

# INTRODUCING CASCADING STYLE SHEETS

**When you complete this chapter, you will be able to:**

- ◆ Understand CSS style rules
- ◆ Build a basic style sheet
- ◆ Understand the cascade
- ◆ Use basic selection techniques
- ◆ Use advanced selection techniques

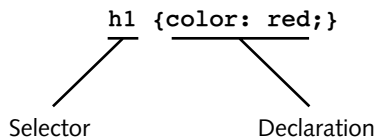
Cascading Style Sheets (CSS) let you control the display characteristics of your Web site. In this chapter, you examine the basic syntax of CSS and learn how to combine CSS rules with your XHTML code. You start by examining the CSS style rules, then apply them to build a basic style sheet. Then you learn how to cascade style sheets or let multiple style sheets and style rules apply to the same document. You also learn basic selection techniques to apply a particular style declaration to an element in your document. Finally, you build on this information to use the advanced selection techniques of using the class attribute and the `<div>` and `<span>` elements.

## UNDERSTANDING CSS STYLE RULES

In CSS **style rules** express the style characteristics for an XHTML element. A set of style rules is called a **style sheet**. Style rules are easy to write and interpret. The following code shows a simple style rule for the `<p>` element. Note that the style rules are contained in the `<style>` element in the document's `<head>` section. This rule sets all `<p>` elements in the document to blue 24-point text:

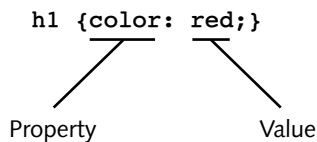
```
<head>
<style type="text/css">
p {color: blue; font-size: 24pt;}
</style>
</head>
```

A style rule is composed of two parts: a selector and a declaration. The style rule expresses the style information for an element. The **selector** determines the element to which the rule is applied. The **declaration** details the exact property values. Figure 6-1 shows an example of a simple style rule that sets all `<h1>` headings to red:



**Figure 6-1** Style rule syntax

As illustrated in Figure 6-2, the declaration contains a property and a value. The **property** is a quality or characteristic, such as color, font size, or margin, followed by a colon (:). The **value** is the precise specification of the property, such as blue for color, 12 pt (point) for font size, or 30 px (pixels) for margin, followed by a semicolon (;). CSS contains a wide variety of properties, each with a specific list of values.



**Figure 6-2** Property declaration syntax

Figure 6-2 shows a basic style rule. As you will see later in this chapter, you can combine selectors and property declarations in a variety of ways.

**TIP**

This chapter uses a variety of CSS style rules as examples. Although you have not yet learned about their properties in detail, you will see that the CSS property names express common desktop publishing characteristics such as font family, margin, text indent, and so on. The property values sometimes use abbreviations such as “px” for pixel and “pt” for point, percentages such as 200%, or keywords such as “bold”. You will learn about these properties and values in detail as you progress through this book.

## Combining CSS Style Rules with XHTML

You can combine CSS rules with XHTML code in the following three ways:

- The style attribute
- The <style> element
- An external style sheet

Each method is discussed in detail in the following sections.

### Using the style Attribute

You can define the style for a single element using the style attribute.

```
<h1 style="color: blue">Some Text</h1>
```

You generally use the style attribute to override a style that was set at a higher level in the document, as when you want a particular heading to be a different color from the rest of the headings on the page. The style attribute is also useful for testing styles during development. You will probably use this method of styling an element the least, because it affects only one instance of an element in a document.

### Using the <style> Element

The <style> element is always contained in the <head> section of the document. Style rules contained in the <style> element affect only the document in which they reside. The following code shows a <style> element that contains a single style rule:

```
<head>
<title>Sample Document</title>
<style type="text/css">
h1 {color: red;}
</style>
</head>
```

In the previous code, note the type attribute to the <style> element. The value “text/css” defines the style language as Cascading Style Sheets. Although not required, the type attribute should always be included in all of your <style> elements for future compatibility as more style languages become available.

## Using External Style Sheets

Placing style sheets in an external document lets you specify rules for multiple Web pages. This is an easy and powerful way to use style sheets. An external style sheet is simply a text document that contains the style rules. External style sheets have a .css extension. Here's an example of a simple external style sheet named styles.css:

```
h1 {color: white; background-color: green;}
h2 {color: red;}
```

The style sheet file does not contain any XHTML code, just CSS style rules, because the style sheet is not an XHTML document. It is not necessary to use the <style> element in an external style sheet.

## Linking to an External Style Sheet

The <link> element lets you establish document relationships. It can be used only within the <head> section of a document. To link to an external style sheet, add the <link> element as shown in the following code:

```
<head>
<title>Sample Document</title>
<style type="text/css">
<link href="styles.css" rel="stylesheet">
</style>
</head>
```

The <link> element in this code tells the browser to find the specified style sheet. The href attribute states the relative URL of the style sheet. The rel attribute specifies the relationship between the linked and current documents. The browser displays the Web page based on the CSS display information. The advantage of the external style sheet is that you can state the style rules in one document and affect all the pages on a Web site. When you want to update a style, you have to change the style rule only once in the external style sheet.

## Adding Comments

CSS allows comments within the <style> element or in an external style sheet. CSS comments begin with the slash and asterisk characters (/\*) and end with the asterisk and slash characters (\*). You can use comments in a variety of ways, as shown in the following code:

```
<style type="text/css">
/* This is the basic style sheet */
h1 {color: gray;} /* The headline color */
h2 {color: red;} /* The sub-head color */
</style>
```

Comments provide documentation for your style rules. Because they are embedded directly in the style sheet, they provide immediate information to anyone who needs to understand how the style rules work. Comments are always useful and you should consider using them in all of your code, whether as a simple reminder to yourself or as an aid to others with whom you work.

## BUILDING A BASIC STYLE SHEET

In the following set of steps, you will build and test a basic style sheet. Save your file and test your work in the browser as you complete each step. Refer to Figure 6-3 as you progress through the steps to see the results you will achieve.

To build a basic style sheet:

1. Copy the **basic.htm** file from the Chapter06 folder provided with your Data Files to the Chapter06 folder in your work folder. (Create the Chapter06 folder, if necessary.)
2. In your browser, open **basic.htm**. When you open the file, it looks like Figure 6-3.

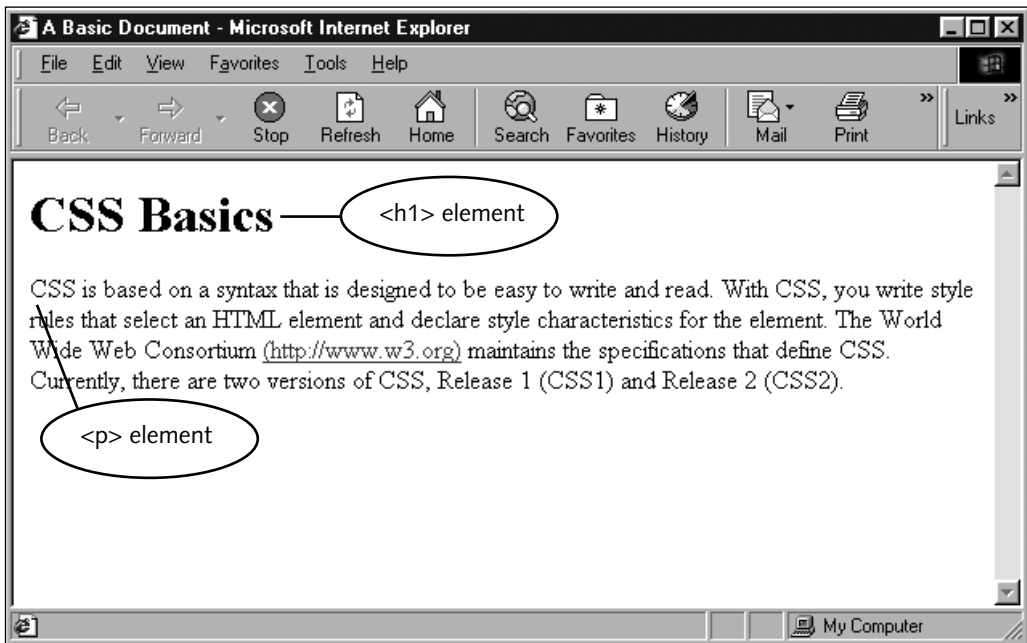


Figure 6-3 The original HTML document

- Open the **basic.htm** file in your XHTML editor and examine the code. Notice that the file contains basic XHTML code with no style information. The complete code for the page follows:

```
<html>
<head>
<title>A Basic Document</title>
</head>
<body>
<h1>CSS Basics</h1>
<p>CSS is based on a syntax that is designed to
be easy to write and read. With CSS, you write
style rules that select an HTML element and
declare style characteristics for the element.
The World Wide Web Consortium <a
href="http://www.w3.org">(http://www.w3.org)</a>
maintains the specifications that define CSS.
Currently, there are two versions of CSS, Release
1 (CSS1) and Release 2 (CSS2).</p>
</body>
</html>
```

- Add a `<style>` element in the **<head>** section to contain your style rules as shown in the following code. Leave a few lines of white space between the `<style>` tags to contain the style rules.

```
<head>
<style type="text/css">

</style>
</head>
```

- Add a style rule for the `<h1>` element as shown in the following shaded code fragment. This style rule uses the `text-align` property to center the heading.

```
<head>
<style type="text/css">
h1 {text-align: center;}
</style>
</head>
```

- Save the file as **basic.htm** in the Chapter06 folder in your work folder, then reload the file in the browser. The `<h1>` element is now centered, as shown in Figure 6-4.

7. Add a style rule for the `<p>` element as shown in the following code fragment. This style rule uses the `font-family` property to specify sans-serif font for the paragraph text.

```
<head>
<style type="text/css">
h1 {text-align: center;}
p {font-family: sans-serif;}
</style>
</head>
```

8. Save the file as **basic.htm** in the Chapter06 folder, then reload it in the browser. Figure 6-4 shows the finished Web page. Notice that the `<p>` element is now displayed in a sans-serif typeface.

6

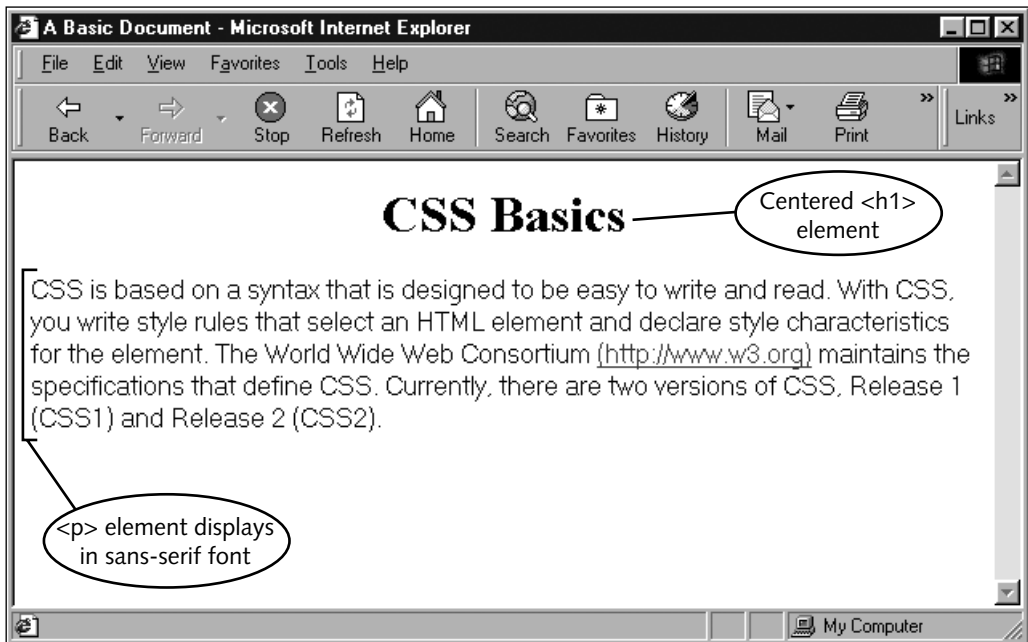


Figure 6-4 The HTML document styled with CSS

## UNDERSTANDING THE CASCADE

One of the fundamental features of CSS is that style sheets **cascade**. This means that multiple style sheets and style rules can apply to the same document. XHTML authors can attach a preferred style sheet, while the reader might have a personal style sheet to adjust for preferences such as human or technological handicaps. However, only one rule

can apply to an element. The CSS cascading mechanism determines which rules are applied to document elements by assigning a weight to each rule based on the following four variables, listed in the order in which they are applied:

- Use of the `!important` keyword
- Origin of the rule
- Specificity of the selector
- Order of the rule in the style sheet

## Determining Rule Weight with the `!important` Keyword

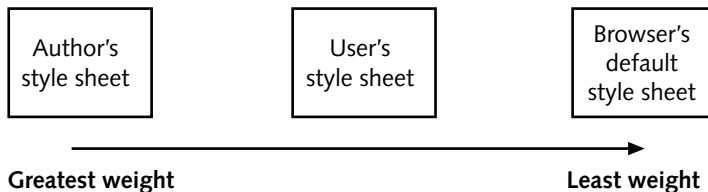
A conflict can arise when both the author's and user's style sheet contain a rule for the same element. By default, rules in an author's style sheet override those in a user's style sheet. To balance the bias toward the author's style sheet, CSS has an `!important` keyword. **`!important`** lets the user override the author's style setting for a particular element. The following user's style sheet states a rule for `<p>` elements that sets the font size to 18 points, regardless of the rule supplied by the author of the document:

```
<style type="text/css">
p {font-size: 18pt !important}
</style>
```

This CSS feature improves accessibility of documents by giving users with special requirements control over document presentation, such as increasing font size or changing color contrast.

## Determining Rule Weight by Origin

A style rule's weight can be determined by the style sheet in which it resides. CSS allows style sheets to be applied by the author, the user, and the browser. Figure 6-5 shows the style sheet order of precedence.



**Figure 6-5** The cascading order of precedence

In the cascading order, rules from the author's style sheet have the greatest weight. This is the page display that most users want to see—the author's intended page design.

The user's style sheet is next in order of importance. Although the designer's rules have more weight, users have the option of turning off the author's styles in the browser or

using the `!important` keyword to give their rules more weight. If the browser allows, the user can attach his or her style sheet to the document. This allows the user to adjust, for example, the font size or link color to make a page more legible.

The browser's style sheet has the least weight. This is the style sheet that contains the default display information, such as displaying an `<h1>` heading in Times Bold with a carriage return before and after. The browser's style sheet controls the display of elements that do not have an associated style rule.

## Determining Rule Weight by Specificity

Another method of determining style rule weight is the specificity of the rule's element selector. Rules with more specific selectors take precedence over rules with less specific selectors. Examine the following style rules:

```
body {color: black;}
h1 {color: red;}
```

The first rule uses a nonspecific selector, the `<body>` element. This rule sets the text color for all elements within `<body>` to black. The second rule has a much more specific selector that sets a rule only for `<h1>` elements. Because the second rule has a more specific selector, it takes precedence for all `<h1>` elements within the document.

## Determining Rule Weight by Order

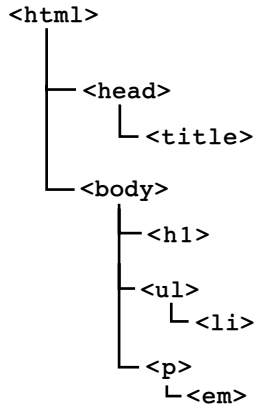
CSS applies weight to a rule based on its order within a style sheet. Rules that are included later in the style sheet order take precedence over earlier rules. Examine the following style rules for an example:

```
body {color: black;}
h1 {color: red;}
h1 {color: green;}
```

In this example, `<h1>` elements in the document appear green because the last style rule specifies green as the color.

## Understanding Inheritance

The elements in an XHTML document are structured in a hierarchy of parent and child elements. Figure 6-6 represents the structure of a simple XHTML document.



**Figure 6-6** HTML document structure

Note the hierarchical structure of the elements. `<html>` is the parent element of the document. **Parent elements** contain nested elements called **child elements**. Both `<head>` and `<body>` are immediate child elements of `<html>`. `<head>` and `<body>` are parent elements as well, because they contain other nested elements. As you travel further down the document hierarchy, you find other elements that are both parent and child elements, such as `<p>` and `<ul>`.

By default, CSS rules are inherited from parent elements to child elements. Therefore, if you set a style rule for `<ul>` elements in the document shown in Figure 6-6, the `<li>` elements inherit the style rules, unless you have specifically set a rule for `<li>`.

You can style multiple document elements with just a few style rules if you let inheritance work for you. For example, consider the following set of style rules for a document.

```

<style type="text/css">
h1 {color: red;}
p {color: red;}
ul {color: red;}
em {color: red;}
li {color: red;}
</style>

```

This style sheet sets the color to red for five different elements in the document. Inheritance lets you write a far simpler rule to accomplish the same results:

```

<style type="text/css">
body {color: red;}
</style>

```

This rule works because all of the elements are children of `<body>` and because all the rules are the same. It is much more efficient to write a single rule for the parent element and let the child elements inherit the style. Because `<body>` is the parent element of the content area of the XHTML file, it is the selector to use whenever you want to apply a style across the entire document.

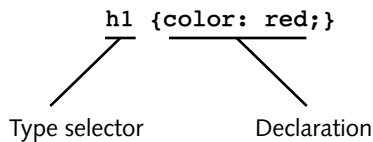
## UNDERSTANDING BASIC SELECTION TECHNIQUES

In this section you will review style rule syntax and learn about the following basic selection techniques:

- Using type selectors
- Grouping selectors
- Combining declarations
- Using descendant selectors

### Using Type Selectors

As you learned previously, the selector determines the element to which a style declaration is applied. To review, examine the syntax of the style rule shown in Figure 6-7. This rule selects the `<h1>` element in the document and sets the text color to red.



**Figure 6-7** Style rule syntax

This rule uses a **type selector** to apply the rule to every instance of the element in the document. This is the simplest type of selector, and many style sheets are composed primarily of type selector style rules, as shown in the following code:

```
body {color: gray;}
h2 {color: red;}
p {font-size: 10pt;}
```

### Grouping Selectors

To make your style rules more concise, you can group selectors to which the same rules apply. For example, the following style rules set the same declaration for two different elements—they set the color of `<h1>` and `<h2>` elements to red:

```
h1 {color: red;}
h2 {color: red;}
```

These two style rules can be expressed in a simpler way by separating the selectors with commas:

```
h1, h2 {color: red;}
```

## Combining Declarations

In many instances you want to state multiple property declarations for the same selector. The following style rules set the <p> element to 12-point blue text:

```
p {color: blue;}
p {font-size: 12pt;}
```

These two style rules can be expressed in a simpler fashion by combining the declarations in one rule. The declarations are separated by semicolons:

```
p {color: blue; font-size: 12pt;}
```

## Using Descendant Selectors

A descendant selector (sometimes known as a contextual selector) is based on the hierarchical structure of the elements in the document tree. This selector lets you select elements that are the descendants of other elements. For example, the following rule selects only <b> elements that are contained within <p> elements. All other <b> elements in the document are not affected.

```
p b {color: blue;}
```

Notice that the selector contains multiple elements, separated only by white space. You can use more than two elements if you prefer to choose more specific selection characteristics. For example, the following rule selects <b> elements within <li> elements within <ul> elements only:

```
ul li b {color: blue;}
```

## Using the Basic Selection Techniques

In the following set of steps, you will build a style sheet that uses basic selection techniques. Save your file and test your work in the browser as you complete each step. Refer to Figure 6-8 as you progress through the steps to see the results you will achieve.

To build the style sheet:

1. Copy the **oz.htm** file from the Chapter06 folder provided with your Data Files to the Chapter06 folder in your work folder. Open the file **oz.htm** in your HTML editor and save it as **oz1.htm** in the same location.
2. In your browser, open the file **oz1.htm**. When you open the file, it looks like Figure 6-8.

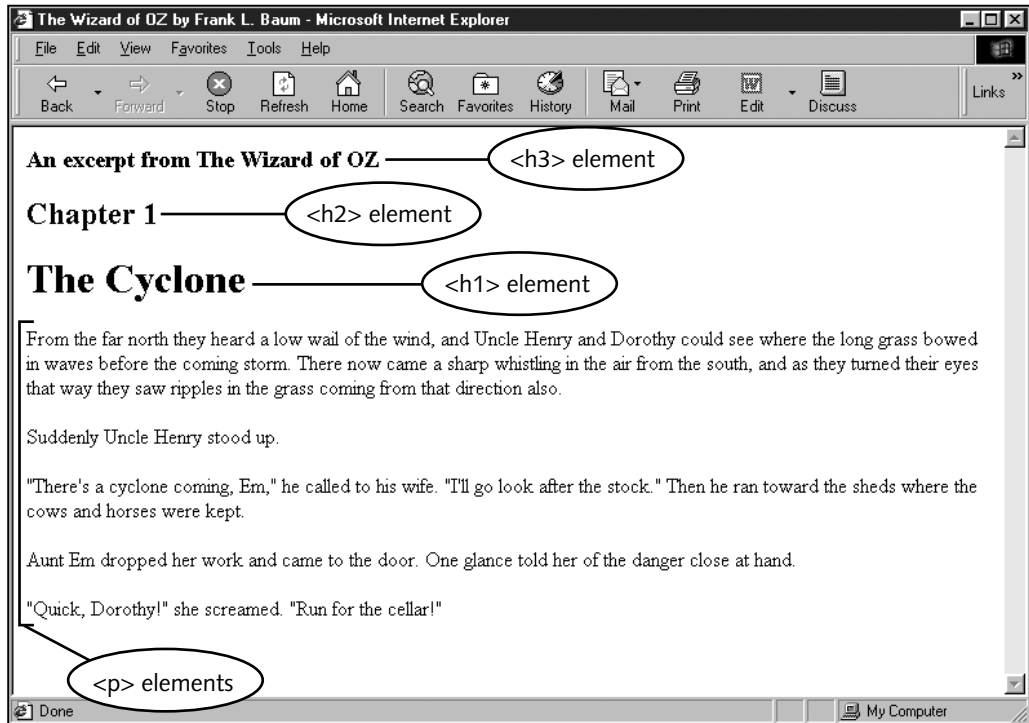


Figure 6-8 The original oz1.htm HTML document

3. Examine the code. Notice that the file contains basic XHTML code with no style information. The complete code for the page follows:

```

<html>
<head>
<title>The Wizard of OZ by Frank L. Baum</title>
</head>
<body>
<h3>An excerpt from The Wizard of OZ</h3>
<h2>Chapter 1</h2>
<h1>The Cyclone</h1>
<p>From the far north they heard a low wail of the wind, and Uncle Henry and Dorothy could see where the long grass bowed in waves before the coming storm. There now came a sharp whistling in the air from the south, and as they turned their eyes that way they saw ripples in the grass coming from that direction also.</p>
<p>Suddenly Uncle Henry stood up.</p>
<p>"There's a cyclone coming, Em," he called to his wife. "I'll go look after the stock." Then he ran toward the sheds where the cows and horses were kept.</p>

```

```

<p>Aunt Em dropped her work and came to the door.
One glance told her of the danger close at
hand.</p>
<p>"Quick, Dorothy!" she screamed. "Run for the
cellar!"</p>
</body>
</html>

```

4. Add a `<style>` element in the `<head>` section to contain your style rules as shown in the following code. Leave a few lines of white space between the `<style>` tags to contain the style rules.

```

<head>
<style type="text/css">

</style>
</head>

```

5. Write the style rule for the `<h3>` element. The requirements for this element are right-aligned gray text. The style rule looks like this:

```

<head>
<style type="text/css">
h3 {text-align: right; color: gray;}
</style>
</head>

```

6. Write the style rules for the `<h1>` and `<h2>` elements, which share some common property values. Both elements have a left margin of 20 pixels (abbreviated as px) and a sans-serif font style. Because they share these properties, group the two elements to share the same style rule as shown in the following code:

```

<head>
<style type="text/css">
h3 {text-align: right; color: gray;}
h1, h2 {margin-left: 20px; font-family: sans-serif;}
</style>
</head>

```

7. Write an additional rule for the `<h1>` element. The `<h1>` element has two style properties that it does not share with `<h2>`, so a separate style rule is necessary to express the border and padding white space within the border. This rule uses the border shortcut property to specify multiple border characteristics—a 1-pixel border weight and solid border style.

```

<head>
<style type="text/css">
h3 {text-align: right; color: gray;}
h1, h2 {margin-left: 20px; font-family: sans-serif;}
h1 {border: 1px solid; padding: 5px;}
</style>
</head>

```

- Write a style rule for the <p> elements so they have a 20-pixel left margin (to line up with the other elements on the page), a serif font style, and a 14-point font size.

```
<head>
<style type="text/css">
h3 {text-align: right; color: gray;}
h1, h2 {margin-left: 20px; font-family: sans-serif;}
h1 {border: 1px solid; padding: 5px;}
p {margin-left: 20px; font-family: serif; font-size: 14pt;}
</style>
</head>
```

Figure 6-9 shows the finished document with the style properties.

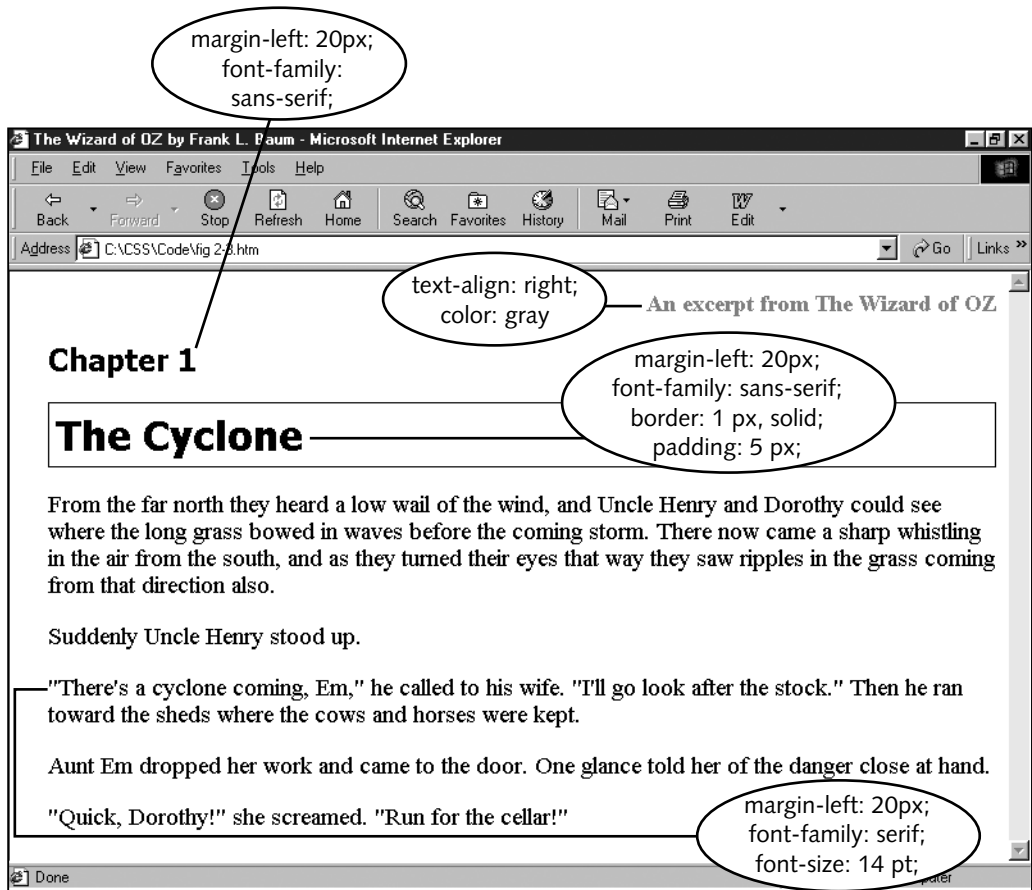


Figure 6-9 The oz1.htm HTML document styled with CSS rules

## UNDERSTANDING ADVANCED SELECTION TECHNIQUES

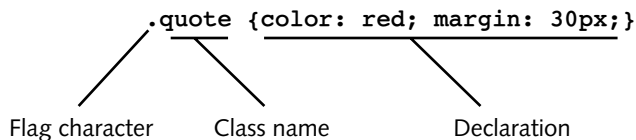
This section describes the CSS advanced selection techniques that are supported in Internet Explorer 6.0, Netscape 7.1, and Opera 6.0 (and later versions of these browsers). These techniques allow more than the basic element-based selection capabilities described in the previous section. You will learn to select elements of an XHTML document using the following methods:

- The class attribute
- The <div> and <span> elements

### Using the class Attribute Selector

The class attribute lets you write rules, give them a name, and then apply that name to any elements you choose. You can use the class attribute with any XHTML element because it is a core attribute. The **core attributes** are id, class, style, and title, and they apply to all XHTML elements. To apply a style rule to an element, you can add the class attribute to the element and set it to the name you have specified.

To create a class, declare a style rule. The period (.) flag character indicates that the selector is a class selector. Figure 6-10 shows an example of a rule with a class selector.



**Figure 6-10** Class syntax

After writing the style rule, add it to the document by using the class attribute, as shown in the following code fragment:

```
<p class="quote">This text will appear red with a 30-pixel margin.</p>
```

The class attribute lets you select elements with greater precision. For example, read the following style rule:

```
p {font-size: 10pt;}
```

This rule sets all <p> elements in the document to a font size of 10 points. Now suppose that you want one <p> element in your document to have a different style characteristic, such as bold text. You need a way to specifically select that one paragraph. To do this, use a class selector. The following style rule sets the style for the class named special.

```
.special {font-size: 10pt; font-weight: bold;}
```

The class selector can be any name you choose. In this instance, the class name special denotes a special paragraph of the document. Now apply the rule to the <p> element in the document using the class attribute:

```
<p class="special">This is the first paragraph of
the document. It has a different style based on
the "special" class selector.</p>
```

```
<p>This is the second paragraph of text in the
document. It is a standard paragraph without a
class attribute.</p>
```

Figure 6-11 shows the result of the style rule.

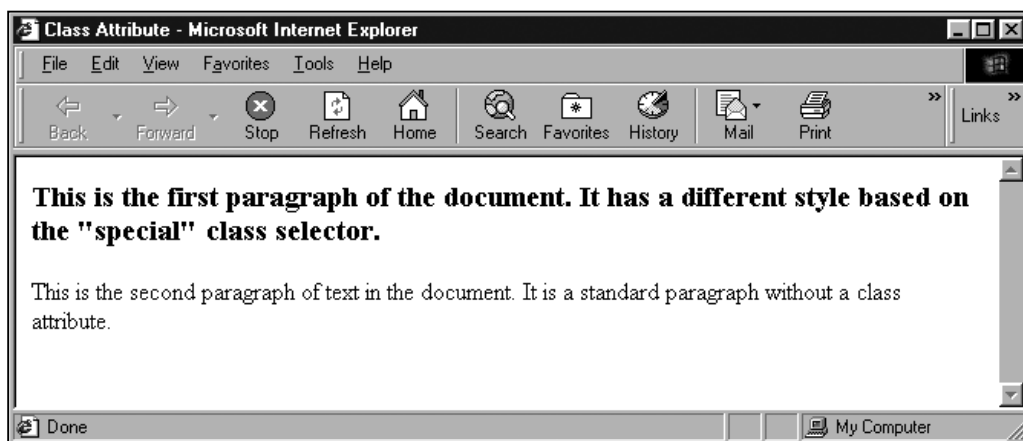


Figure 6-11 Styling with a class attribute

## Making class Selectors More Specific

Using the class attribute is a powerful selection technique, because it allows you to write style rules with names that are meaningful to your organization or information type. The more specific your class names become, the greater control you need over the way they are applied. In the preceding example, you saw a style rule named special that was applied to a <p> element. However, the special style can be applied to any element in the document, not just <p>. To solve this problem, you can restrict the use of the class attribute to a single element type.

For example, your organization might use a special style for a procedure heading. The style is based on an <h1> element, with a sans-serif font and left margin of 20 pixels. Everyone in your organization knows this style is named procedure. You can use this same style name in your style sheet as shown in the following style rule:

```
.procedure {font-family: sans-serif; margin-left:
20px;}
```

To use these rules in the document, you apply the class attribute as shown in the following code fragment:

```
<h1 class="procedure">Procedure Heading</h1>
```

This works well, but what happens if someone on your staff neglects to apply the classes properly? For the style rule to work, it must be applied to an `<h1>` element. To restrict the use of the class to `<h1>` elements, include a prefix for the class selector with the element to which you want it applied:

```
h1.procedure {font-family: sans-serif; margin-left: 20px;}
```

These style rules restrict the use of the procedure style to `<h1>`.

## Using the `<div>` and `<span>` Elements

The `<div>` (division) and `<span>` (span of words) elements are designed to be used with CSS. They let you specify logical divisions within a document that have their own name and style properties. The difference between `<div>` and `<span>` is their element display type, which is described in more detail in Chapter 9. `<div>` is a block-level element, and `<span>` is its inline equivalent. Used with the class and id attributes, `<div>` and `<span>` let you effectively create your own element names for your XHTML documents.

### Working with `<div>`

You can use `<div>` with the class attribute to create customized block-level elements. Like other block-level elements, `<div>` contains a leading and trailing line break. However, unlike other block-level elements, `<div>` contains no additional white space around the element. You can set the margin or padding to any value that you wish. You will learn more about these properties in Chapter 9.

To create a customized division, declare it with a class or selector in the style rule. The following example specifies a division with a class named `introduction` as the selector for the rule:

```
div.introduction {font-size: 14pt; margin: 24pt; text-indent: 28pt;}
```

To apply this rule, specify the `<div>` element in the document. Then use the class attribute to specify the exact type of division. In the following example, the code defines the `<div>` element as the class named `introduction`.

```
<div class="introduction">This is the  
introduction to the document.</div>
```

### Working with `<span>`

The `<span>` element lets you specify inline elements within a document that have their own name and style properties. Inline elements reside within a line of text, such as the

`<b>` or `<em>` element. You can use `<span>` with a class or attribute to create customized inline elements.

To create a span, declare it within the `<style>` element first. The following example specifies a span named `logo` as the selector for the rule:

```
span.logo {color: white; background-color:
black;}
```

Next, specify the `<span>` element in the document. Then use the class attribute to specify the exact type of span. In the following example, the code defines the `<span>` element as the class named `logo`.

```
<p>Welcome to the <span class="logo">Wonder
Software</span> Web site.</p>
```

Figure 6-12 shows the result of the style rule.



Figure 6-12 Using the `<span>` element

## CHAPTER SUMMARY

This chapter presents the basic syntax of the CSS language. You learned about the different methods of selecting elements and how to apply style rules in a variety of ways. You saw that the CSS basic selection techniques are often powerful enough to handle most document styling. Also, you learned that the class attribute lets you create naming conventions for styles that are meaningful to your organization or information type. As you will see in the upcoming chapters, CSS is an easy-to-use style language that lets you gain visual control over the display of your Web content.

- CSS rules can be combined with your XHTML code in a number of ways. CSS rules are easy to write and read.
- CSS uses cascading and inheritance to determine which style rules take precedence. The `!important` declaration lets users override the author's style rules.
- Basic style rules let you apply style rules based on standard element selectors. You can combine the selectors and declarations to create more powerful style expressions. You can also select elements based on the contextual relationship of elements in the document tree.
- The advanced selection techniques allow you to use the class attribute selector, which is often paired with the `<div>` and `<span>` XHTML elements. These elements have no style of their own, but offer a convenient way of expressing style for any section of a document, whether block-level or inline. Additionally, the class attribute allows you to choose a meaningful naming convention for your style rules.

---

## REVIEW QUESTIONS

1. What are the two parts of a style rule?
2. What are the three ways to combine CSS rules with your XHTML code?
3. List two reasons to state a style using the style attribute.
4. What are the advantages of using an external style sheet?
5. What is the inheritance default for CSS rules?
6. What is the benefit of the !important declaration?
7. Write a basic style rule that selects <h1> elements and sets the color property to red.
8. Add the <p> element as an additional selector to the rule you created for question 7.
9. Add a font-size property to the rule and set the size to 14 points.
10. Write a style rule that selects <ul> elements only when they appear within <p> elements and set the color property to red.
11. Write the style rule for a class selector named note. Set the font-weight property to bold.
12. Restrict the rule you developed for question 11 so it can be used only with <p> elements.
13. What is the difference between <div> and <span>?
14. Write a style rule that sets the default document text color to red.
15. What is the advantage of working with the class attribute?
16. What element does this selector choose?  
`p ul li`
17. What element does this selector choose?  
`div p *`
18. What element does this selector choose?  
`p.warning`

---

## HANDS-ON PROJECTS



1. Visit the World Wide Web Consortium Web site ([www.w3.org](http://www.w3.org)). Find the Cascading Style Sheets Release 2 specification. List and describe ten style properties that you can affect with a style rule.
2. By yourself or with a partner, choose a mainstream publishing Web site, such as a newspaper or periodical site. Examine the style characteristics of the site. What common styles can be applied across the site, such as headings, paragraphs, and bylines? Write an analysis of the site's style requirements, and list the styles you would include in the site's style sheet.

3. Jakob Nielsen is a well-known expert on interface design. Read his article on CSS at [www.useit.com/alertbox/9707a.html](http://www.useit.com/alertbox/9707a.html) and write a short paper describing his views and what you learned that you can implement in your own CSS design efforts.
4. In this project you will have a chance to test a few simple style rules on a standard XHTML document and view the results in your browser.
  - a. Using your XHTML editor, create a simple XHTML file (or open an existing file) that contains `<body>`, `<h1>`, and `<p>` elements. Save the file as **csstest1.htm** in the Chapter06 folder in your work folder.
  - b. Add a `<style>` element to the `<head>` section as shown in the following code:

```
<head>
<title>CSS Test Document</title>
<style type="text/css">

</style>
</head>
```

- c. Add a style rule that uses `body` as a selector and sets the `color` property to green, as shown in the following code:
- d. Save **csstest1.htm** and view it in the browser. All of the document text should now be green.
- e. Now add a style rule that sets `<h1>` elements to be displayed in black:

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

- f. Save **csstest1.htm** and view the results in the browser.
- g. Finally, add a style rule that sets a margin for `<p>` elements to 30 pixels:

```
<style type="text/css">
body {color: green;}
h1 {color: black;}
p {margin: 30px;}
</style>
```

- h. Save **csstest1.htm** and view the results in the browser.
5. In this project you will have a chance to test a few basic selection techniques on a standard XHTML document and view the results in your browser. Save the file and view it in your browser after completing each step.
  - a. Using your XHTML editor, create a simple XHTML file (or open an existing file) that contains `<body>`, `<h1>`, `<p>` elements, and so on. Save the file in the Chapter06 folder in your work folder as **csstest2.htm**.

- b. Add a `<style>` element to the `<head>` section as shown in the following code:

```
<head>
<title>CSS Test Document</title>
<style type="text/css">

</style>
</head>
```

- c. Write a style rule that uses `body` as a selector and sets the `color` property to the color of your choice.
- d. Find two elements on the page, such as `<h1>` and `<h2>`, that can share the same characteristics. Write a single style rule that applies to both elements. Set the `color` property to red and the `margin` property to 20 pixels.
- e. Find one element that contains another, such as a `<b>` or `<q>` element within a `<p>` element. Write a descendant selector rule that affects the contained element and sets the `color` property to green.
6. In this project you will have a chance to test a few advanced selection techniques on a standard XHTML document and view the results in your browser. Save the file and view it in your browser after completing each step.
- a. Using your XHTML editor, create a simple XHTML file (or open an existing file) that contains `<body>`, `<h1>`, `<p>` elements, and so on. Save the file in the `Chapter06` folder in your work folder as **csstest3.htm**.
- b. Add a `<style>` element to the `<head>` section as shown in Exercise 4.
- c. Write a rule for a class selector named `heading`. Set the `color` property to red and the `font-size` property to 36 points. Apply the `heading` class to an `<h1>` or `<h2>` element in the document.
- d. Write a rule for a class selector named `emphasis`. Set the `color` property to yellow. In the document, add a `<span>` element to a span of words that you want to highlight. Apply the `emphasis` class to the `<span>` element.

---

## CASE PROJECT



Revisit your project proposal and the page designs you created in Chapter 5. How will you implement Cascading Style Sheets into your project Web site? In the next few chapters, you will learn how to control typography, white space, borders, colors, and backgrounds with CSS. Think about each of these style characteristics and how you will apply them to your page designs. Additionally, make a list of possible class names you might use to identify your content. For example, consider using class names for the following page characteristics, as well as creating some of your own:

- Body copy
- Header (possible different levels)
- Footer