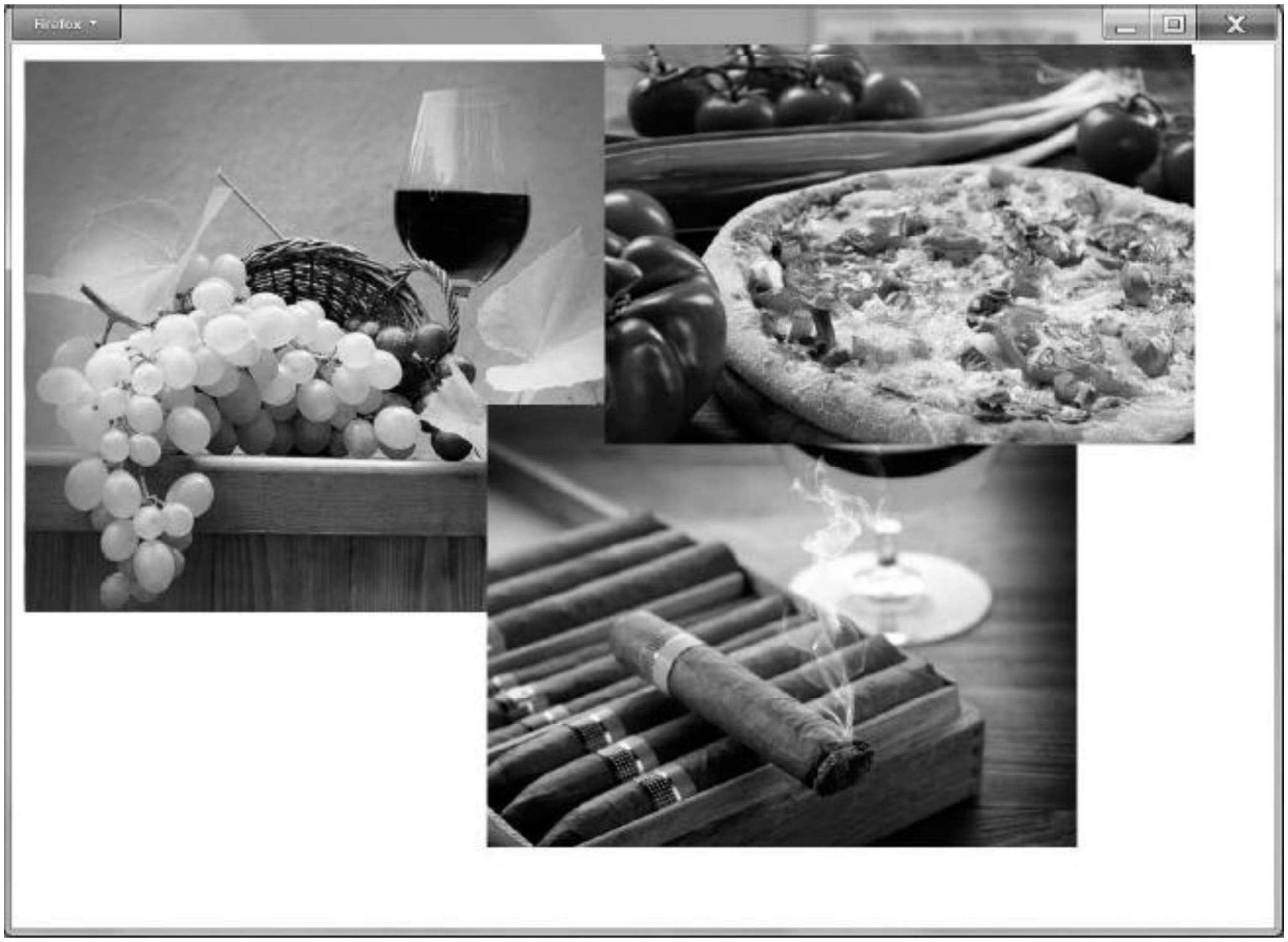


FIGURE 7-3 Using CSS styles to specify image coordinates

Credit (clockwise from upper left): © bitt24/Shutterstock; © Vima/Shutterstock; © marco mayer/Shutterstock

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:red; font-family:arial;">Dogs</h1>
<p style="color:blue; font-decoration:italic; text-align:left;">Dogs are
great!</p>
<img style="border:0;"
src="http://www.websitedevelopmentbook.com/chapter07/dogs.jpg"/>
<h1 style="color:red; font-family:arial;">Cats</h1>
<p style="color:blue; font-decoration:italic; text-align:left;">Cats are
great!</p>
;<img style="border:0;"
src="http://www.websitedevelopmentbook.com/chapter07/cats.jpg"/>
<h1 style="color:red; font-family:arial;">Horses</h1>
<p style="color:blue; font-decoration:italic; text-align:left;">Horses are
great!</p>
<img style="border:0;"
src="http://www.websitedevelopmentbook.com/chapter07/horses.jpg"/>
</body>
</html>
```

When you view the file's contents, your browser will display the content shown in **FIGURE 7-4**.

As you may start to guess, if your website has multiple pages of similar content, repeating the style definitions for each tag is time-consuming at best. Worse yet, if you decide to change an element's look and feel on your page, it requires changing every style definition throughout your files, which not only takes time, but is prone to errors.

When you need to apply the same format to each occurrence of an HTML tag throughout a page, embedded CSS styles provide the solution.

DEFINING EMBEDDED STYLE DEFINITIONS

An inline style is so named because it appears "inline" within an HTML tag. The inline style applies only to the tag within which it appears. In contrast, an embedded style, which you will define using

the **<style>** and **</style>** tag pair, will, by default, apply to all occurrences of a tag.
To define an embedded style, you place the **<style>** and **</style>** tag pair inside the **<head>** and **</head>** tag pair at the top of your file. Within the **<style>** and **</style>** tags, you will define your tag properties:

```
<head>
<html>
<style type="text/css">
/* Style Attributes Here */
</style>
</head>
```

To define styles for a specific HTML tag, specify the tag selector, which is the letter or letters that appear between the left and right brackets, followed by left and right braces that contain the property settings you desire. The following style definition, for example, directs the browser to set the text color for the **<h1>** tag to blue:

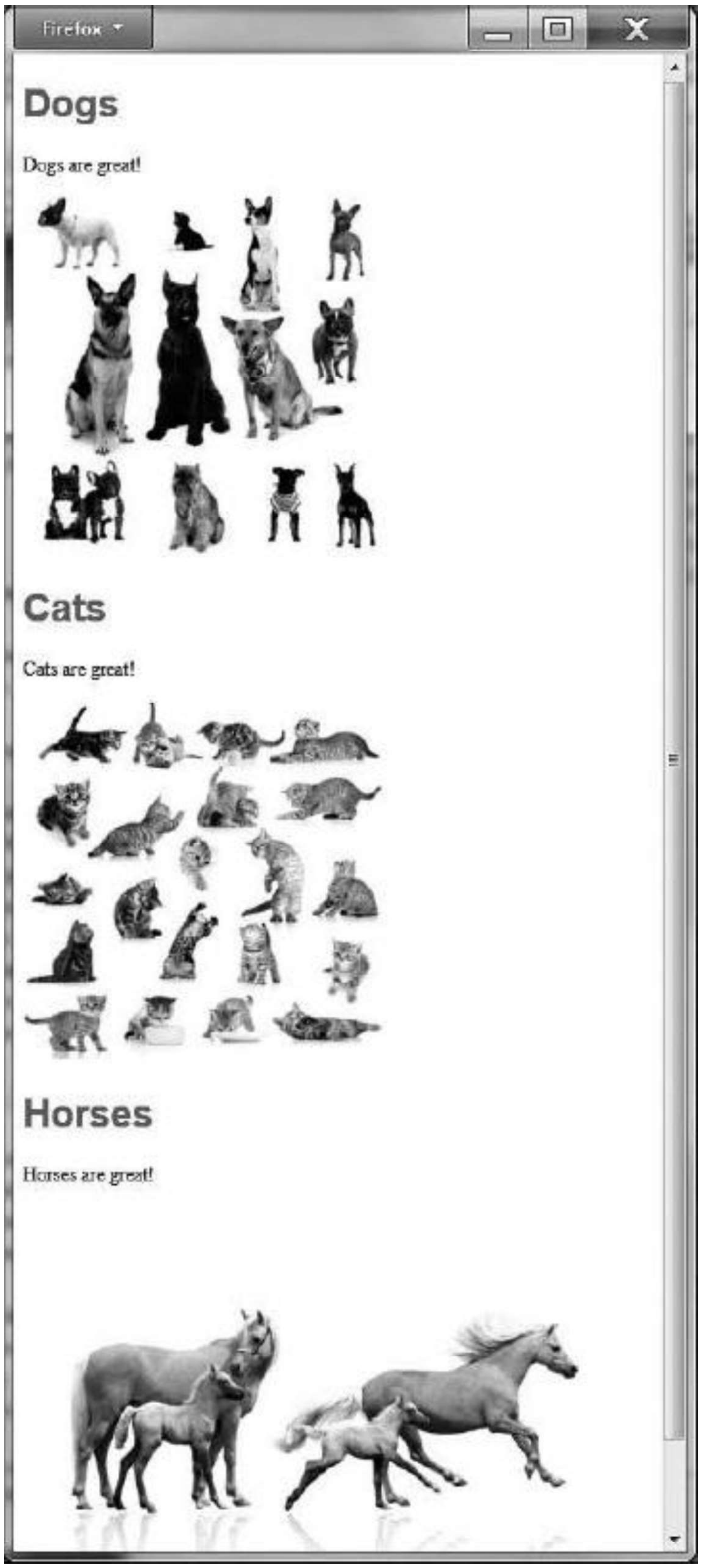


FIGURE 7-4 Repeating CSS property values throughout an HTML file

Credit (from top): © Marina Jay/Shutterstock; © Oksana Kuzmina/Shutterstock; © mariait/Shutterstock

```
<head>
<style type="text/css">
h1 { color:blue; }
</style>
</head>
<body>
```

Within the **<style>** tag, use the **type** attribute to tell the browser that you are specifying text values for CSS styles. As you can see, to specify a tag, you do not include the brackets, which normally enclose the tag name. Instead, you just specify, in this case, **h1** for the **<h1>** tag. Developers refer to the tag name without the brackets as a *selector*. You place the desired style property value within left and right curly braces.

Often, when assigning attributes to a tag selector, you specify values for multiple properties. To do so, separate the property settings with a semicolon, as shown here:

```
h1 { color:red; font-decoration:italic; text-align:left; }
```

In this case, the style definition selects a red, italic font for the **<h1>** tag that uses left-justified text.

In addition, you will often define the properties for multiple elements at the same time, as shown here:

```
<style type="text/css">
h1 { color:red; font-decoration:italic; text-align:left; }
p { color:blue; font-decoration:italic; text-align:left; }
img { border:0; }
</style>
```

In this case, the statements specify the style properties for three tags: **h1**, **p**, and **img**.

The following HTML file, ThreeEmbeddedStyles.html, uses embedded styles to format the heading, paragraph, and image tags previously individually formatted using inline styles within the file RepeatStyles.html:

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
h1 { color:red; font-decoration:italic; text-align:left; }
p { color:blue; font-decoration:italic; text-align:left; }
img { border:0; }
```

```
</style>
</head>
<body>
<h1>Dogs</h1>
<p>Dogs are great!</p>
<img
src="http://www.websitedevelopmentbook.com/chapter07/dogs.jpg"/>
<h1>Cats</h1>
<p>Cats are great!</p>
<img
src="http://www.websitedevelopmentbook.com/chapter07/cats.jpg"/>
<h1>Horses</h1>
<p>Horses are great!</p>
<img
src="http://www.websitedevelopmentbook.com/chapter07/horses.jpg"/>
</body>
</html>
```

Using the embedded styles, you need to specify the style settings only one time. The browser, in turn, will apply the styles to each occurrence of the specified HTML tag. Should you decide to change a style, perhaps changing the **<h1>** tag's color from red to blue, you need to change only the one style definition. In this way, you can quickly and easily make changes throughout your entire page.

Inline Styles Override Embedded Styles

When you define style properties for an HTML element using embedded styles, you set the default format for each occurrence of the tag throughout your HTML file. After you define a default style, there may be times to use a unique format for a specific tag within your page. In such cases, override the embedded style definition using an inline style.

The following HTML file, MiddleYellow.html, uses embedded styles to define the background color and font for the paragraph selector:

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
p { color:blue; background-color:orange; }
</style>
</head>
<body>
```

```
<p>11111111</p>
<p style="background-color:yellow">2222222</p>
<p>33333333</p>
</body>
</html>
```

As you can see, the embedded style definition sets the background color for the **<p>** tag to orange. Upon examining the file's second paragraph tag, you will find that it uses an inline style to override the background color, setting it for that specific tag to a yellow background color:

```
<p style="background-color:yellow">2222222</p>
```

When you display the file's contents, the browser will display the contents shown in **FIGURE 7-5**.

The embedded style definitions set the default settings for each occurrence of an HTML tag throughout your page, making it easy to apply consistent formatting quickly across your entire page. Should you need one element, such as a paragraph, to look different from the default settings, use an inline style.

Making Quick Changes to Your Page

Using embedded styles, a developer can quickly apply the same styles to elements within an HTML file. In addition, should the developer later need to change a style definition, the one definition can be edited, and the change immediately applied throughout the page. Take time to experiment with the **<p>** tag style that appears at the top of the MiddleYellow.html file previously shown, perhaps changing the **background-color** property from orange to white. After you save and refresh the file's contents, the browser immediately applies the change throughout your file.

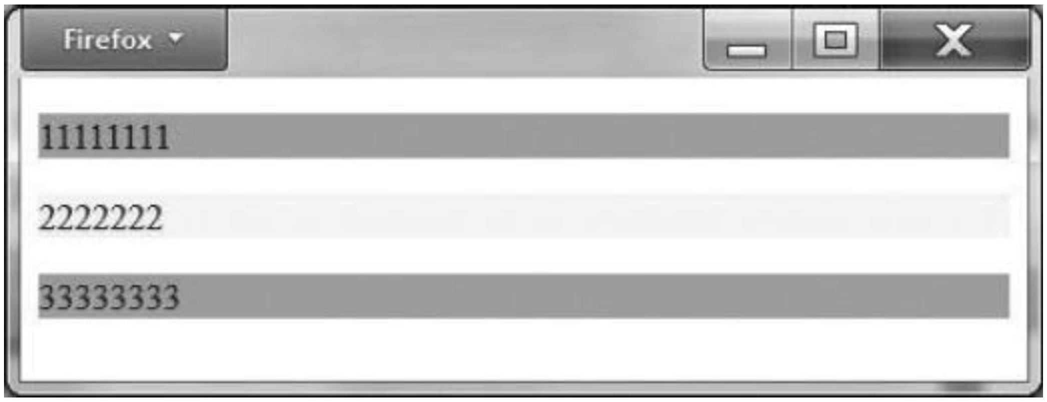


FIGURE 7-5 Using inline styles to override an embedded style

CREATING EXTERNAL STYLE SHEETS

As you have learned, embedded styles let you control the format of HTML elements throughout your entire page. In this way, you can quickly apply or later change the entire look and feel of your page. Most websites, however, consist of multiple pages. Admittedly, you could place the same embedded style definitions at the top of each page. However, doing so requires repetitive work. Further, should you later decide to change an element's appearance, it would become necessary to edit every page's corresponding file again—which is not only time-consuming but also error-prone.

An **external style sheet** lets you place all of your CSS style definitions within one file that you then reference from within each of your HTML page files. When a page is viewed, the browser downloads and applies the style definitions provided in the corresponding CSS file. Should you later need to change a style, simply edit the CSS file, and the changes automatically apply to the pages that use the file.

To create an external style sheet file, use a text editor, just as you would to create an HTML file. Within the external style sheet file, do not use the **<style>** and **</style>** tag pair to define your style properties. Instead, simply place the definitions within the file:

```
h1 { color:red; background-color:yellow;
font-family:'Comic Sans MS'; }
p { color:blue; background-color:orange;
font-family:Arial; }
```

You should assign a meaningful name to the external style sheet file and .CSS extension, such as SiteStyle.css.

Then, within the **<head>** and **</head>** tags for each of your HTML pages, place a **<link>** tag that specifies the URL of your style sheet file. If the CSS file resides in the same folder as your HTML files, use a relative URL, such as:

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

Within the **<link>** tag, the **rel** attribute defines the relationship between the HTML file and the external file, which in this case, specifies that the file contains a CSS style sheet. Likewise, the **type**attribute specifies the document's Multipurpose Internet Mail Extension (MIME) type, which specifies the document's underlying format.

If the CSS file resides in a different location from your HTML files, use an absolute URL:

```
<head>
```

```
<link rel="stylesheet" type="text/css"
href="http://www.WebSiteDevelopmentBook.com/Chapter07/SiteStyle.cs
s"/>
</head>
```

The following CSS style sheet, SiteStyles.css, defines several styles:

```
h1 { color:red; background-color:yellow;
font-family:'Comic Sans MS'; }
p { color:blue; background-color:orange;
font-family:Arial; }
h2 { color:white; background-color:grey; }
```

Next, the files One.html, Two.html, and Three.html all use the style definitions:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css"
href="http://www.WebSiteDevelopmentBook.com/Chapter07/SiteStyles.c
ss"/>
</head>
<body>
<h1>One</h1>
<h2>111</h2>
<p>One 1 One 1 One 1</p>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css"
href="http://www.WebSiteDevelopmentBook.com/Chapter07/SiteStyles.c
ss"/>
</head>
<body>
<h1>Two</h1>
<h2>222</h2>
<p>Two 2 Two 2 Two 2</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css"
href="http://www.WebSiteDevelopmentBook.com/Chapter07/SiteStyles.c
ss"/>
</head>
<body>
<h1>Three</h1>
<h2>3 3 3</h2>
<p>Three 3 Three 3 Three 3</p>
</body>
</html>
```

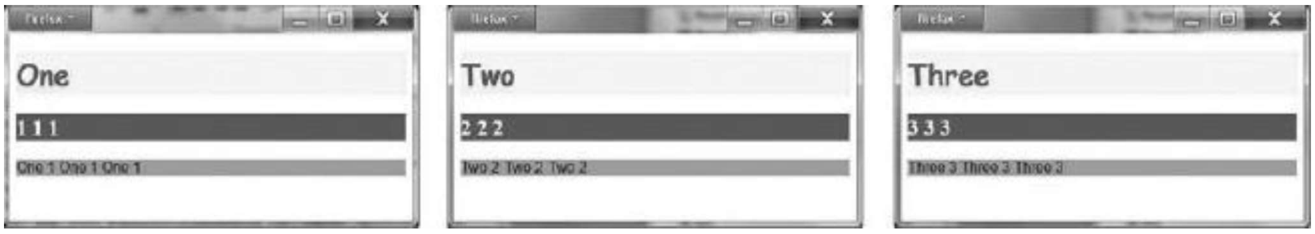


FIGURE 7-6 Using external style sheet definitions to style multiple pages

As you can see, each HTML file links in the external CSS file. When you view the files, each will use the style definitions, as shown in **FIGURE 7-6**. Edit the SiteStyles.css file, and change the **h1** selector's definition as follows:

h1 { color:white; background-color:blue; }

Save the SiteStyles.css file's contents. When you later view the HTML file's One.html, Two.html, or Three.html, each immediately displays the updated style, as shown in **FIGURE 7-7**.

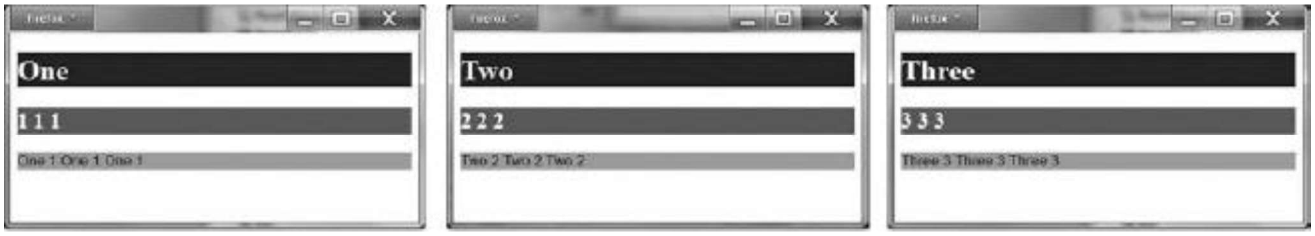


FIGURE 7-7 Applying a change to an external style sheet to multiple pages in one step

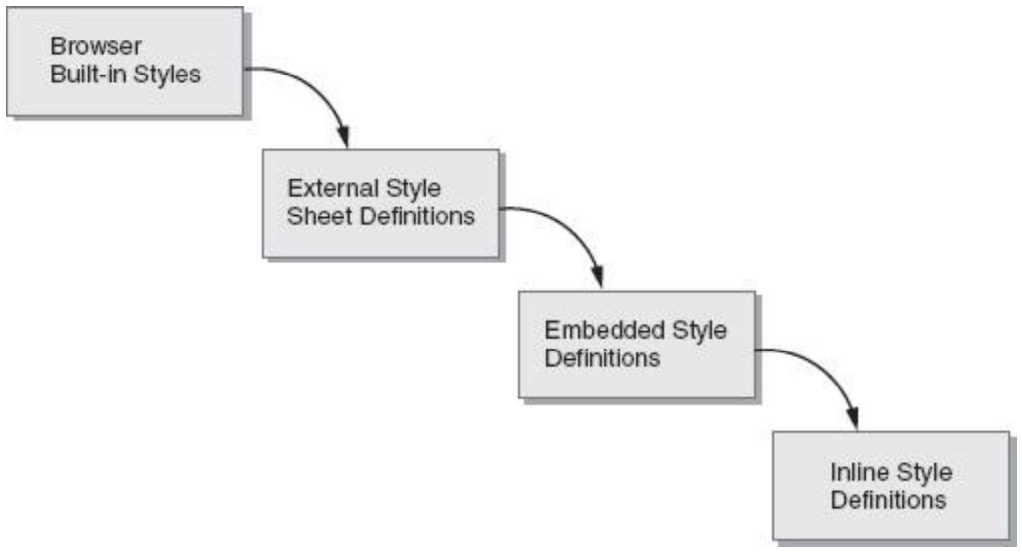


FIGURE 7-8 A browser will cascade through style definitions, applying the lowest level definition it encounters.

When your website has multiple pages, you should place most, if not all, of your style definitions in an external style sheet file.

UNDERSTANDING THE "CASCADING" IN CASCADING STYLE SHEETS

If you do not override a tag style using inline, embedded, or external style definitions, the browser will use its default (built-in) formatting for each HTML tag.

To apply styles across your entire site, use an external CSS style sheet. To change a style for a tag on a specific page, use an embedded style definition. Finally, to apply a style to a single element within a page, use an inline style.

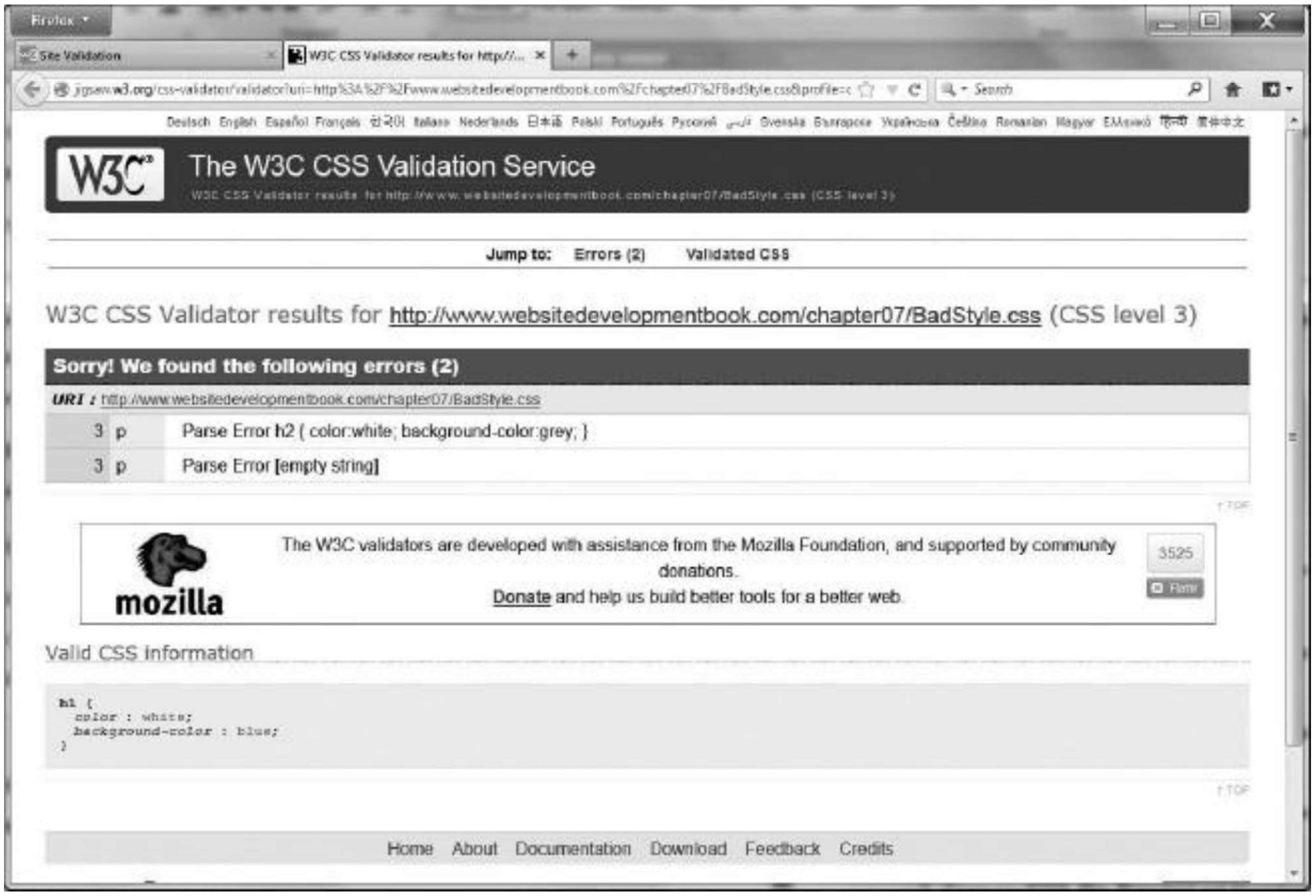
When the browser later displays your page, it will fall through the "cascade" of style definitions, applying the lowest level definitions that it encounters, as shown in **FIGURE 7-8**.

REAL-WORLD WEB DESIGN

Web developers should keep the W3Schools.com website close at hand. Not only does the site provide a detailed coverage of HTML, it also presents the available CSS properties. In addition, the site provides a CSS validator you can use to examine the styles you define at *http://www.w3schools.com/web/web_validate.asp*.

The following CSS style sheet file, BadStyle.css, has an error within the **p** selector definition; the definition does not include a closing right brace:

```
h1 { color:white; background-color:blue; }
p { color:blue; background-color:orange;
font-family:Arial;
h2 { color:white; background-color:grey; }
```


The W3C CSS Validation Service

Jump to: Errors (2) Validated CSS

W3C CSS Validator results for http://www.websitedevelopmentbook.com/chapter07/BadStyle.css (CSS level 3)

Sorry! We found the following errors (2)

URI : http://www.websitedevelopmentbook.com/chapter07/BadStyle.css

3	p	Parse Error h2 { color:white; background-color:grey; }
3	p	Parse Error [empty string]

The W3C validators are developed with assistance from the Mozilla Foundation, and supported by community donations.
Donate and help us build better tools for a better web.

Valid CSS information

```
h1 {
  color : white;
  background-color : blue;
}
```

FIGURE 7-9 Using the W3C CSS Validation service to examine a CSS file

Reproduced with permission of the World Wide Web Consortium (W3C)

If you examine the file using the validator, your screen will display an error message similar to that shown in **FIGURE 7-9**, which shows the errors caused by a single missing brace.
As a rule, each time you create an external style sheet, use the validator to confirm the file's contents.

HANDS-ON HTML

To view the HTML files or to experiment with the files presented in this chapter, visit this book's companion website at *http://www.websitedevelopmentbook.com/Chapter07/TryIt.html*.

CHAPTER SUMMARY

Across the Web, developers make extensive use of cascading style sheets to format elements within a webpage. By default, if a Web developer does not specify CSS attributes for an HTML element, a browser will use its own default, built-in formatting. To specify formatting using CSS, developers can use inline, embedded, and external styles. An inline style appears within an HTML element tag and affects only that occurrence of the element. An embedded style, in contrast, affects all of the corresponding tags within an HTML file. Finally, an external style sheet file defines styles that a developer can easily apply to multiple pages within a website. Using external style sheets, the developer can apply consistent styles throughout a site and easily and effectively add or modify styles. In Chapter 8, you will continue to drill down into CSS formatting capabilities.

KEY TERMS

Embedded style
External style
External style sheet
Inline style
Property
Style

CHAPTER REVIEW

1. Create an HTML file that uses inline styles and the **color** property to display **<h1>** tag contents in blue.

2. Create an HTML file that uses an inline style and the **font-family** and **font-size** properties to display a paragraph using a 14-point Arial font.

3. Create an HTML file that uses embedded style definitions to display all paragraph text in red italics at 12 points.

4. Combine embedded and inline styles to display two paragraphs in red italics text at 12 points and one paragraph at 14-point blue text.

5. Create two HTML files, Home.html and About.html, each of which uses the external style sheet definitions in a file named Custom.css (that you create) to display:

a. Blue background

b. Yellow heading text

c. White paragraph text

6. Use an embedded style sheet to modify the file Home.html, which you created in Question 5, to override the **<h1>** style definition to display the text in orange.

7. Modify the Home.html file from Question 6 to add a second **<h1>** tag that displays its text in white using an inline style.

8. Use the W3School's validator at *http://www.w3schools.com/web/web_validate.asp* to validate the contents of your Custom.css file, and discuss your findings.