Text Alignment and Spacing

CSS includes a great many properties for controlling how text appears on a Web page. If you've ever wondered how to indent paragraphs, space out lines, or center a title, these are the tools you need.

Table 6-3 has the details on all your alignment options.

Table 6-3. *Alignment and spacing properties*

Property	Description	Common Values	Can Be Inherited?
text-align	Lines text up on one or both edges of a page.	left, right, center, justify	Yes
text-indent	Indents the first line of text (typically in a paragraph).	A pixel value (indicating the amount to indent) or percentage of the width of the containing element.	Yes
margin	Sets the spacing added around the outside of a block element (page 113). You can also use the similar properties margin-bottom, margin-left, margin-right, and margin-top to change the margin on just one side.	A pixel value or percentage indicating the amount of space to add around the element.	No
padding	Sets the spacing added around the inside of a block element. Has the same effect as margin, unless you have an element with a border or background color.	A pixel value or percentage indicating the amount of space to add around the element.	No
word-spacing	Sets the space between words.	A pixel value or percentage.	Yes
letter-spacing	Sets the space between letters.	A pixel value or percentage.	Yes
line-height	Sets the space between lines.	A pixel value or percentage. You can also use a multiple (for example, use 2 for double-spacing).	Yes
white-space	Tells the browser how to deal with spaces in your text.	normal, pre, nowrap	Yes

For example, if you want to create a page that has indented paragraphs (as a novel or newspaper does), use this style sheet rule:

```
p {
  text-indent: 20px
}
```

In the following sections, you'll see examples that use the alignment and margin properties.

Alignment

Ordinarily, all text in a Web page lines up on the left side of the browser window. Using the *text-align* property, you can center text, line it up on the right edge, or justify it. Figure 6-9 shows your options.

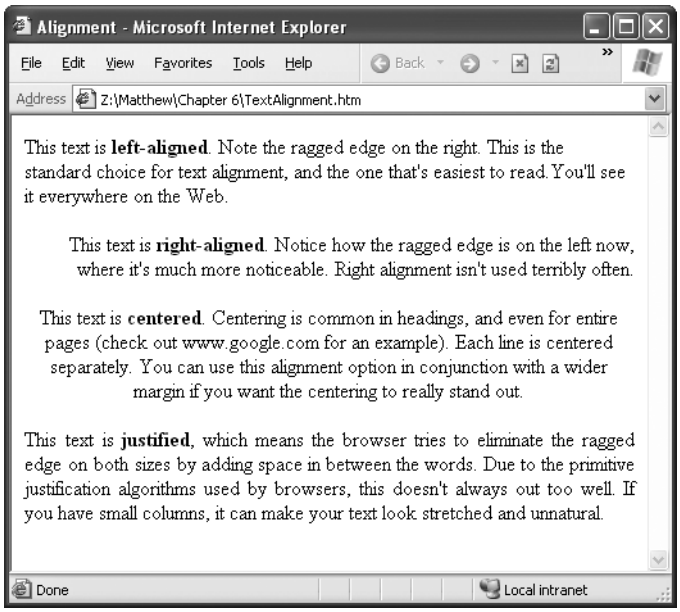

Figure 6-9:
This page shows common types of text alignment.

The most interesting alignment choice is full justification, which formats text so that it appears flush with both the left and right margins of a page, like the text in this book. You specify full justification with the *justify* setting. Originally, printers preferred full justification because it crams more words onto each page, reducing a book's page count and, therefore, its printing cost. These days, it's a way of life. Many people feel that text with full justification looks neater and cleaner than text with a ragged edge, even though tests show plain, unjustified text is easier to read.

Justification doesn't work as well in the Web world as in print. A key problem is a lack of word-splitting rules, which split long words into syllables, hyphenate them, and extend them over two lines. Browsers use a relatively simplistic method to justify text. Essentially, they add words to a line one at a time, until no more words can fit, at which point they add extra spacing between the words to pad the line to its full length. By comparison, the best page layout systems for print analyze an entire paragraph and find the optimum justification strategy that best satisfies every line. In problematic cases, a skilled typesetter may need to step in and adjust the line breaking manually. Compared to this approach, Web browsers are irredeemably primitive, as you can see in Figure 6-10.

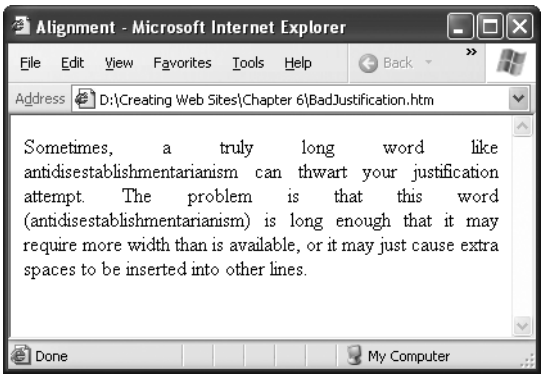

Figure 6-10:
If you decide to use full justification on a Web page, make sure you use fairly wide paragraphs. Otherwise, you'll quickly wind up with gaps and rivers of white space. Few Web sites use justification.

Spacing

To adjust the spacing around any element, use the *margin* property. For example, here's a rule that adds a fixed spacing of 8 pixels to all sides of a paragraph:

```
p {
  margin: 8px;
}
```

This particular rule doesn't have much effect, because 8 pixels is the standard margin that Web browsers apply around block elements on all sides. The 8-pixel margin ensures a basic bit of breathing space. However, if you want to create dense pages of information, this space allowance can be a bit too generous. Therefore, many Web site developers look for ways to slim down the margins a little bit.

One common trick is to close the gap between headings and the text that follows them. Here's an example that puts this into action using inline styles:

```
<h2 style="margin-bottom: 0px">This heading has no bottom margin</h2>
<p style="margin-top: 0px">This paragraph has no top margin.</p>
```

You'll notice that this style rule uses the more targeted *margin-top* and *margin-bottom* properties to home in on just one margin. You can also use *margin-left* and *margin-right* to set side margins. Figure 6-11 compares some different margin choices.

If you're daring, you can even use *negative* margins. Taken to its extreme, this can cause two elements to overlap. However, a better approach for overlapping elements is absolute positioning, a style trick you'll pick up on page 253.

Note: Unlike most other CSS properties, margin settings are never inherited. That means if you change the margins of one element, other elements inside that element aren't affected.

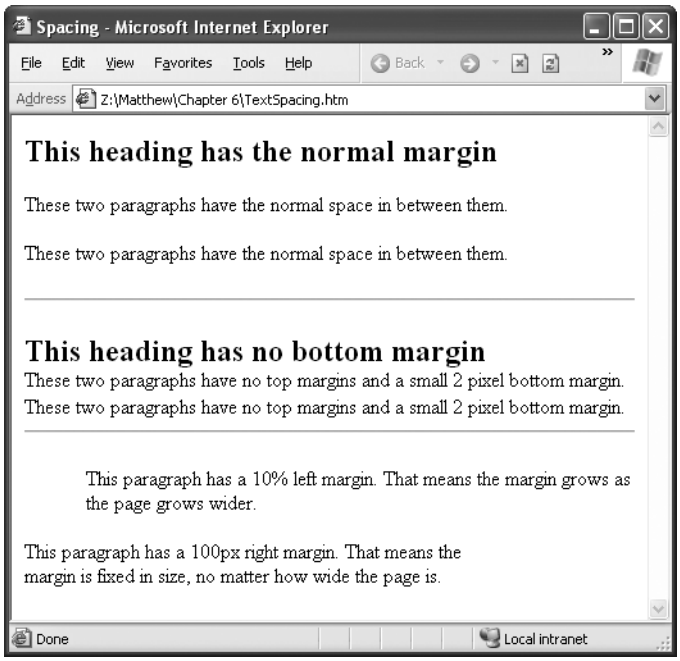

Figure 6-11:
When you want to change the spacing between page elements like headers and paragraphs, you need to consider both the element above and the element below. For example, if you stack two paragraphs on top of each other, two factors come into play–the bottom margin of the top paragraph, and the top margin of the bottom paragraph. Browsers use the larger of these two values. That means there's no point in shrinking the top margin of the bottom element unless you also shrink the bottom margin of the top element. On the other hand, if you want more space, you only need to increase the margin of one of the two elements.

White Space

As you learned in earlier chapters, XHTML has a quirky way of dealing with spaces. If you put several blank spaces in a row, XHTML treats that first space as a single space character, and ignores the others. That makes it easy for you to write clear XHTML markup because you can add spaces wherever you like without worrying about it affecting your Web page.

You've already learned about two ways to change how browsers deal with spaces: the   character entity (page 117) and the <pre> element (page 119). You can replace both of these workarounds with the *white-space* style sheet property.

First, consider the   character entity. It has two purposes—it lets you insert spaces that a browser won't ignore, and it prevents the browser from wrapping a line in the middle of a company name or some other important term. Here's an example of the latter technique:

```
<p>You can trust the discretion of
Hush Hush Private Plumbers</p>
```

This works (the page displays the text *Hush Hush Private Plumbers* and doesn't wrap the company name to a second line), but it makes the markup hard to read. Here's the style-sheet equivalent with the *white-space* property set to *nowrap*:

```
<p>You can trust the discretion of
<span style="white-space: nowrap">Hush Hush Private Plumbers</span></p>
```

To make this trick work, your XHTML needs to wrap the company name in a container that applies the formatting. The element (page 123) is a good choice, because it doesn't apply any formatting except what you explicitly add.

Now, consider the <pre> element, which tells a browser to pay attention to every space in the content inside. On page 119, you saw how you could use the <pre> element to give the correct spacing to an e. e. cummings poem. You can get the same effect by setting the *white-space* property of an element (say, a , , or <p> element) to *pre*:

```
<p style="white-space: pre">Your browser won't    ignore    these
                          s   p  a  c  e  s    .</p>
```

When you use the *pre* value for the *white-space* property, the browser displays all spaces, tabs, and hard returns (the line breaks you create when you hit the Enter key). But unlike the <pre> element, the *pre* value of the *white-space* property doesn't change the text font. If you want to use a fixed-width font like Courier to space your letters and spaces proportionally, you need to add a *font-family* property (page 154).

Borders

The last group of style sheet properties you'll learn about in this chapter lets you add borders to your Web page (Figure 6-12). Borders are a great way to separate small pieces or entire blocks of content.

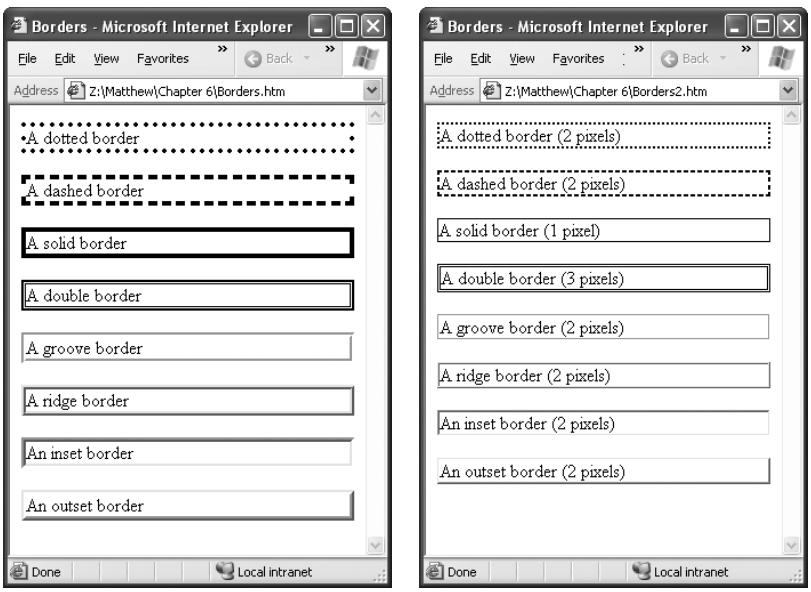

Figure 6-12: *Left: The basic border styles look a bit old-fashioned in today's sleek Web.*

Right: Shrink these borders down to one or two pixels, and they blend in much better.

Table 6-4 lists the three key border properties.

Table 6-4. *Border properties*

Property	Description	Common Values	Can Be Inherited?
border-width	Sets the thickness of the border line. Usually, you'll want to pare this down.	A pixel width.	No
border-style	Browsers have eight built-in border styles. The border style determines what the border looks like.	none, dotted, dashed, solid, double, groove, ridge, inset, outset	No
border-color	The color of the border.	A color name, hexadecimal color code, or RGB value (see page 151).	No

Basic Borders

The first choice you make when you create a border is the style you want it to have. You can use a dashed or dotted line, a groove or a ridge, or just a normal thin hairline (which often looks best). Here's a rule that creates a dashed border:

```
p {
  border-style: dashed;
}
```

To make a border look respectable, you need to reduce the border width. The standard border width is almost always too clunky. You should reduce it to one or two pixels, depending on the border style:

```
p {
  border-style: dashed;
  border-width: 2px;
}
```

You can also use properties like *border-top-style* and *border-left-width* to set different styles, width, and colors for every side of your element. Using many properties at once can occasionally create an unusual effect, but you usually don't need to get this detailed. Instead, check out the border optimization tips in the next section.

Making Better Borders

In Figure 6-12 the actual borders look fine, but they're too close to the text inside the boxes formed by the borderlines, as well as by the edges of the page.

To make a border stand out, consider using the *border* property in conjunction with three other properties:

- **background-color** (page 151) applies a background color to your element. When used in conjunction with a border, it makes your element look like a floating box, much like a sidebar in a magazine article.

- **margin** (page 163) lets you set the spacing between your border box and the rest of your page. Increase the margin so that your boxes aren't crowded up against the rest of the page's content or the sides of a browser window.
- **padding** works like the *margin* property, but it sets spacing *inside* your element, between the edges of the box and the actual content within it. Increase the padding so that there's a good amount of space between a border and your box text. Figure 6-13 shows the difference between margin and padding.

Here's an example of a paragraph that looks like a shaded box:

```
p {
  background-color: #FDF5E6;
  margin: 20px;
  padding: 20px;
  border-style: solid;
  border-width: 1px;
}
```

Figure 6-13 shows how the *margin*, *padding*, and *background-color* properties change an ordinary paragraph into a shaded box.

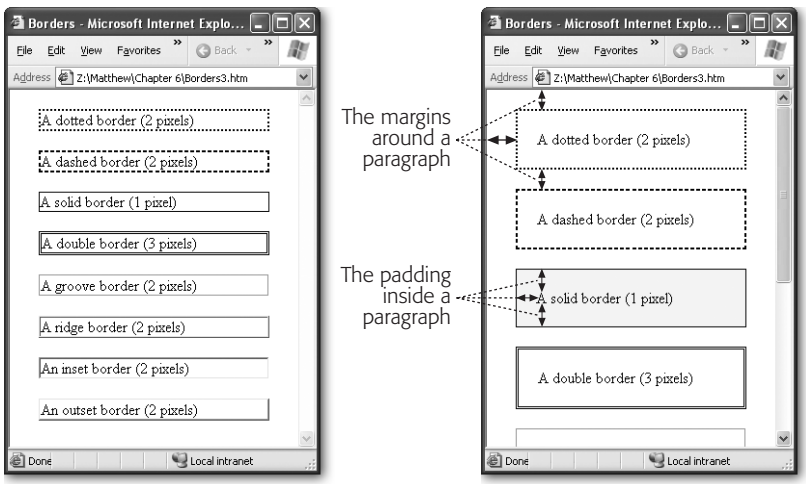

Figure 6-13: *Usually, you can't tell the difference between margins and padding, because you can't see the edges of the element. For example, a <p> element displays a paragraph in an invisible box, but you won't see its sides. When you add a border, this changes.*

Left: These boxes have some extra margin spacing, but no additional padding.

Right: The result is much better when you increase both the margin and padding. For added effect, throw in a light background color (like the solid border box shown here).

Using Borders to Separate Sections

In Chapter 5 (page 119), you learned about the unremarkable <hr> element, which gives you a quick and easy way to separate one section of text from another with a horizontal line. With style sheets, you get several more ways to create attractive separators.

The first line of attack is to style the <hr> element itself. You can use the *width* property to shrink the separator horizontally. You specify width in terms of the percentage of a line's full span. For example, here's a half-width line centered on a page:

```
hr {
  width: 50%;
}
```

You can also thicken the line by using the *height* property and supplying a thickness in pixels. Here's a thick line:

```
hr {
  height: 5px;
}
```

For a variety of more interesting effects, you can bring borders into the mix. For example, here's a rule that thickens the horizontal line, applies the *double border* style, and adopts a modern light gray color:

```
hr {
  height: 3px;
  border-top-width: 3px;
  border-top-style: double;
  border-top-color: #D8D8D8;
}
```

This gives you a quick way to revitalize all your separators. However, if you aren't already using the <hr> element, you don't need to start now.

Another option is to bind the horizontal line to another element, like a heading. For example, the following <h1> element adds a grooved line at the top of a heading. The *margin* property sets the space between the line and previous element, while the padding sets the space between the line and the heading text:

```
h1 {
  margin-top: 30px;
  margin-bottom: 20px;
  padding-top: 10px;
  border-top-width: 2px;
  border-top-style: groove;
}
```

Figure 6-14 shows both of these examples.

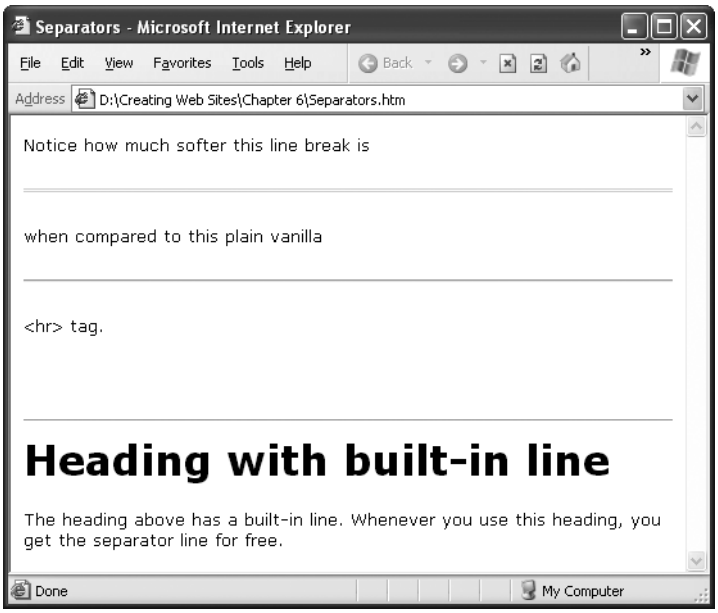

Figure 6-14:
This document includes (from top to bottom), a customized <hr> line, a normal <hr> separator, and an <h1> heading with a top border.

Class Selectors

So far, you've seen how formatting rules apply to every occurrence of a specific XHTML element (except when you use an inline style). The selectors in these universal styles are known as *type selectors*.

Type selectors are powerful, but not that flexible. Sometimes you need a little more flexibility to modify subsections or small portions of an XHTML document. Fortunately, style sheets have the perfect solution with *class selectors*.

Class selectors are one of the most practical style sheet tricks around. They let you separate your rules from your elements, and use them wherever you please. The basic idea is that you carve your Web page content into conceptual groups, or *classes*. Once you take this step, you can apply different formatting rules to each class. The trick is to choose where you want to use each class in your Web page. For example, you might have two identical <h1> headings, but assign them to unique classes so you can format each heading differently.

For a more detailed example, consider the page shown in Figure 6-15. In the following sections, you'll work with this example to apply class-based style rules.

Creating Class Rules

To use classes, you begin by mentally dividing your page into different kinds of content. In this case, it makes sense to create a specialized class for book reviews and an author byline.

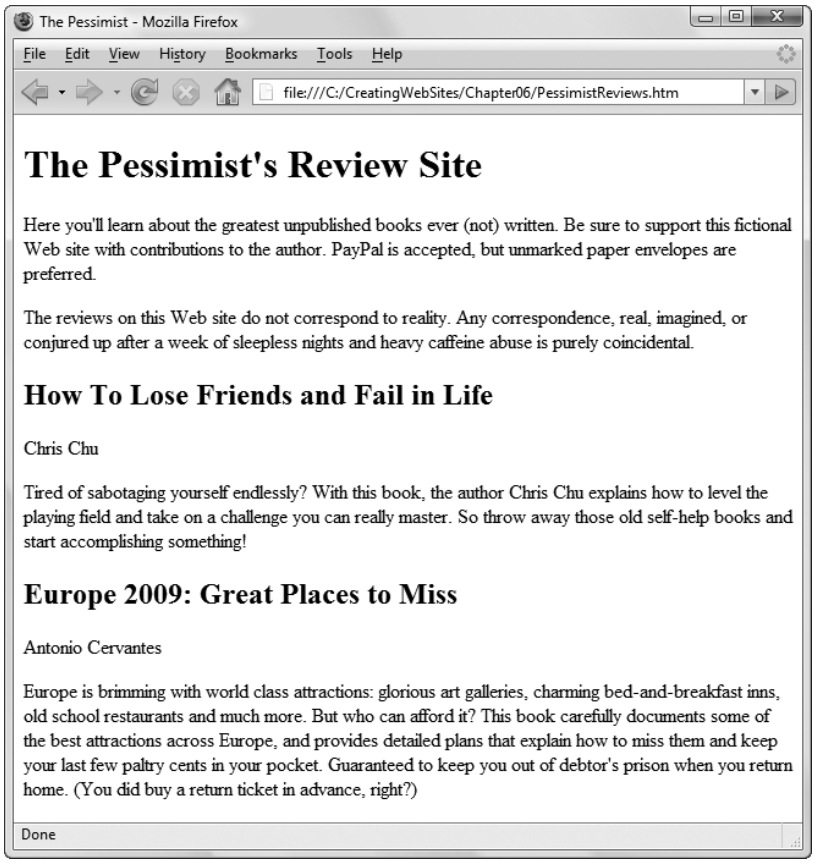

Figure 6-15: *In the average XHTML document, you have a sea of similar elements—even a complex page often boils down to just headings and paragraph elements. This page has a general introduction followed by a series of book reviews. The general introduction, the author credits, and the book summaries are all marked up with <p> elements, but they shouldn't have the same formatting because they represent different types of content. A better approach is to format the different types of content (title, author, and description) in different ways.*

To create a class-specific rule, you use a two-part name, like this:

```
p.review {
  ...
}
```

The first part of the name indicates the element that the rule applies to—in this case, the paragraph element. The second part (the part after the period) is the class name. You can choose whatever class name you want, as long as you stick to letters, digits, and dashes, and make sure that the first character is always a letter.

The point of the class name is to provide a succinct description of the type of content you want to format. In this example, the class name is *review*, because you're going to apply this style to all the paragraphs that are book reviews.

Tip: Good class names describe the *function* of the class rather than its appearance. For example, *WarningNote* is a good class name, while *BoldRedArialBox* isn't. The problem with the latter is that it won't make sense if you decide to change the formatting of your warning note box (for example, giving it red lettering).

So how does a browser know when to apply a rule that uses a class selector? It turns out that browsers never apply class rules automatically. You have to add the class names in your XHTML file to the element you want formatted. Here's an example that applies the review class to a paragraph:

```
<p class="review">The actual review would go right here.</p>
```

As long as the class name in the element matches a class name in the style sheet, the browser applies the formatting. If the browser can't find a style with a matching class name, nothing happens.

Note: Class rules work *in addition* to any other rules. For example, if you create a rule for the <p> element, that rule applies to all paragraphs, including those that are part of a specialized class. However, if the class rule conflicts with any other rules, the class rule wins.

Here's the complete style sheet you might use to format the book review page:

```
/* Set the font for the whole page. */
body {
  font-family: Georgia,serif;
}

/* Set some standard margins for paragraphs. */
p
{
  margin-top: 2px;
  margin-bottom: 6px;
}

/* Format the heading with a background color. */
h1 {
  background-color: #FDF5E6;
  padding: 20px;
  text-align: center;
}

/* Make the bylines small and italicized. */
p.byline {
  font-size: 65%;
  font-style: italic;
  border-bottom-style: outset;
  border-bottom-width: 1px;
  margin-bottom: 5px;
  margin-top: 0px;
}
```

```
/* Make book reviews a little smaller, and justified. */
p.review {
  font-size: 83%;
  text-align: justify;
}

/* Make the review headings blue. */
h2.review {
  font-size: 100%;
  color: blue;
  margin-bottom: 0px;
}
```

This style sheet includes three type selector rules. The first formats the <body> element, thereby applying the same font to the whole Web page. The second gives every <p> element the same margins, and the third changes the alignment and background color of <h1> headings. Next, the style sheet defines two new paragraph classes—one for the author byline, and one for the review text. Lastly, the sheet creates a class for the review headings.

This example also introduces another feature—CSS comments. CSS comments don't look like XHTML comments. They always start with the characters /* and end with the characters */. Comments let you document what each class represents. Without comments, it's all too easy to forget what each style rule does in a complicated style sheet.

And here's the XHTML markup that uses the classes defined in the above style sheet. (To save space, most of the text is left out, but the essential structure is there.)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

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

<head>
  <link rel="stylesheet" href="PessimistReviews1.css"
   type="text/css" />
  <title>The Pessimist</title>
</head>

<body>

  <h1>The Pessimist's Review Site</h1>
  <p>...</p>
  <p>...</p>
  <h2 class="review">How To Lose Friends and Fail in Life</h2>
  <p class="byline">Chris Chu</p>
  <p class="review">...</p>
```

```
    <h2 class="review">Europe 2009: Great Places to Miss</h2>
    <p class="byline">Antonio Cervantes</p>
    <p class="review">...</p>

</body>
</html>
```

Figure 6-16 shows the result.

Figure 6-16: *Class rules let you format different parts of a document differently, even if they use the same element (like the ever-common <p> element).*

Note: Creating style sheets is an art and takes a fair bit of practice. To make the best use of them, you need to become comfortable with class rules. Not only do class rules give you complete flexibility, they also help you think in a more logical, structured way about your Web pages.

Saving Work with the Element

It can get tedious applying the class attribute to every element you want to format in your Web page. Fortunately, there's a great shortcut, courtesy of the element.

You may remember the element from the previous chapter (page 122). It's a block element that lets you group together arbitrary sections of your Web page.

You can group as many elements with the element as you want, including headings, paragraphs, lists, and more.

Thanks to style sheet inheritance (page 148), if you apply a class name to a element, inheritance automatically applies the defined style to all the nested elements. That means you can change this:

```
<p class="review">...</p>
<p class="review">...</p>
<p class="review">...</p>
```

To this:

```
<div class="review">
  <p>...</p>
  <p>...</p>
  <p>...</p>
</div>
```

Essentially, when you format this element, all the paragraphs inside of it inherit that format. And although there are some style properties (like margin and padding) that don't support inheritance, most do. Figure 6-17 shows this example.

The element is a great way to save loads of time. Web experts use it regularly.

More Generic Class Rules

You can also create a rule that has a class name but doesn't specify a type of element. All you need to do is leave the first part of the selector (the portion before the period) blank. Here's an example:

```
.emphasize {
  color: red;
  font-weight: bolder;
}
```

The great thing about a rule like this is that you can use it with *any* element, so long as you use the right class name. In other words, you can use it to format paragraphs, headings, lists, and more with bold, red lettering. The class name in this example reflects its more general-purpose use. Instead of indicating the type of *content*, it indicates the type of *formatting*.

Most Web designers use both element-specific class rules, and more generic class rules that don't specify an element. Although you could stick exclusively with generic rules, if you know that you'll use a certain set of formatting options only with a specific type of element, it's good to clearly indicate this fact with an element-specific class rule. That way, you won't forget the purpose of the rule when you edit your style sheet later on.

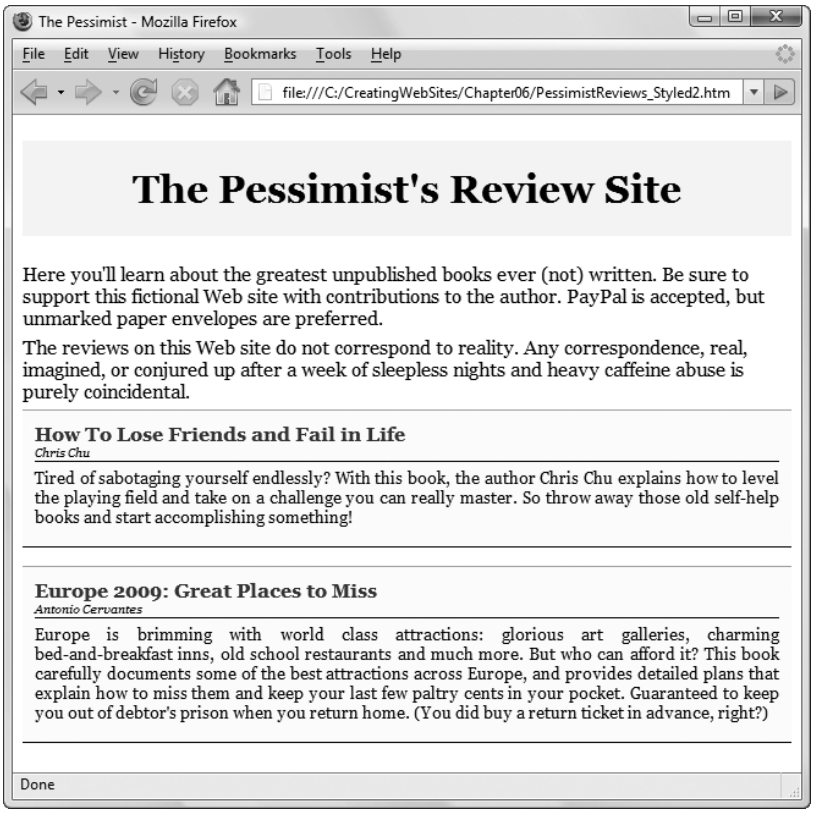

Figure 6-17: *In this example, the XHTML markup wraps each review in a element. The element applies a background color and some borders, separating the reviews from the rest of the page. Techniques like these can help organize dense pages with lots of information.*

Note: There are still a few more advanced types of selectors you haven't considered yet. For example, you can use selectors that target certain types of elements when they appear inside another, specific element. Selectors like these are useful when you start using your style sheet mojo to create sophisticated page layouts, and you'll learn about them in Chapter 9.

Creating a Style Sheet for Your Entire Web Site

Class rules aren't just useful for separating different types of content. They're also handy if you want to define rules for your entire site in a single style sheet.

In a typical Web site, you'll have pages or groups of pages you want to format differently. For example, you might have a page with your résumé, several pages chronicling your trip to Guadeloupe, and another group of pages that make up an online photo gallery. Rather than create three style sheets, you can create a single style sheet that handles everything. The trick is to use different class names for each

section. In other words, you'll create a résumé class, a trip diary class, and a photo gallery class. Here's a basic outline of this approach:

```
/* Used for the resume pages. */
p.resume { ... }
h1.resume { ... }
h2.resume { ... }
...
/* Used for the trip diary pages. */
p.trip { ... }
h1.trip { ... }
h2.trip { ... }
...
/* Used for the online photo gallery. */
p.gallery { ... }
h1.gallery { ... }
h2.gallery { ... }
...
```

Obviously, each page will use only a few of these rules. However, it's often easier to maintain your site when you keep your styles together in one place.