**Chapter 3. Understanding Cascading Style Sheets**

WHAT YOU’LL LEARN IN THIS CHAPTER:

• How to create a basic style sheet

• How to use style classes

• How to use style IDs

• How to construct internal style sheets and inline styles

In the previous chapter, you learned the basics of HTML and XHTML, including how to set up a

skeletal HTML template for all your web content. In this chapter, you will learn how to fine-tune the

display of your web content using *Cascading Style Sheets (CSS).*

The concept behind style sheets is simple: You create a style sheet document that specifies the fonts,

colors, spacing, and other characteristics that establish a unique look for a website. You then link

every page that should have that look to the style sheet, instead of specifying all those styles

repeatedly in each separate document. Therefore, when you decide to change your official corporate

typeface or color scheme, you can modify all your web pages at once just by changing one or two

entries in your style sheet rather than changing them in all of your static web files. So, a *style sheet* is

a grouping of formatting instructions that controls the appearance of several HTML pages at once.

Style sheets enable you to set a great number of formatting characteristics, including exacting typeface

controls, letter and line spacing, and margins and page borders, just to name a few. Style sheets also

enable sizes and other measurements to be specified in familiar units, such as inches, millimeters,

points, and picas. You can also use style sheets to precisely position graphics and text anywhere on a

web page, either at specific coordinates or relative to other items on the page.

**Note**

If you have three or more web pages that share (or should share) similar formatting and fonts, you might

want to create a style sheet for them as you read this chapter. Even if you choose not to create a

complete style sheet, you’ll find it helpful to apply styles to individual HTML elements directly within a

web page.

In short, style sheets bring a sophisticated level of display to the Web. And they do so—you’ll pardon

the expression—with style.

**How CSS Works**

The technology behind style sheets is called CSS, which stands for Cascading Style Sheets. CSS is a

language that defines style constructs such as fonts, colors, and positioning, which are used to

describe how information on a web page is formatted and displayed. CSS styles can be stored

directly in an HTML web page or in a separate style sheet file. Either way, style sheets contain style

rules that apply styles to elements of a given type. When used externally, style sheet rules are placed

in an external style sheet document with the file extension .css.

A *style rule* is a formatting instruction that can be applied to an element on a web page, such as a

paragraph of text or a link. Style rules consist of one or more *style properties* and their associated

values. An *internal style sheet* is placed directly within a web page, whereas an *external style sheet*

exists in a separate document and is simply linked to a web page via a special tag—more on this tag

in a moment.

The *cascading* part of the name CSS refers to the manner in which style sheet rules are applied to

elements in an HTML document. More specifically, styles in a CSS style sheet form a hierarchy in

which more specific styles override more general styles. It is the responsibility of CSS to determine

the precedence of style rules according to this hierarchy, which establishes a cascading effect. If that

sounds a bit confusing, just think of the cascading mechanism in CSS as being similar to genetic

inheritance, in which general traits are passed from parents to a child, but more specific traits are

entirely unique to the child. Base style rules are applied throughout a style sheet but can be

overridden by more specific style rules.

**Note**

You might notice that I use the term *element* a fair amount in this chapter (and I will for the rest of the

book, for that matter). An *element* is simply a piece of information (content) in a web page, such as an

image, a paragraph, or a link. Tags are used to code elements, and you can think of an element as a tag

complete with descriptive information (attributes, text, images, and so on) within the tag.

A quick example should clear things up. Take a look at the following code to see whether you can tell

what’s going on with the color of the text:

<div style="color:green">

This text is green.

<p style="color:blue">This text is blue.</p>

<p>This text is still green.</p>

</div>

In the previous example, the color green is applied to the <div> tag via the color style property. Therefore,

the text in the <div> tag is colored green. Because both <p> tags are children of the <div> tag, the green text

style cascades down to them. However, the first <p> tag overrides the color style and changes it to

blue. The end result is that the first line (not surrounded by a paragraph tag) is green, the first official

paragraph is blue, and the second official paragraph retains the cascaded green color.

If you made it through that description on your own, congratulations. If you understood it after I

explained it in the text, congratulations to you as well. Understanding CSS isn’t like understanding

rocket science, although many people will try to convince you that it is (so that they can charge high

consultation fees, most likely!).

Like many web technologies, CSS has evolved over the years. The original version of CSS, known as

*Cascading Style Sheets Level 1* (*CSS1)* was created in 1996. The later CSS 2 standard was created

in 1998, and CSS 2 is still in use today. All modern web browsers support CSS 2, and you can safely

use CSS 2 style sheets without too much concern. So when I talk about CSS throughout the book, I’m

referring to CSS 2.

You’ll find a complete reference guide to CSS at http://www.w3.org/Style/CSS/. The rest of this

chapter explains how to put CSS to good use.

**A Basic Style Sheet**

Despite their intimidating power, style sheets can be simple to create. Consider the web pages shown

in Figure 3.1 and Figure 3.2. These pages share several visual properties that could be put into a

common style sheet:

• They use a large, bold Verdana font for the headings and a normal size and weight Verdana font

for the body text.

• They use an image named logo.gif floating within the content and on the right side of the page.

• All text is black except for subheadings, which are purple.

• They have margins on the left side and at the top.

• There is vertical space between lines of text.

• The footnotes are centered and in small print.

**Figure 3.1** This page uses a style sheet to fine-tune the appearance and spacing of the text and images.

**Figure 3.2** This page uses the same style sheet as the one shown in Figure 3.1, thus maintaining a consistent look and feel.

Example 3.1 shows the code for the style sheet specifying these properties.

Example 3.1 A Single External Style Sheet

body {

font-size: 10pt;

font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;

color: black;

line-height: 14pt;

padding-left: 5pt;

padding-right: 5pt;

padding-top: 5pt;

}

h1 {

font: 14pt Verdana, Geneva, Arial, Helvetica, sans-serif;

font-weight: bold;

line-height: 20pt;

}

p.subheader {

font-weight: bold;

color: #593d87;

}

img {

padding: 3pt;

float: right;

}

a {

text-decoration: none;

}

a:link, a:visited {

color: #8094d6;

}

a:hover, a:active {

color: #FF9933;

}

div.footer {

font-size: 9pt;

font-style: italic;

line-height: 12pt;

text-align: center;

padding-top: 30pt;

}

This might initially appear to be a lot of code, but if you look closely, you’ll see that there isn’t a lot

of information on each line of code. It’s fairly standard to place individual style rules on their own

line to help make style sheets more readable, but that is a personal preference; you could put all the

rules on one line as long as you kept using the semicolon to separate each rule (more on that in a bit).

Speaking of code readability, perhaps the first thing you noticed about this style sheet code is that it

doesn’t look anything like normal HTML code. CSS uses a language all its own to specify style

sheets.

Of course, the example includes some familiar HTML tags. As you might guess, body, h1, p, img, a, and div in

the style sheet refer to the corresponding tags in the HTML documents to which the style sheet will be

applied. The curly braces after each tag name contain the specifications for how all content within

that tag should appear.

In this case, the style sheet says that all body text should be rendered at a size of 10 points, in the

Verdana font (if possible), with the color black, and 14 points between lines. If the user does not have

the Verdana font installed, the list of fonts in the style sheet represents the order in which the browser

should search for fonts to use: Geneva, then Arial, and then Helvetica. If the user has none of those

fonts, the browser will use whatever default sans serif font is available. Additionally, the page should

have left, right, and top margins of 5 points each.

Any text within an <h1> tag should be rendered in boldface Verdana at a size of 14 points. Moving on,

any paragraph that uses only the <p> tag will inherit all the styles indicated by the body element.

However, if the <p> tag uses a special class named subheader, the text will appear bold and in the color

#593d87 (a purple color).

The pt after each measurement in Example 3.1 means *points* (there are 72 points in an inch). If you

prefer, you can specify any style sheet measurement in inches (in), centimeters (cm), pixels (px), or

widths-of-a-letter-m, which are called ems (em).

**Note**

You can specify font sizes as large as you like with style sheets, although some display devices and

printers will not correctly handle fonts larger than 200 points.

You might have noticed that each style rule in the example ends with a semicolon (;). Semicolons are

used to separate style rules from each other. It is therefore customary to end each style rule with a

semicolon, so you can easily add another style rule after it.

To link this style sheet to HTML documents, include a <link /> tag in the <head> section of each document.

Example 3.2 shows the HTML code for the page shown in Figure 3.1. It contains the following <link />

tag:

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

This assumes that the style sheet is stored under the name styles.css in the same folder as the HTML

document. As long as the web browser supports style sheets—and all modern browsers do support

style sheets—the properties specified in the style sheet will apply to the content in the page without

the need for any special HTML formatting code. This confirms the ultimate goal of XHTML, which is

to provide a separation between the content in a web page and the specific formatting required to

display that content.

Example 3.2 HTML Code for the Page Shown in Figure 3.1

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<title>About BAWSI</title>

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

</head>

<body>

<h1>About BAWSI</h1>

<p><img src="logo.gif" alt="BAWSI logo"/>The Bay Area Women's

Sports Initiative (BAWSI) is a public benefit, nonprofit

corporation with a mission to create programs and partnerships

through which women athletes bring health, hope and wholeness to

our community. Founded in 2005 by Olympic and World Cup soccer

stars Brandi Chastain and Julie Foudy and Marlene Bjornsrud,

former general manager of the San Jose CyberRays women's

professional soccer team, BAWSI provides a meaningful path for

women athletes to become a more visible and valued part of the

Bay Area sports culture.</p>

<p class="subheader">BAWSI's History</p>

<p>The concept of BAWSI was inspired by one of the most

spectacular achievements in women's sports history and born out

of one its biggest disappointments... </p>

<p><a href="secondpage.html">[continue reading]</a></p>

<div class="footer">Copyright &copy; 2005-2009 BAWSI

(www.bawsi.org). All rights reserved. Used with permission.</div>

</body>

</html>

**Tip**

In most web browsers, you can view the style rules in a style sheet by opening the .css file and choosing

Notepad or another text editor as the helper application to view the file. (To determine the name of the

.css file, look at the HTML source of any web page that links to it.) To edit your own style sheets, just

use a text editor.

**Note**

Although CSS is widely supported in all modern web browsers, it hasn’t always enjoyed such wide

support. Additionally, not every browser’s support of CSS is flawless. To find out about how major

browsers compare to each other in terms of CSS support, take a look at this website:

http://www.quirksmode.org/css/contents.html.

The code in Example 3.2 is interesting because it contains no formatting of any kind. In other words,

there is nothing in the HTML code that dictates how the text and images are to be displayed—no

colors, no fonts, nothing. Yet the page is carefully formatted and rendered to the screen, thanks to the

link to the external style sheet, styles.css. The real benefit to this approach is that you can easily

create a site with multiple pages that maintains a consistent look and feel. And you have the benefit of

isolating the visual style of the page to a single document (the style sheet) so that one change impacts

all pages.

**Try It Yourself: Create a Style Sheet of Your Own**

Starting from scratch, create a new text document called mystyles.css and add some style rules for the following basic HTML

tags: <body>, <p>, <h1>, and <h2>. After your style sheet has been created, make a new HTML file that contains these basic tags.

Play around with different style rules and see for yourself how simple it is to change entire blocks of text in paragraphs with one

simple change in a style sheet file.