CHAPTER 4

Cascading Style Sheets

HTML styles were introduced with HTML 4.0. Styles let you define how each element appears within a web page. Attributes like colors, fonts, and borders can be controlled through the use of styles. Styles are generally stored inside a *style sheet*. A style sheet can be defined either internally within the <head> element of the HTML page itself, or in an external ".css" file. Multiple external style sheets can be used within a single HTML page, giving the ability to break style-sheet definitions into reusable "modules" to be used in different parts of the HTML document.

The term *cascading style sheets* refers to the ability to define HTML element styles at multiple levels. This hierarchical approach allows generic styles to be overridden by specific HTML elements. The "cascading" hierarchy of styles is executed in the following order:

1. Default settings for the browser
2. External CSS style sheet definition
3. Internal CSS style sheet defined within the <head> element
4. Style defined within individual HTML elements

Styles defined at lower levels override those defined at higher levels. Style information on an HTML element, for example, will always override style information defined within an internal or external style sheet. Before we review the elements of style definition, let's examine how styles are defined using each of these methods.

Default/External Style Sheet

We'll examine default and external style sheets together because the default style sheet is simply an external style sheet that has been linked to the browser, to be used when no specific style definition exists. Within Internet Explorer, this link is defined from within the "Internet Options" dialog by clicking on "Accessibility" and selecting the "Format documents using my style sheet" checkbox. You supply the name of the file containing the external style sheet in the "Style sheet" field, as shown in Figure 4.1. The value specified in this field can be any external CSS file.

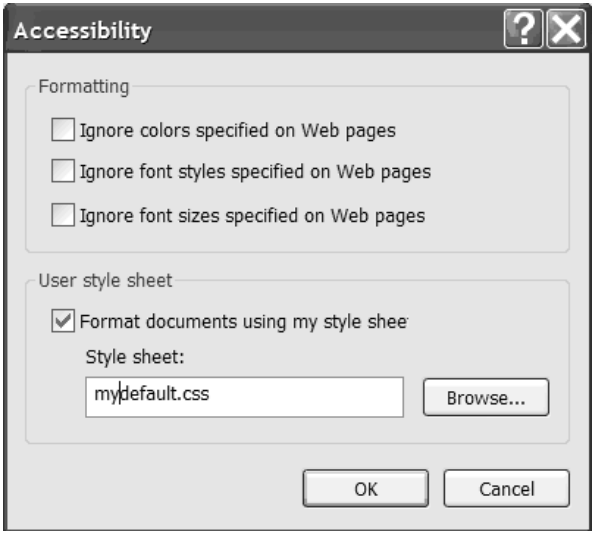

Figure 4.1: This dialog box is used to define the default style sheet in Internet Explorer.

External style sheets contain the same type of information that would be provided on an internal style sheet or an HTML element using the "style" attribute. To include an external CSS definition inside of an HTML web page, use the <link> element and specify the "href" attribute, as shown here:

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

This statement indicates that the external style sheet named "mystylesheet.css" should be loaded and used for this web page.

Only those elements with style definitions within that external style sheet will be affected by this. For that reason, multiple <link> elements can be specified within a single HTML page. When multiple <link> elements are specified, the first one takes precedence over the second, which takes precedence over the third, and so on. The attribute *rel="stylesheet"* specified here indicates an external style sheet. It's also possible to specify "alternate style sheet," to identify an alternative style sheet that can be used based on user selection. You'll learn more about switching style sheets a little later in this chapter.

Internal Style Sheet

Internal style sheet definitions let you incorporate style definitions directly into your HTML. This can be useful when you want to override settings in the default CSS or an external CSS for specific elements of your web page. Internal style sheets are defined using the <style> element within the <head> section of an HTML document, as shown here:

```
<head>
 <style type="text/css">
   body {background-color:BLUE; color:RED; text-align: center}
</style>
</head>
```

In this example, the <style> element is used to define the background color, font color, and text alignment for the body of this HTML document. The biggest drawback to using internally defined style sheets is that they become specific to the page in which they are contained.

As mentioned earlier, it's possible to combine external style sheets with internal definitions, giving you the ability to use a standard style definition and override it as needed. The following source illustrates this:

```
<head>
 <link href="standard.css" rel="stylesheet" type="text/css"/>
 <style type="text/css">
   td {background-color:Yellow; color:Black; text-align: center}
 </style>
</head>
```

This example derives most of its style information from the external style sheet file "standards.css." Within this page, however, the <td> elements will have their "background-color," "color," and "text-align" attributes overridden. This page-specific override will be in effect for all <td> elements in this page. You can also override specific elements within the page.

Inline HTML Style Definition

So far, you've seen how to define style information globally for your browser, using external style sheets, and using internal style sheets. It's also possible to override style information for individual elements within a web page. This capability is achieved through the "style" attribute. Most HTML elements support the "style" attribute.

The following lines of code use this attribute on several different HTML elements:

```
<p style="color:red; text-align:center;">My Text</p>
<table style="color:Blue;
border-style:solid;"><tr><td>Data</td></tr></table>
<input type="text" style="background-color:Orange;" name="textbox">
```

In this example, the <p> element is defined as having red text, centered within the document. The <table> element is defined as having blue text and a solid border around the table. With the <input> element, the resulting text box is displayed with an orange background.

As you can see, the style properties are defined in the format *property-name: property value.* Also, notice that multiple properties can be specified by separating them with semicolons (;).

As mentioned earlier, the concept of cascading style sheets means that higher-level definitions can be overridden at a lower level. The example below combines the previous two examples to help illustrate this:

```
<head>
 <link href="standard.css" rel="stylesheet" type="text/css"/>
 <style type="text/css">
   td {background-color:Yellow; color:Black; text-align: center}
 </style>
</head>
<body>
  <p style="color:red; text-align:center;">My Text</p>
  <table style="color:Blue; border-style:solid;">
  <tr><td>Data</td></tr></table>
  <input type="text"
         style="background-color:Orange;" name="textbox">
</body>
```

Within this page, any style properties defined in "standard.css" are applied to the page, with the exception of any <td> elements. These elements use the style defined in the internal style sheet. The color:Black property defined within that internal style sheet is overridden by the specific table definition, which redefines this property as color:Blue.

Style Sheet Properties

Now that you've seen where to define style sheet properties, let's take a look at what style sheet properties exist, along with their uses.

Properties used to defined style elements within an HTML document can be broken down into a set of simple categories. Each of these categories affects a different aspect of the way the document appears to the user.

Background Properties

The background properties define the look of the background of a displayed HTML document. Table 4.1 lists these style sheet properties.

Table 4.1: Background Style Sheet Properties	
Property	**Description**
background	This property can be used to define any of the other background properties.
Background-attachment	This property identifies whether a defined background image is fixed or scrolls with the rest of the window. Possible values are "scroll" and "fixed." This property is used when the background-image is also specified.
background-color	The background color for the document is defined using this property. The value can be defined as a color name, an RGB value, or a hex color code.
background-image	This is used to define an image file to be displayed as the background for a defined HTML document.
Background-position	The position of a background image within the document is defined using this property. This value can be specified as a constant (top left, etc.), a percent of the page down and across from the top left corner (such as 10% or 25%), or a specific number of pixels down and across from the top left corner.
background-repeat	This value identifies whether or not a background image should be repeated throughout the document.

The generic "background" property can be used instead of the other specific values. For example, the following style sheet assignment would be perfectly acceptable:

```
body {
background: red url(image.jpeg) no-repeat top left
}
```

This example defines the background information for the HTML <body> element. The color is defined using the named color "red." A background image is defined using the url() modifier. The "no-repeat" value indicates that the background image should be displayed only once. The positional values "top" and "left" indicate that the image should be positioned in the upper-left corner of the document.

It's also possible to explicitly define individual background properties. The "background-color" property identifies a color to be displayed as the background of a given element. The format of the value used to define this property can be any of the following:

- **Color name:** These are English color names, such as "red," "blue," or "green."
- **RGB color value:** This is in the format *rgb(red value, green value, blue value)*, where each color value is a number between zero and 255.
- **Hexadecimal color code:** This is a six-digit hex (hexadecimal) value, where the first two digits represent the amount of red, the next two digits represent the amount of green, and the last two digits represent the amount of blue.

Which method you use is largely a matter of personal preference. For example, the following three color definitions would all result in a red background:

```
background-color: red
background-color: rgb(255, 0, 0)
background-color: #ff0000
```

The "background-image" property lets you define an image file to be displayed in the background of a defined element. The image file itself is identified using the url() modifier, with the name and path to the image file indicated in parentheses. You can also specify a value of "none" to indicate that no background image should be displayed. The example below illustrates the use of this property:

```
td {
background-image: url(logo.gif)
}
```

This example results in the image "logo.gif" being displayed as the background to any <td> elements within the document. It's important to note here that many elements can use the "background-color" and "background-image" properties. There are also several background properties that act as modifiers to the "background-image" property. The "background-attachment" property, for example, identifies whether or not a defined background image scrolls with the rest of the document. The two possible values for this property are "fixed," to identify that the image should not

be scrolled with the rest of the page, or "scroll," to indicate that the image should scroll with the rest of the page. Here are two examples of defining this property:

```
<body style='background-image:url(back.gif);
background-attachement:fixed'>
…
<style type="text/css">
   body {
      background-image: url(image.gif);
      background-attachment: scroll;
   }
</script>
```

In the first example here, the style definition is made within the <body> element directly. A background image named "back.gif" is defined as a fixed image. In the second example, the background image definition is again associated with the body element, but this time, it's defined within a style sheet. The image "image.gif" will also scroll with the rest of the document.

In addition to being able to define whether or not the image scrolls, it's also possible to define the position of the background image within the page. This is accomplished through the "background-position" property. This property can be defined using constants defining the position relative to the page, a relative position expressed as a percentage of the page, or a position expressed in pixels relative to the top left corner of the page. The following lines illustrate each of these methods:

```
background-position: top left; //upper left corner
background-position: 10% 25%; //10% of the page down,
   25% of the page over
background-position: 75 45;  // 75 pixels down, 45 pixels over
```

Valid constants for vertical positioning are "top," "middle," or "bottom." For horizontal positioning, valid values are "left," "center," or "right." As illustrated in the lines here, the vertical and horizontal values can be combined. If a vertical value is specified with no horizontal value, "center" is assumed.

The "background-repeat" property identifies whether or not a background image is tiled if necessary to fill a space. The valid

values for this property are “repeat” to indicate that the image should be tiled across the element, “repeat-x” to indicate that the image should be repeated horizontally across the element, “repeat-y” to repeat the image vertically, and “no-repeat” to indicate that the background image should appear only once.

Border Properties

Many HTML elements, such as tables, table cells, images and text boxes, support visible borders. Several CSS properties allow you to control how these borders are displayed. Table 4.2 lists these border properties.

Table 4.2: Border Style-Sheet Properties

Property	Description
border	This property lets you set all border properties in a single declaration.
border-bottom	This property defines the bottom border for a specified element.
border-bottom-color	This property identifies the color of the bottom border.
border-bottom-style	This property identifies the style of the bottom border.
border-bottom-width	This property identifies the width of the bottom border.
border-color	This property identifies the color of all borders for the specified element.
border-left	This property defines all of the properties associated with the left border.
border-left-color	This property identifies the left border color.
border-left-style	This property identifies the style of the left border.
border-left-width	This property identifies the width of the left border.
border-right	This property defines all of the properties associated with the right border.
border-right-color	This property identifies the right border color.
border-right-style	This property identifies the style of the right border.
border-right-width	This property identifies the width of the right border.
border-style	This property defines the style for all four borders in a single declaration.

Table 4.2: Border Style-Sheet Properties	
Property	**Description**
border-top	This property defines all of the properties associated with the top border.
border-top-color	This property identifies the top border color.
border-top-style	This property identifies the style of the top border.
border-top-width	This property identifies the width of the top border.
border-width	This property defines the width for all four borders in a single declaration.

You might have noticed that there is a hierarchical structure to these properties. The simple "border" property can be used to define all of the attributes associated with the element border in a single declaration. The "border-left" property can be used to set all of the properties associated with the left border. The "border-left-color" property defines the color of the left border only. An example of the use of the "border" property is shown below:

```
table
{
border: thin dotted blue
}
```

In this example, all of the table's borders would have a thin width, a dotted style, and a blue color. Alternatively, the "border-bottom," "border-top," "border-left," and "border-right" properties could be used to define these same three properties for the specific border in question, as shown here:

```
td
{
border-top: thin solid gray;
border-left: thin solid gray;
border-bottom: medium solid black;
border-right: medium solid black;
}
```

This example declares that <td> elements will have borders with thin, solid, gray lines on the top and left, and medium-width,

solid, black borders on the bottom and right. The result is a somewhat 3-D effect, as shown in Figure 4.2.

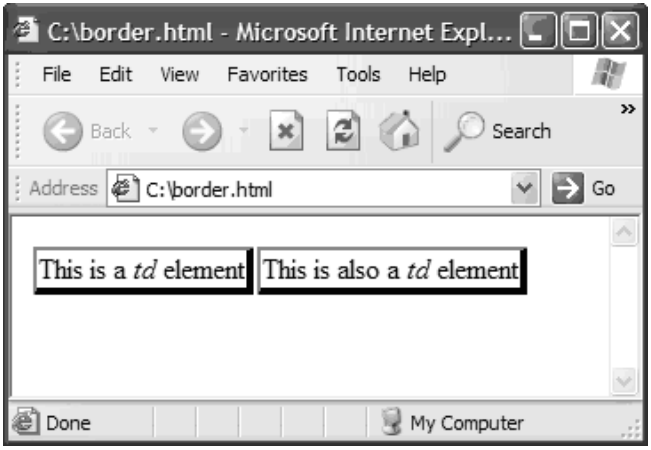

Figure 4.2: Defining border properties for table cells can give a 3-D effect.

It's also possible to define the specific portions of the border properties individually. The "border-color" property, for example, can be used to define all four border colors in a single declaration. Up to four color values can be specified with this property. These values can be either color names, RGB color values, or hexadecimal color codes. Here are three examples of defining border color using this property:

```
p {
  border-color: red white blue yellow;
}

tr {
   border-color: black orange;
}

img {
    border-color: green;
}
```

The first example sets all four border colors individually, starting at the top and going clockwise, so the top border is red, the right border is white, the bottom border is blue, and the left border is yellow. The second example uses only two colors. The first value

identifies the color for the top and bottom borders, and the second value defines the color for the left and right borders. (To define an individual border color, use a property such as "border-left-color.") When defining any of the "border color" parameters, it's also possible to use the constant "transparent" to indicate that no border should be generated. In that case, whatever coloring is "behind" the element will be visible.

The "border-style" and "border-width properties" can be used to define the style and width for all four borders in a single declaration. Here are examples of each of these:

```
table {border-style: dotted dashed
border-width: thin medium}

tr {border-style: dotted dashed inset double
border-width: 3 4 5 6}

td {border-style: groove
thick}
```

The first example defines the top and bottom table borders as thin dotted lines and the left and right borders as dashed medium-width lines. The second example defines each border for table rows individually. Note that the widths here are pixels instead of constants. The final example defines a thick, 3-D grooved border style for <td>. In these examples, the "border-width" properties are defined using either a constant such as "thin" or a numeric pixel value. The "border-style" property is defined using one of the constants shown in Table 4.3.

Table 4.3: Possible Border Style Values	
Value	**Description**
none	No border is displayed.
hidden	As with “none,” no border is displayed.
dotted	This value displays a dotted-line border.
dashed	This value displays a dashed-line border.
solid	This value displays a solid-line border.
double	This value displays a double-line border.
groove	A 3-D grooved border is displayed.
ridge	A ridged 3-D border is used.
inset	A border with an inset 3-D effect is displayed.
outset	A border with an outset 3-D effect is displayed.

Each of these border styles gives a unique effect to the HTML element with which it is used, as shown in Figure 4.3.

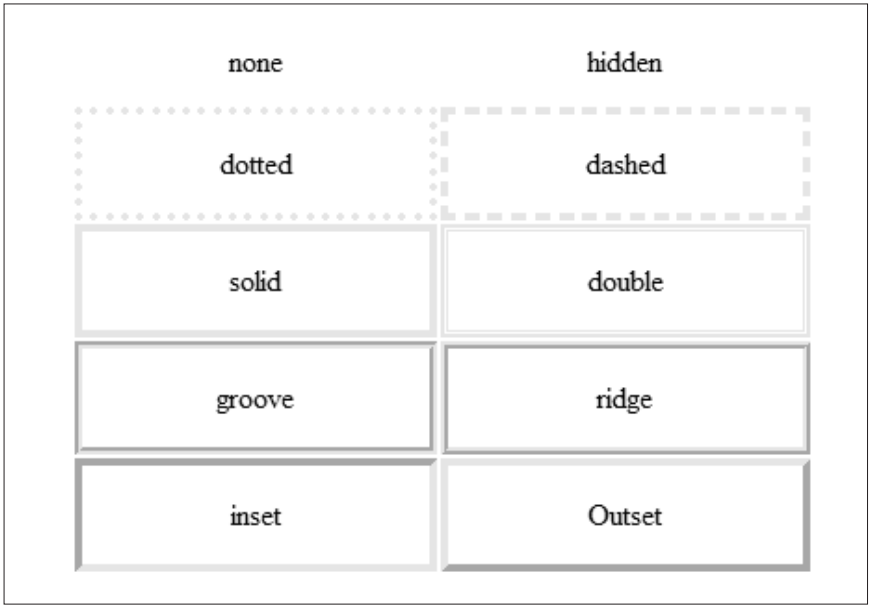

Figure 4.3: Each border style gives a different effect.

It's also possible to specifically define border property elements individually. For example, you could use the "border-left-color," "border-left-width," and "border-left-style" properties to individually define the properties associated with the left border, as shown in this example:

```
table {border-left-color: red;
       border-left-style: dotted;
       border-left-width: medium;}
```

This identifies that the left table border should have a medium-width, red, dotted line. It is equivalent to using "border-left: red dotted medium." Which method you choose to define the border properties depends on personal preference and the need you are attempting to meet. However, the use of specific property definitions is easier to read and understand.

Font Properties

The way text is displayed in HTML elements is controlled via the font properties. Table 4.4 lists these properties.

Table 4.4: CSS Font Properties

Property	Description
font	This property lets you define all of the attributes of a font in one declaration.
font-family	This property lists the font family names to be applied to an element, in the order in which they should be applied.
font-size	This property defines the size of the font for a specified element, using a defined constant, a specified size, or a percentage.
font-style	This property defines font styling using one of the constants "none," "italic," or "oblique."
font-variant	This property indicates whether to display text in a small-caps font or a normal font.
font-weight	This property defines the font weight using a constant value ("normal," "bold," "bolder," "lighter") or a numeric value from 100 to 900.

As with the other CSS properties you've seen, the font properties let you define the same attributes at several different levels

through different properties. The "font" property, for example, lets you define all font attributes in a single declaration, as shown below:

```
td
{
font: italic bold small-caps 12px arial
}
```

In this example, the text associated with all <td> elements will be displayed using Arial as the font, in a size of 12 pixels, with italic style, bold weight, and small caps.

You can also define fonts to match settings on the system, using a set of constant values. These font constants allow you to define the look of your web page to match that of the environment in which the page is displayed. This way, custom menus you create using CSS can match the look and feel that the user is used to. Table 4.5 lists these constants and their definitions.

Table 4.5: Font Constants

Font Constant	Description
caption	Use the same font defined for captioned controls (radio buttons, drop downs, etc.).
icon	Use the same font as text displayed with icons.
menu	Use the same font displayed on window menus.
message-box	Use the same font displayed within message boxes.
status-bar	Use the same font displayed within the window's status bar.

The "font-family" property explicitly defines font names to be used with the specified element. The property accepts a list of fonts, which is applied in order from left to right. If the first font listed on the property does not exist on the client system, the next font will be used, and so on. The "font-size" property sets the size of the text font associated with the element. This property can be set using one of several constants, a specified size in points, or a percentage of the element containing the font. The constant values are listed in Table 4.6.

Table 4.6: Font-Size Constants

Font Size Constant	Description
xx-small	Use a double extra-small font size, based on the browser's font settings.
x-small	Use the extra small font.
Small	Display the font in the small font size.
Medium	Display the font using the medium font size.
Large	Display the font using the large font size.
x-large	Display the font using the extra-large font size.
xx-large	Display the font using the double extra-large font size.
smaller	Display the font one font size smaller than the current setting in effect.
larger	Display the font one font size larger than the current setting in effect.

The "font-style" property sets special font characteristics. The three possible options for this property are "normal," "italic," and "oblique." Similarly, the "font-variant" property further defines how the font is displayed. This property accepts two possible values: "normal" and "small-caps." The "small-caps" value indicates that lowercase letters should be displayed as smaller versions of the corresponding uppercase letters.

The final property for defining the look of the font is "font-weight." This property is used to control bold printing. Possible values for this property are the absolute values "normal" and "bold," or the relative values "bolder" and "lighter" to adjust the level of boldness relative to the current setting. It's also possible to use numeric values from 100 through 900, indicating the level of boldness. The lower values indicate lighter font thickness, while higher values indicate heavier thickness.

List Properties

List properties let you define text to be displayed as a bulleted list, along with defining the style of that bulleted list. Table 4.7 lists these properties.

Table 4.7: List Properties	
Property	**Description**
list-style	Use this property to set all of the list attributes.
list-style-image	This property defines an image file as the bullet for the list.
list-style-position	This property defines the location of the list marker. Possible values are “inside” and “outside.”
list-style-type	This property defines the type of marker displayed with the list.

The “list-style” property defines all of the list’s attributes in a single statement, as shown below:

```
{
  list-style: outside decimal
}
```

This example indicates that the list associated with the style should be decimal and numeric, with the list marker located in an outside position.

The “list-style-image” property indicates that an image file should be used as the list marker. The example below illustrates the use of this property:

```
{
  list-style-image: url(bullet.gif);
  list-style-type: disc;
}
```

In this case, the image file bullet.gif will be used as the list marker. Note that the “list-style-type” property also allows for the possibility that the list image is not available.

The “list-style-position” property identifies the location of the list item marker. A value of “inside” indicates that the marker should be left-justified relative to the list item’s text. A value of “outside” leaves the list marker right-justified.

The “list-style-type” property, as you have already seen, defines the type of marker to be displayed with the list. This property lets you define if a simple bulleted list should be displayed, or if a

numbered list should be used instead. Table 4.8 lists the available values for this property.

Table 4.8: Possible List-Style-Type Values

List Style Type	Description
none	No list marker is displayed.
disc	A filled circle (disc) is displayed.
circle	An open circle is displayed.
square	A square is displayed.
decimal	Numbers are displayed.
decimal-leading-zero	Numbers with leading zeros are displayed.
lower-roman	Lowercase roman numerals are displayed.
upper-roman	Uppercase roman numerals are displayed.
lower-alpha	Lowercase letters are used for markers (a, b, c).
upper-alpha	Uppercase letters are used for markers (A, B, C).

These list properties define the style for the list tags <ul> and <li>. The example below illustrates the use of the "list-style-type" property within a <ul> tag:

```
<ul style="list-style-type: upper-roman; list-style-position:
outside;">
<li>First Item</li>
<li>Second Item</li>
<li>Third Item</li>
<li>Fourth Item</li>
</ul>
```

This example also incorporates the "list-style-position property" to cause the list marker to be right-justified. Figure 4.4 shows the output from this example.

Figure 4.4: This list is generated using the list properties.

Margin Properties

To adjust the amount of space around an element, you can use the margin CSS properties. Five properties are available to define margins. These are shown in Table 4.9.

Table 4.9: Properties to Define Element Margins

Property	Description
margin	This property sets all margins in a single definition.
margin-bottom	This property defines the bottom margin.
margin-left	This property defines the left margin.
margin-right	This property defines the right margin.
margin-top	This property defines the top margin.

For any of the margin-defining properties in Table 4.9, the value supplied should be either a numeric value representing pixels, a percentage representing the percent of the total page, or the constant value “auto” to allow the browser to adjust the margin automatically. It’s also possible to define a negative value, which results in elements overlapping within the page.

If all four margins are to be the same, you can provide a single value. Alternatively, you can specify two values to define the top and bottom margins the same, and the right and left margins the

same. If you want each of the four margins to be different, specify the values in the order top, right, bottom, and left. Below are examples of each of these scenarios:

```
{
margin: auto; // All four margins the same
margin: 8px 3%; // Top and bottom 8 pixels, left and right margins
at 3%
margin: 8px 7px 6px 5px; // Each margin defined individually
}
```

Padding Properties

While margin properties define the space between elements, padding properties define the space between an element's border and the content contained within the element. The best way to illustrate this is to look side-by-side at examples of padding and margins.

The source below defines two table cells, one with a three-pixel margin and the other with no margin, but a three-pixel padding:

```
<table><tr>
<td style='margin: 3px; border=2px'>Cell data</td>
<td style='margin: 0px; padding:3px; border:2px;'>Cell data</td>
</tr>
</table>
```

The first cell will have white space around its border. The second cell will have the white space between the border and the cell data. Figure 4.5 shows the resulting output.

As with the margin, padding attributes for all four sides of an element can be set using the "padding-bottom," "padding-left," "padding-right," and "padding-top" properties, or with the single "padding" property. Supported values for padding properties include a numeric pixel value

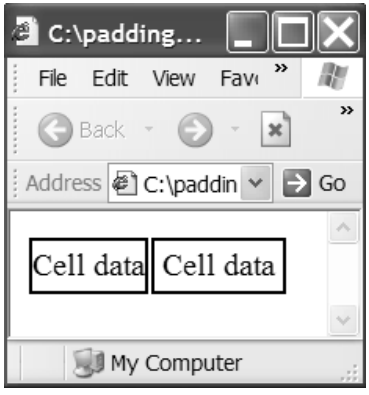

Figure 4.5: These cells illustrate margin versus padding.

or a value that represents a percentage of the total page size. Again, as with the "margin" property, the "padding" property can be set using a single value for all four sides, two values to identify the top and bottom as a pair and the left and right as another pair, or four values to define all four sides individually. The example below illustrates each of these scenarios:

```
{
padding: 3px; // all four sides
padding: 3px 2px; // top/bottom set to 3 pixels, left/right set to
2 pixels
padding: 2% 3% 1% 5%; // each side set individually
}
```

Text Properties

In addition to the font properties you saw earlier, there several other properties for defining the look of the text associated with an element. A list of these properties is shown in Table 4.10.

Table 4.10: Properties to Define How Text Is Displayed within an Element

Property	Description
color	This property defines the text color.
direction	This property indicates whether text should be displayed right to left or left to right.
letter-spacing	This property defines the spacing between letters.
text-align	This property identifies how the text should be aligned relative to the element.
text-decoration	This property defines special text attributes, such as underlined or strike-through.
text-indent	This property defines indentation for the first text line in an element.
text-transform	This property allows you to transform the case of the supplied text.
white-space	This property defines how the white space within an element is handled relative to the text within the element.
word-spacing	This property defines the amount of spacing between words within an element.

The value of the "color" property can be supplied as an RGB() value, a hex code representation of a color, or one of the defined color constants supported by the browser. For example, in the following lines, the text in the first cell is displayed in red, the text in the second cell is black, and the text in the third cell is gray:

```
<td style="color:Red">Red Text</td>
<td style="color:#000000">Black Text</td>
<td style="color:rgb(128, 128, 128)Gray Text</td>
```

The "direction" property identifies the direction in which the text is displayed. Supported values for this property are "ltr" to indicate that the text is displayed from left to right, or "rtl" to indicate that the text should be displayed from right to left. The two lines below illustrate each scenario:

```
<td style="direction:rtl">backward</td>
<td style="direction:ltr">forward</td>
```

The text itself will appear as it is typed; however, it will be aligned according to the supplied value. Right-to-left alignment is used for languages normally read from right to left, like Arabic or Hebrew.

The "letter-spacing" property defines the spacing between letters. The value supplied should be either a numeric pixel value or the constant "normal" to identify that standard spacing for the defined font should be used. It's possible to define a negative value for this property, resulting in decreased spacing between characters. Both increased and decreased spacing are illustrated in this example:

```
<p style="letter-spacing:3px">Increased Spacing</p>
<p style="letter-spacing:-3px">Decreased Spacing</p>
```

The first line here increases the letter spacing by three pixels, while the second decreases that spacing by three pixels. Figure 4.6 shows the results of this example in the browser window.

The "text-align" property defines the alignment of text within an element. In addition to the values of "left," "center," "right," this property also supports "justify" to indicate that the text should fill the available space from left to right.

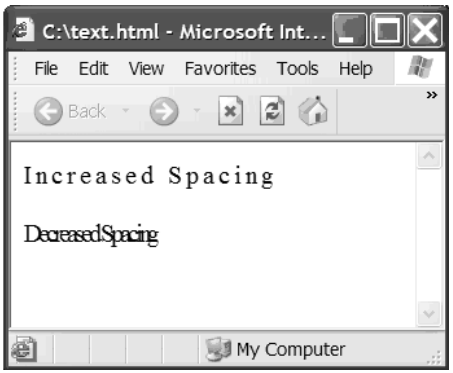

Figure 4.6: These two examples illustrate the "letter-spacing" property.

The "text-decoration" property indicates that certain special effects should be applied to the text. A list of the possible values for this property is shown in Table 4.11.

Table 4.11: Possible Values for the Text-Decoration Property

Value	Description
Blink	The text blinks when displayed.
line-through	The text is displayed in a strikethrough style.
none	No decoration is applied.
Overline	A line is displayed above the text.
underline	A line is displayed under the text.

Figure 4.7: Here are the possible text-decoration styles.

Each of these possible values gives you the ability to further customize the look of text associated with a given element, as shown in Figure 4.7. (Note that the "blink" value is not supported by some browsers.)

The "text-indent" property defines the amount of indentation for the first line of text. The value provided can be a numeric value representing a number of pixels, or a percentage that identifies a percentage of the total

width of the element. It's also possible to use a negative value on this property, causing the text to be shifted to the left, rather than indented to the right.

The "text-transform" property changes the case of the text. Possible values for this property include "capitalize" to change the first letter of each word to a capital letter and all others to lowercase, "lowercase" to convert all letters to lowercase, "none" to leave the text as-is, and "uppercase" to convert the text to all uppercase letters. The lines in the example below illustrate these values:

```
<p style="text-transform:capitalize">
The quick brown fox jumps over the lazy dog</p>
<p style="text-transform:lowercase">
The quick brown fox jumps over the lazy dog</p>
<p style="text-transform:none">
The quick brown fox jumps over the lazy dog</p>
<p style="text-transform:uppercase">
The quick brown fox jumps over the lazy dog</p>
```

The text enclosed within the <p> element is identical in each line, but the value of the "text-transform" property is different. Figure 4.8 shows the results of this example.

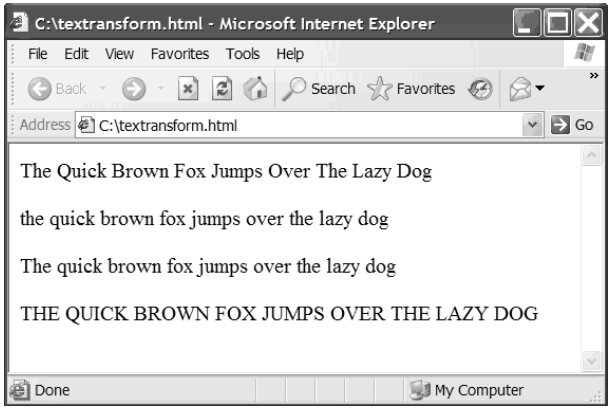

Figure 4.8: Using the "text-transform" property changes text case.

The “white-space” property defines how the browser should deal with white space within the element. Possible values are “normal” to identify that the browser should ignore white space, “pre” to leave the text spacing as-is in the same way that the <pre> tag does, and “nowrap” to identify that the text should not be wrapped within the element. Similarly, the “word-spacing” property lets you increase or decrease the amount of spacing between words within the element. The value “normal” on this property defines that the normal word spacing for the font should be used. A numeric value represents a fixed number of pixels to space between words. A negative value can also be used to decrease the amount of spacing between words.

CSS and JavaScript

The CSS properties discussed in this chapter allow you to create a very customizable user interface within a browser application. CSS properties go hand-in-hand with JavaScript, in that JavaScript can access and change any of these properties. Style-sheet properties are accessed from within JavaScript through the use of the JavaScript “style” object. This object gives full access to all of the style-sheet properties for a given element. Below is an example of using this function to change the background color of an HTML table cell (<td>) element with an ID of “td1”:

```
<table><tr><td id="td1">Cell 1</td></tr></table>
<Script language="JavaScript">
var td1 = document.all(“td1");
td1.style.backgroundColor = “Yellow”;
</script>
```

This example illustrates the one difference between accessing a style-sheet property in CSS and accessing the same property in JavaScript. Inside of the JavaScript “style” object, capitalization takes the place of hyphens for multiple-word properties. For example, the CSS property “background-color” is replaced with “backgroundColor.” Other than this difference in naming convention, all of the functionality explained in this chapter applies to defining values for style-sheet properties in JavaScript.

Summary

In this chapter, you saw how style-sheet properties can be used to customize the look of a web page. In the next chapter, you'll examine JavaScript function templates to perform common tasks. These function templates can be customized to meet your exact needs.