**A CSS Style Primer**

You now have a basic knowledge of CSS style sheets and how they are based on style rules that

describe the appearance of information in web pages. The next few sections of this chapter provide a

quick overview of some of the most important style properties and allow you to get started using CSS

in your own style sheets.

CSS includes various style properties that are used to control fonts, colors, alignment, and margins, to

name just a few. The style properties in CSS can be generally grouped into two major categories:

• **Layout properties**—Consist of properties that affect the positioning of elements on a web page,

such as margins, padding, alignment, and so on

• **Formatting properties**—Consist of properties that affect the visual display of elements within a

website, such as the font type, size, color, and so on

**Layout Properties**

CSS layout properties are used to determine how content is placed on a web page. One of the most

important layout properties is the display property, which describes how an element is displayed with

respect to other elements. There are four possible values for the display property:

• **block**—The element is displayed on a new line, as in a new paragraph.

**Note**

The display property relies on a concept known as *relative positioning*, which means that elements are

positioned relative to the location of other elements on a page. CSS also supports *absolute positioning*,

which enables you to place an element at an exact location on a page independent of other elements.

You’ll learn more about both of these types of positioning in Part III, “Advanced Web Page Design with

CSS.”

• **list-item**—The element is displayed on a new line with a list-item mark (bullet) next to it.

• **inline**—The element is displayed inline with the current paragraph.

• **none**—The element is not displayed; it is hidden.

It’s easier to understand the display property if you visualize each element on a web page occupying a

rectangular area when displayed—the display property controls the manner in which this rectangular

area is displayed. For example, the block value results in the element being placed on a new line by

itself, whereas the inline value places the element next to the content just before it. The display property is

one of the few style properties that can be applied in most style rules. Following is an example of

how to set the display property:

display:block;

You control the size of the rectangular area for an element with the width and height properties. Like many

size-related CSS properties, width and height property values can be specified in several different units of

measurement:

• **in**—Inches

• **cm**—Centimeters

• **mm**—Millimeters

• **px**—Pixels

• **pt**—Points

You can mix and match units however you choose within a style sheet, but it’s generally a good idea

to be consistent across a set of similar style properties. For example, you might want to stick with

points for font properties or pixels for dimensions. Following is an example of setting the width of an

element using pixel units:

width: 200px;

**Formatting Properties**

CSS formatting properties are used to control the appearance of content on a web page, as opposed to

controlling the physical positioning of the content. One of the most popular formatting properties is

the border property, which is used to establish a visible boundary around an element with a box or

partial box. The following border properties provide a means of describing the borders of an element:

• **border-width**—The width of the border edge

• **border-color**—The color of the border edge

• **border-style**—The style of the border edge

• **border-left**—The left side of the border

• **border-right**—The right side of the border

• **border-top**—The top of the border

• **border-bottom**—The bottom of the border

• **border**—All the border sides

The border-width property is used to establish the width of the border edge. It is often expressed in pixels,

as the following code demonstrates:

border-width:5px;

Not surprisingly, the border-color and border-style properties are used to set the border color and style.

Following is an example of how these two properties are set:

border-color:blue;

border-style:dotted;

The border-style property can be set to any of the following values:

• **solid**—A single-line border

• **double**—A double-line border

• **dashed**—A dashed border

• **dotted**—A dotted border

• **groove**—A border with a groove appearance

• **ridge**—A border with a ridge appearance

• **inset**—A border with an inset appearance

• **outset**—A border with an outset appearance

• **none**—No border

The default value of the border-style property is none, which is why elements don’t have a border unless you

set the border property to a different style. The most common border styles are the solid and double styles.

**Note**

The exception to the default border-style of none is when an image is placed within an <a> tag so that it

serves as a linked image. In that case, a solid border is automatically set by default. That’s why you

often see linked images with the style border-style:none, which turns off the automatic border.

The border-left, border-right, border-top, and border-bottom properties enable you to set the border for each side of an

element individually. If you want a border to appear the same on all four sides, you can use the single

border property by itself, which expects the following styles separated by a space: border-width, border-style, and

border-color. Following is an example of using the border property to set a border that consists of two

(double) red lines that are a total of 10 pixels in width:

border:10px double red;

Whereas the color of an element’s border is set with the border-color property, the color of the inner

region of an element is set using the color and background-color properties. The color property sets the color of

text in an element (foreground) and the background-color property sets the color of the background behind the

text. Following is an example of setting both color properties to predefined colors:

color:black;

background-color:orange;

You can also assign custom colors to these properties by specifying the colors in hexadecimal

(covered in more detail in Chapter 8, “Working with Colors, Images, and Multimedia”) or as RGB

(Red, Green, Blue) decimal values, just as you do in HTML:

background-color:#999999;

color:rgb(0,0,255);

You can also control the alignment and indentation of web page content without too much trouble.

This is accomplished with the text-align and text-indent properties, as the following code demonstrates:

text-align:center;

text-indent:12px;

After you have an element properly aligned and indented, you might be interested in setting its font.

The following font properties are used to set the various parameters associated with fonts:

• **font-family**—The family of the font

• **font-size**—The size of the font

• **font-style**—The style of the font (normal or italic)

• **font-weight**—The weight of the font (light, medium, bold, and so on)

The font-family property specifies a prioritized list of font family names. A prioritized list is used instead

of a single value to provide alternatives in case a font isn’t available on a given system. The font-size

property specifies the size of the font using a unit of measurement, usually points. Finally, the font-style

property sets the style of the font and the font-weight property sets the weight of the font. Following is an

example of setting these font properties:

font-family: Arial, sans-serif;

font-size: 36pt;

font-style: italic;

font-weight: medium;

Now that you know a whole lot more about style properties and how they work, refer back at Example

3.1 and see whether it makes a bit more sense. Here’s a recap of the style properties used in that style

sheet, which you can use as a guide for understanding how it works:

• **font**—Lets you set many font properties at once. You can specify a list of font names separated by

commas; if the first is not available, the next is tried, and so on. You can also include the words bold

and/or italic and a font size. Each of these font properties can be specified separately with font-family,

font-size, font-weight, and font-style if you prefer.

• **line-height**—Also known in the publishing world as *leading*. This sets the height of each line of text,

usually in points.

• **color**—Sets the text color using the standard color names or hexadecimal color codes (see Chapter

8 for more details).

• **text-decoration**—Useful for turning link underlining off—simply set it to none. The values of underline, italic,

and line-through are also supported. The application of styles to links is covered in more detail in

Chapter 7, “Using External and Internal Links.”

• **text-align**—Aligns text to the left, right, or center, along with justifying the text with a value of justify.

• **padding**—Adds padding to the left, right, top, and bottom of an element; this padding can be in

measurement units or a percentage of the page width. Use padding-left and padding-right if you want to add

padding to the left and right of the element independently. Use padding-top or padding-bottom to add padding

to the top or bottom of the element, as appropriate. You’ll learn more about these style properties

in Chapters 9, “Working with Margins, Padding, Alignment, and Floating,” and 10, “Understanding

the CSS Box Model and Positioning.”

**Using Style Classes**

This is a “teach yourself” book, so you don’t have to go to a single class to learn how to give your

pages great style, although you do need to learn what a style class is. Whenever you want some of the

text on your pages to look different from the other text, you can create what amounts to a custom-built

HTML tag. Each type of specially formatted text you define is called a *style class*. A *style class* is a

custom set of formatting specifications that can be applied to any element in a web page.

Before showing you a style class, I need to take a quick step back and clarify some CSS terminology.

First off, a CSS *style property* is a specific style that can be assigned a value, such as color or font-size.

You associate a style property and its respective value with elements on a web page by using a

selector. A *selector* is used to identify tags on a page to which you apply styles. Following is an

example of a selector, a property, and a value all included in a basic style rule:

h1 { font: 36pt Courier; }

In this code, h1 is the selector, font is the style property, and 36pt Courier is the value. The selector is

important because it means that the font setting will be applied to all h1 elements in the web page. But

maybe you want to differentiate between some of the h1 elements—what then? The answer lies in style

classes.

Suppose you want two different kinds of <h1> headings for use in your documents. You would create a

style class for each one by putting the following CSS code in a style sheet:

h1.silly { font: 36pt Comic Sans; }

h1.serious { font: 36pt Arial; }

Notice that these selectors include a period (.) after h1, followed by a descriptive class name. To

choose between the two style classes, use the class attribute, like this:

<h1 class="silly">Marvin's Munchies Inc. </h1>

<p>Text about Marvin's Munchies goes here. </p>

Or you could use this:

<h1 class="serious">MMI Investor Information</h1>

<p>Text for business investors goes here.</p>

When referencing a style class in HTML code, simply specify the class name in the class attribute of an

element. In the previous example, the words Marvin's Munchies Inc. would appear in a 36-point Comic Sans

font, assuming that you included a <link /> to the style sheet at the top of the web page and assuming that

the user has the Comic Sans font installed. The words MMI Investor Information would appear in the 36-point

Arial font instead. You can see another example of classes in action in Example 3.2; look for the subheader

<p> class and the footer <div> class.

What if you want to create a style class that could be applied to any element, rather than just headings

or some other particular tag? You can associate a style class with the <div> tag, as in Example 3.2, which

is used to enclose any text in a block that is somewhat similar to a paragraph of text; the <div> tag is

another useful container element.

You can essentially create your own custom HTML tag by using the div selector followed by a period

(.) followed by any style class name you make up and any style specifications you choose. That tag

can control any number of font, spacing, and margin settings all at once. Wherever you want to apply

your custom tag in a page, use a <div> tag with the class attribute followed by the class name you created.

**Tip**

You might have noticed a change in the coding style when multiple properties are included in a style

rule. For style rules with a single style, you’ll commonly see the property placed on the same line as the

rule, like this:

div.footer { font-size: 9pt; }

However, when a style rule contains multiple style properties, it’s much easier to read and understand

the code if you list the properties one-per-line, like this:

div.footer {

font-size:9pt;

font-style: italic;

line-height:12pt;

text-align: center;

padding-top: 30pt;

}

For example, the style sheet in Example 3.1 includes the following style class specification:

div.footer {

font-size: 9pt;

font-style: italic;

line-height: 12pt;

text-align: center;

padding-top: 30pt;

}

This style class is applied in Example 3.2 with the following tag:

<div class="footer">

Everything between that tag and the closing </div> tag in Example 3.2 appears in 9-point, centered, italic

text with 12-point vertical line spacing and 30 points of padding at the top of the element.

What makes style classes so valuable is how they isolate style code from web pages, effectively

allowing you to focus your HTML code on the actual content in a page, not how it is going to appear

on the screen. Then you can focus on how the content is rendered to the screen by fine-tuning the style

sheet. You might be surprised by how a relatively small amount of code in a style sheet can have

significant effects across an entire website. This makes your pages much easier to maintain and

manipulate.