CHAPTER 1

INTRODUCING CASCADING STYLE SHEETS

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

♦ Understand the history of HTML and CSS, and how CSS is supported in the major browsers
♦ Explain why the separation of HTML structure from style information is so important to the future of the Web
♦ Identify what enhancements are included in CSS Release 2 (CSS2)
♦ Use basic CSS syntax and combine style rules with your HTML code
♦ Understand the concepts of the cascading mechanism and inheritance

In this chapter, you will learn about the history and implementation of Cascading Style Sheets (CSS). By examining the ongoing evolution of Hypertext Markup Language (HTML), you will see how the popularity of the Web quickly overcame the capabilities of HTML and why the need for an easy-to-use, expressive style language became apparent. You will also learn how the support for CSS in the popular browsers dictated whether HTML authors could successfully implement this new style language and why it has taken so long for CSS to be accepted. You will see why HTML structure information must be separate from style information and why the current methods of styling Web pages is hindering the accessibility of Web documents. You will also examine the basic syntax of CSS and learn how to combine CSS rules with your HTML code.
SECTION 1: A BRIEF HISTORY OF HTML AND CSS

When Tim Berners-Lee first proposed HTML in 1989, he was looking for a simple way to manage and share large amounts of information among colleagues. Berners-Lee created HTML from the Standard Generalized Markup Language (SGML), a standard system for specifying document structure. When Berners-Lee created HTML, he adopted only the necessary elements of SGML for representing basic office documents such as memos and reports. The first working draft of HTML included elements such as titles, headings, paragraphs, and lists.

Berners-Lee wanted to make documents as portable as possible, so he decided to let the user’s browser determine how the HTML pages were displayed. The first browser, Mosaic, had little support for any style characteristics, and most Web pages were simple, left-justified documents. Figure 1-1 shows a basic Web page displayed in the Mosaic 1.0 browser.

As the Web evolved, its potential to be an ideal publishing medium became obvious. What Web developers needed was a way to have more control over the visual display of content.
Users and publishers who were familiar with the visual display power of modern page layout and word-processing tools found HTML sorely lacking in its ability to handle even the most basic display characteristics, such as changing the color or typeface of text. Berners-Lee and his colleagues realized early in the development of HTML that a style and display language, expressed separately from the structural HTML code, would give authors more control over the visual display of their content.

In 1994, Hakon Lie of Conseil Européen pour la Recherche Nucléaire (CERN), the European Laboratory for Particle Physics, released a proposal for the Cascading Style Sheet Language. This style language contained a number of common design characteristics and included the ability to apply a number of different style sheets—a “cascade” of styles—to the same document. The ability to cascade multiple style sheets, and to determine which sheets and styles take precedence, is a fundamental feature of CSS. Applying multiple style sheets to a document means that users can supply their own style sheets that override the author’s style sheet. The novel idea of allowing the users to supply their own style sheets lets users adapt content to their own preferences. For example, users with sight disabilities could add a style sheet that increases the size of text to make it more legible. You will read more about applying multiple style sheets to a document later in this chapter.

A few days after the release of Lie’s proposal, the first beta version of Netscape Navigator, nickname Mozilla, was released. Among other enhancements, the Netscape browser supported the <CENTER> tag, a new element that was not contained in the HTML specification. Elements like <CENTER> change HTML from a simple document structure language into a language that displays both style and structure. As you see later in the chapter, this mixing of style and structure can cause significant compatibility problems for the future of the Web.

Netscape continued to add more elements, including <FONT>, to the HTML mix, hoping to garner a larger market share for their browser. Netscape reasoned that if they gave HTML authors the display control they wanted, the authors would build Web pages that favored the Netscape browser. These new elements quickly became so popular and pervasive that competing products such as Internet Explorer had to support them or cease being viable as a browser.

As the Web continued to evolve, more factors contributed to the loosening of the structural nature of HTML. The browser war between Netscape and Microsoft started to heat up, and each company introduced proprietary elements. These elements, such as <MARQUEE> (Internet Explorer only) and <BLINK> (Netscape only), work only within the browser for which they were designed. Using proprietary elements like these defeat the open, portable nature of HTML. Also, the advent of HTML table elements brought a new level of complexity to HTML code. Although originally intended for tabular data, Web designers quickly realized that they could build print-like design structures that allowed them to break away from the left-alignment constraints of basic HTML. Unfortunately, this method of coding (still the predominate method of coding on the Web today) meant that HTML code had evolved into a mixture of display and structure information. Web pages had become hard-coded for a specific destination media, the computer screen. Display information that is designed for only one medium severely limits the future portability of Web-based content, which now needs to be viewed on a variety of different devices, as you will see later in this section.
Battling against this increasing fracturing of HTML was the newly founded World Wide Web Consortium (W3C). Founded in 1994, the W3C sets standards for HTML and provides an open, nonproprietary forum for industry leaders and academics to add to the evolution of this new medium. In 1996, the W3C released the first completed specification for Cascading Style Sheets—CSS1. CSS1 consisted of an easy-to-use, declarative syntax that used familiar typesetting and design terminology and allowed HTML authors much more expressive control over the appearance of their Web pages. The only problem was that there was not a single browser that supported CSS.

**Browser Support for CSS**

Microsoft Internet Explorer 3.0 was the first browser to offer rudimentary support for CSS1. As Internet Explorer evolved, it continued to provide increasing support for CSS, although it did not fully support style sheets until release 5.0 for the Macintosh in 1999. Netscape has resisted CSS, supporting only the most basic features through release 4.75. Figures 1-2 through 1-5 illustrate the difference in support between browsers. Figure 1-2 shows an HTML file styled with CSS style rules in Microsoft Internet Explorer 5.5. This page includes a number of CSS properties, including specifying font characteristics, alignment, margins, and backgrounds. You will learn how to write the style rules for this page in a later chapter.
In contrast, Figure 1-3 illustrates the poor support for CSS in Netscape Navigator 4.75. The reluctance of Netscape to support the CSS standard has been the single most important limiting factor in the acceptance of CSS among HTML authors, who were reluctant to address the complexity of cross-browser compatibility issues.
Netscape finally gave in to the CSS trend, and, as a result, Netscape 6 offers excellent support for CSS. Figure 1-4 shows the same page in Netscape Navigator 6 and the great strides Netscape has made in supporting CSS.

Figure 1-5 shows the sample page in Opera, the popular browser from Norway. This browser, now in release 5.0, also contains good support for CSS properties.

Four years after its initial release (a long time at the current pace of Web development), CSS is ready for prime time. HTML authors who have been clamoring for page layout control since the inception of the Web now have a powerful style language at their disposal. The W3C has continued the development of CSS, and in 1998 CSS2 was released. CSS2 incorporates many new features while retaining most of the original CSS specification. You will read more about CSS2's enhancements later in this chapter.
Browser Support for CSS

Figure 1-4  A CSS sample page in Netscape Navigator 6.0
SEPARATING STYLE FROM STRUCTURE

View the source code for any page on the Web, and you see a jumbled hodgepodge of structural HTML elements, such as `<H1>`, `<P>`, and `<DIV>`, mixed with visual elements, such as `<FONT>`, `<TABLE>`, and `<CENTER>`. Adding to the general confusion are a number of HTML attributes whose sole purpose is to provide visual display information to the browser, such as `BGCOLOR`, `FACE`, and `ALIGN`. Although this may not seem to be a problem, consider how much code on the average Web page is devoted to structural information and how much to display instructions. In many cases, the display information is more than double the amount of code. Not only does the additional code add to the complexity of the HTML, display instructions are a very inefficient way to handle the display characteristics of multipage Web sites. For example, assume that you want all of your `<H1>` headings to appear green and centered everywhere on your Web site. For every instance of an `<H1>` element, you would have to include the following code in each HTML document:

```html
<H1 ALIGN="CENTER"><FONT COLOR="GREEN">Some Heading Text</FONT></H1>
```
Using a single CSS rule, you can express the same style this way:

\[
\text{H1 \{color: green; text-align: center;\}}
\]

If you place this rule in a single document, every <H1> element in that document will be green and centered. If you place this rule in an external file called a style sheet, and then connect every page on your Web site to that style sheet (explained in detail later in this chapter), then every <H1> element in your Web site will be green and centered. With a minimum of code, you’ve expressed the same result. Later, if you want to change the <H1> color to red, you simply change the style sheet rule to change every page on your site, resulting in easier code maintenance.

Another equally important reason to separate the style information from the structure of the document is that the mixture of this information limits the cross-platform compatibility of the content. The display information embedded in Web pages is tailored toward one type of display medium, the computer screen. With style sheets, the display properties are separate from the content. This accommodates the diversity of devices that are becoming available to browse the Web. Whether you come to the Web with a Personal Digital Assistant (PDA), a Personal Communication Services (PCS) telephone, or Windows CE device, the Web server can determine the type of requesting device and supply a style sheet that matches the device. Additionally, CSS2 supports a variety of media types, including print, handheld devices, and Braille, that allows content providers to use a single source for their data regardless of the number of destination formats, as illustrated in Figure 1-6.
SECTION 2: CSS BASICS

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 W3C (http://www.w3.org) maintains the specifications that define CSS. Currently, there are two versions of CSS, Release 1 (CSS1) and Release 2 (CSS2). These are also called CSS Level 1 and CSS Level 2, respectively.

CSS1 and CSS2

CSS1 was released in December 1996. Internet Explorer 5.5, Netscape 6, and Opera 5 offer good to excellent support for CSS1. CSS2 was released in May 1998. At the time of this writing, no browsers fully support CSS2, but some of the new properties are variously supported by the browsers listed earlier. As always, you must test your work in multiple browsers to ensure the compatibility of your code.

Some of the more important new features in CSS2 include the following:

- **Media types** - Allows one document to contain style rules for different types of destination media, described in Appendix B
- **Paged media support** - Allows formatting of page breaks and margins when the destination media is printed rather than viewed on the Web, described in Appendix B
- **Aural style sheets** – Provides capabilities for text-to-speech devices, speech synthesis, and audio texts, described in Appendix B
- **Table properties** - Presents table information visually for display devices or aurally for speech synthesis, described in the “Working with Tables and Lists” chapter
- **New selection techniques** - Includes new ways to classify and apply rules to elements in an HTML document. Chapter 2 examines selectors and selection techniques in detail.
- **Display enhancements** - Includes new color, font, and border properties
- **Generated content** - Allows generation of content that is not supplied in the HTML source document, such as automatic numbering of chapters and lists, described in Appendix B
- **Enhanced positioning schemes** - Allows exact positioning of objects on the page without tables or transparent images. Chapter 7 examines the positioning properties in detail.

This book focuses on the CSS2 properties that are supported by the browsers and can be used today. CSS2 features that are not widely supported by the browsers are described in the appendices.

**Understanding Style Rules**

In CSS, **style rules** express the style characteristics for an HTML 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. As you will see in the next chapter, CSS contains a variety of powerful selection techniques. The declaration details the exact property values. Figure 1-7 shows an example of a simple style rule that sets all <H1> headings to red.

```
H1 {color: red;}
```

**Figure 1-7** Style rule syntax

As illustrated in Figure 1-8, 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, 12pt (point) for font-size, or 30px (pixels) for margin, followed by a semicolon (;). CSS contains a wide variety of properties, each with a specific list of values.

```
H1 {color: red;}
```

**Figure 1-8** Property declaration syntax

The style rule in Figure 1-8 is a basic example of a style rule. As you will see in Chapter 2, you can combine selectors and property declarations in a variety of ways.

**Combining CSS Rules with HTML**

You can combine CSS rules with HTML code in the following three ways. Each method is discussed in detail in the following sections.

- The STYLE attribute
- The <STYLE> element
- An external style sheet
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 only affects 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 only affect the document in which they reside. The following code shows a <STYLE> element that contains a single style rule:

```html
<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 different HTML documents. 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:

```css
H1 {color: white; background-color: green;}
H2 {color: red;}
```

The style sheet file does not contain any HTML code, just CSS style rules, because the style sheet is not an HTML 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 only be used within the <HEAD> section of a document. To link to an external style sheet, add the <LINK> element as shown in the following code:

```html
<HEAD>
```

Copyright © 2001 by Course Technology. All rights reserved. This publication is protected by federal copyright law. No part of this publication may be reproduced without prior permission in writing from Course Technology. Some of the product names and company names have been used for identification purposes only and may be trademarks or registered trademarks of their respective manufacturers and sellers.
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 HTML file 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 only have to change the style rule 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:

```html
<style type="text/css">
/* This is the basic style sheet */
H1 {color: grey;} /* 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.

### Combining Multiple Style Sheets

The `@import` keyword lets you import style rules from other style sheets. Multiple `@import` statements are allowed, letting you combine multiple style sheets for one document. Any `@import` rules must precede all rules in a style sheet or they will be ignored by the browser. The `@import` keyword must be followed by the URL of the style sheet you want to include, as shown in the following code:

```html
<style type="text/css">
@import "basic.css";
@import "enhanced.css";
H1 {color: white; background: green;}
H2 {color: red;}
</style>
```
The CSS cascading mechanism (described later in this chapter) resolves conflicts between style sheets imported with the @import keyword. The style rules contained within the document always take precedence over imported style rules. Additionally, the weight of the imported style sheets is based on its import order. In the previous example, the internal style rules for the <H1> and <H2> elements take precedence over competing rules from the imported style sheets. Any rules stated in the basic.css style sheet take precedence over competing rules in enhanced.css because of the import order.

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 1-10 as you progress through the steps to see the results you will achieve.

To build a basic style sheet:

1. Open the file basic.htm in your HTML editor and save it in your work folder as basic1.htm.

2. In your browser, open the file basic1.htm. When you open the file, it looks like Figure 1-9.

---

**CSS Basics**

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 (http://www.w3.org) maintains the specifications that define CSS. Currently, there are two versions of CSS, Release 1 (CSS1) and Release 2 (CSS2).
3. Examine the code. Notice that the file contains basic HTML code with no style information. The complete code for the page follows:

```html
<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>
```

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.

```html
<HEAD>
<TITLE>A Basic Document</TITLE>
<STYLE type="text/css">

</STYLE>
</HEAD>
```

5. Add a style rule for the `<H1>` element as shown in the following code fragment. This style rule uses the text-align property to center the heading.

```html
<HEAD>
<TITLE>A Basic Document</TITLE>
<STYLE type="text/css">
H1 {text-align: center;}
</STYLE>
</HEAD>
```

6. Save the file `basic1.htm`, and then reload it in the browser. The `<H1>` element is now centered, as shown in Figure 1-10.

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.
8. Save the file `basic1.htm`, and then reload it in the browser. Figure 1-10 shows the finished Web page. Notice that `<P>` element is now displayed in a sans-serif typeface.

![CSS Basics](image)

**CSS Basics**

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 (http://www.w3.org) maintains the specifications that define CSS. Currently, there are two versions of CSS, Release 1 (CSS1) and Release 2 (CSS2).

**Figure 1-10** The HTML document styled with CSS

### Linking to an External Style Sheet

In this set of steps, you will link an HTML file to an external style sheet that contains style rules. You can just as easily link multiple files to the same style sheet using the syntax you will learn here.

Tip

Make sure that both `basic2.htm` and `basic2.css` are located in the same directory folder for this procedure to work properly.
1. Open the file basic2.htm in your browser. When you open the file it looks like Figure 1-11.

CSS1 and CSS2

Cascading Style Sheets Level 1 (CSS1) was released in December 1996. Internet Explorer 5.5, Netscape 6, and Opera 5 offer good to excellent support for CSS1. Cascading Style Sheets Level 2 (CSS2) was released in May 1998. At the time of this writing, no browsers fully support CSS2, but some of the new properties are variously supported by the browsers listed above. As always, you must test your work in multiple browsers to ensure the compatibility of your code.

Figure 1-11  The original HTML document

2. Open the basic2.htm file in your HTML editor, and examine the code. Notice that the file contains basic HTML code with no style information. The complete code for the page follows:

```html
<HTML>
<HEAD>
<TITLE>A Basic Document</TITLE>
</HEAD>
<BODY>
<H1>CSS1 and CSS2</H1>
<P>Cascading Style Sheets Level 1 (CSS1) was released in December 1996. Internet Explorer 5.5, Netscape 6, and Opera 5 offer good to excellent support for CSS1. Cascading Style Sheets Level 2 (CSS2) was released in May 1998. At the time of this writing, no browsers fully support CSS2, but some of the new properties are variously supported by the browsers listed above. As always, you must test your work in multiple browsers to ensure the compatibility of your code.</P>
</BODY>
</HTML>
```
3. Add the <LINK> element within the <HEAD> element, as shown in the following code fragment. The HREF attribute specifies the location of the style sheet file. The REL attribute specifies the resource as a style sheet.

```html
<HEAD>
<TITLE>A Basic Document</TITLE>
<LINK HREF="basic2.css" REL="stylesheet">
</HEAD>
```

4. Save the file as **basic2.htm** in your work folder, and then view it in the browser. The Web page now displays the style characteristics specified in the external style sheet, as shown in Figure 1-12.

![CSS1 and CSS2](image)

**Figure 1-12** The HTML document linked to an external style sheet

5. Open the style sheet file **basic2.css** in your HTML editor and examine the style rules. The complete style sheet code follows:

```css
/* This is the style sheet for the basic2.htm file */
H1 {font-family: sans-serif; border-bottom: solid 1px;}
P {font-family: sans-serif; margin-left: 30px;}
```

No code other than the style rules are necessary in an external style sheet. As you can see, the style rules are easy to interpret, even for someone who is not familiar 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. HTML 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 sheets 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 towards 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 HTML document:

```css
<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.

In CSS1, !important in an author’s style sheet took precedence over !important in a user’s style sheet. In CSS2, the reverse is true, giving the user more control over display properties.

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 HTML author, the user, and the browser. Figure 1-13 shows the style sheet 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 own 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 hard 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:

```css
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:

```css
BODY {color: black;}
H1 {color: red;}
H1 {color: green;}
```
In this example, <H1> elements in the document will appear green because of the last style rule which specifies green as the color.

**Understanding Inheritance**

The elements in an HTML document are structured in a hierarchy of parent and child elements. Figure 1-14 represents the structure of a simple HTML document.

```
<HTML>
  <HEAD>
    <TITLE>
  </HEAD>
  <BODY>
    <H1>
    <UL>
      <LI>
      <P>
      <EM>
```

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 inherit from parent elements to child elements. Therefore, if you set a style rule for <UL> elements in the document shown in Figure 1-14, the style rules inherit to the <LI> elements, 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 the original document shown in Figure 1-9:

```
<STYLE TYPE="text/css">
H1 {color: red;}
P {color: red;}
UL {color: red;}
EM {color: red;}
LI {color: red;}
```
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:

```html
<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. `<BODY>` is the parent element of the content area of the HTML file; therefore, it is the selector to use whenever you want to apply a style across the entire document.

## Chapter Summary

This chapter presents the history of CSS development, the need to separate HTML structure from CSS style information, and the basic syntax of the CSS language. CSS has traveled a long road, but it is finally ready for widespread use on the Web. 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 evolved from the popularity of the Web as a publishing medium and the demand from HTML authors who wanted common desktop publishing controls over the display of their Web content.
- CSS was poorly supported by browsers at first but now is becoming widely supported.
- CSS supports the separation of style from structure, allowing repurposing of a single source of data to multiple destination media.
- CSS rules can be combined with your HTML in a number of ways. CSS rules are easy to write and read.
- You can combine multiple style sheets using the `@import` keyword.
- CSS uses cascading and inheritance to determine which style rules take precedence. The `!important` declaration lets users override the author’s style rules.

## Review Questions

1. Who is recognized as the creator of HTML?
2. HTML is a subset of which markup language?
3. How did Berners-Lee attempt to make HTML as portable as possible?
4. What are the benefits of allowing the same document to have multiple style sheets?
5. What are two of the style elements introduced by Netscape?
6. What is a proprietary element?
7. What are two examples of proprietary elements?
8. What is the purpose of the World Wide Web Consortium?
9. What was the first browser to support CSS?
10. What was the single most important limiting factor in the acceptance of CSS?
11. What are two benefits of separating style from structure?
12. List and describe three enhancements in CSS2.
13. What are the two parts of a style rule?
14. What are the three ways to combine CSS rules with your HTML code?
15. List two reasons to state a style using the STYLE attribute.
16. What are the advantages of using an external style sheet?
17. What keyword would you use to import style rules from other style sheets?
18. What is the inheritance default for CSS rules?
19. What is the benefit of the !important declaration?

**Hands-On Projects**

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 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. View and copy the source code from a Web page of your choice into your HTML editor. Examine the code for existing display elements (such as <FONT>) and attributes (such as BGCOLOR, ALIGN). Write a paper describing how CSS properties could replace the existing display information, detailing the benefits of CSS. Refer to Appendix A for a listing of CSS properties. Include the HTML code from the Web page with your report.
5. In this project, you will have a chance to test a few simple style rules on a standard HTML document and to view the results in your browser.

a. Using your HTML editor, create a simple HTML file (or open an existing file) that contains <BODY>, <H1>, and <P> elements. Add some text and an <H1> heading. Save the file as csstest1.htm in your work folder.

b. Add a <STYLE> element to the <HEAD> section, as shown in the following code:

```html
<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:

```html
<STYLE TYPE="text/css">
BODY {color: green;}
</STYLE>
```

d. Save the file, 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 display in black:

```html
<STYLE TYPE="text/css">
BODY {color: green;}
H1 {color: black;}
</STYLE>
```

f. Save the file as 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:

```html
<STYLE TYPE="text/css">
BODY {color: green;}
H1 {color: black;}
P {margin: 30px;}
</STYLE>
```

h. Save the file, and view the results in the browser.

---

**Case Study**

To complete the ongoing case study for this book, you must create a complete stand-alone Web site that uses CSS properties to control all design characteristics of the site. Your instructor may choose to make this either an individual or team project. You can choose your own content for the Web site. For example, you can build a personal interest site, a site for your favorite nonprofit organization, or a work-related topic. At the end of each chapter you will complete a different facet of the project, each contributing to the finished Web site. For this chapter, you will get started by creating a project proposal, as outlined below.
Project Proposal

Create a one-or-two page basic HTML document that states the basic considerations for your project Web site. This is a preliminary draft that you will hand in to your instructor and possibly review with other members of the class. You will have a chance to modify the draft and resubmit a more finished effort at the end of the next chapter.

Include the following items, if applicable.

- **Site title** – The working title of the site
- **Developer** – You and anyone else who will work on the site
- **Content and goals** – What type of content will the site contain? What are the broad goals of the site? Provide an outline of the content topics.
- **Target audience** – Describe the typical audience for the site
- **Style considerations** – Describe the style requirements for the site and how you hope to fill these needs with CSS. What will be your font and color choices, and why? How will your design aid the user? Remember that this is a proposal and you will get to modify these choices as you work through the case study.
- **Site map** – Sketch out a preliminary map of the Web site, indicating individual file names and how the files will be linked together.
- **Limiting factors** – List the technical, audience, or development factors that could limit the design goals of the site.