

**Figure 5-1:**  
Left: This page is plain text, but ready for style sheets. It's been carefully separated into logical sections.

Right: With the application of a style sheet, the page's formatting and layout change dramatically. You'll see an example of this in Chapter 9 (page 253).

## XHTML Elements for Basic Text

As you learned in Chapter 2, there are two things you need to know about every new element you meet. To use the element correctly, without violating the rules of XHTML, you need to answer these two questions:

- Is it a container element or a standalone element?
- Is it a block element or an inline element?

The first question tells you something about the syntax you use when you add an element to a document. Container elements (like the `<b>` element that boldfaces text) require a start tag and an end tag, with the content sandwiched in between. Standalone elements (like the `<img>` element that inserts an image into a page) use a single, all-in-one tag. If standalone elements need additional information, like the location of an image file, you supply it using attributes.

The second question tells you something about *where* you can place an element. Block elements (like the `<p>` element) go inside the main `<body>` element or within other block elements. When you start building the overall structure of your Web page, you always begin with block elements. Inline elements (like the `<img>` element) have to go *inside* block elements. Inline elements don't make sense when they're on their own, floating free of any container.

---

**Tip:** To quickly check if an element is a container or standalone element, and to see if it's a block or inline element, check the XHTML reference in Appendix A.

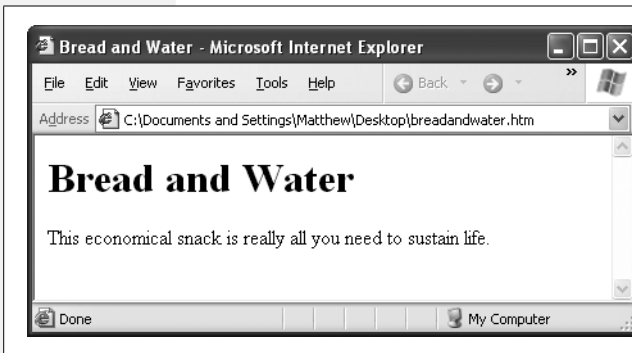
---

Block elements also have an effect on the spacing of your content. Essentially, each block element defines a chunk of content. When you end a block element, your browser automatically adds a line break and a little extra space before the next bit of content.

For example, consider this fragment of XHTML:

```
<h1>Bread and Water</h1><p>This economical snack is really  
all you need to sustain life.</p>
```

This snippet has a title in large, bold letters followed immediately by a paragraph of ordinary text. You might expect to see both parts (the heading and the ordinary text) on the same line. However, the `<h1>` element is a block element. When you close it, the browser does a little housecleaning, and adds a line break and some extra space. The paragraph text starts on a new line, as you can see in Figure 5-2.



**Figure 5-2:**  
*XHTML separates block elements by a distance of approximately one and a half lines (in this figure, that's the space between "Bread and Water" and the sentence below it).*

---

**Tip:** Block elements are nice because they make it easy to format a document. For example, the spaces that exist between block elements help ensure that one section of text doesn't run into another. However, there's also a clear downside. In some cases, you won't be happy with the automatic spacing between block elements. For example, for dense, information-laden pages, the standard spacing looks far too generous. To tighten up your text and shrink the spaces in between block elements, use style sheets to change the margin settings of your elements (page 163).

---

Now that you've learned about the basic types of elements, it's time to take a look through your element toolkit.

## Paragraphs

You've already seen the basic paragraph element, `<p>`. It's a block element that defines a paragraph of text.

```
<p>It was the best of times, it was the worst of times...</p>
```

As you've no doubt noticed by now in your travels across the Internet, XHTML paragraphs aren't indented as they are in print media. That's just the way of the Web, although you can change this with style sheets (page 163). Figure 5-3 shows an example of paragraph elements in action.

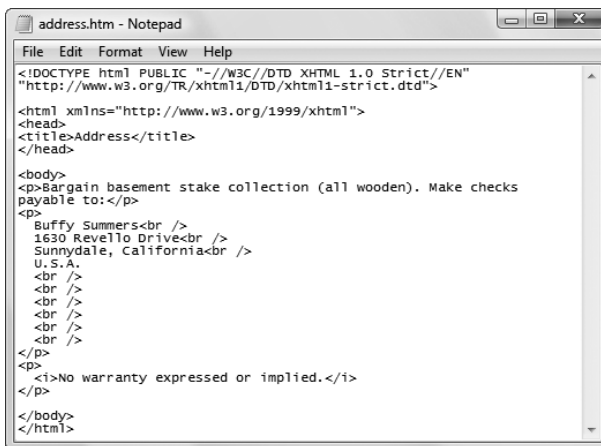


## Line Breaks

Sometimes you want to start a new line of text, but you don't want to use a paragraph element because browsers add extra space between paragraphs. This is the case, for example, when you want to include a business address on your site and you want it to appear in the standard single-spaced three-line format. In situations like this, the standalone line break element `<br />` comes in handy.

Line breaks are exceedingly simple—they tell a browser to move to the start of the following line (see Figure 5-4). They're inline elements, so you need to use them inside a block element, like a paragraph:

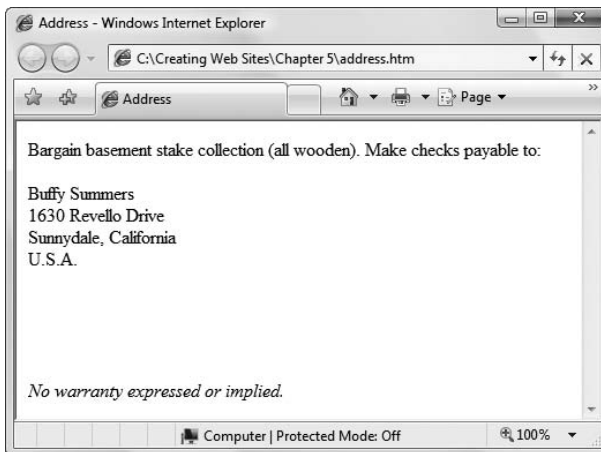
```
<p>This paragraph appears<br />  
on two lines</p>
```



```
address.htm - Notepad
File Edit Format View Help
<!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>
<title>Address</title>
</head>
<body>
<p>Bargain basement stake collection (all wooden). Make checks
payable to:</p>
<p>
  Buffy Summers<br />
  1630 Revello Drive<br />
  Sunnydale, California<br />
  U.S.A.
<br />
<br />
<br />
<br />
<br />
<br />
</p>
<p>
<i>No warranty expressed or implied.</i>
</p>
</body>
</html>
```

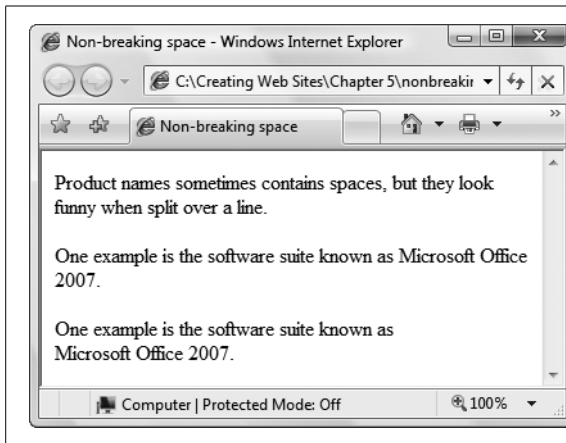
**Figure 5-4:**

*The line break element `<br />` is great for separating addresses. If you want to skip down several lines, you can use a series of `<br />` elements in a row (but it's a better idea to use empty paragraphs, as described in the box on page 117).*



Don't overuse line breaks. Remember, when you resize a browser window, the browser reformats your text to fit the available space. If you try to perfect your paragraphs with line breaks, you'll end up with pages that look bizarre at different browser window sizes. A good rule of thumb is to avoid line breaks in ordinary paragraphs. Instead, use them to force breaks in addresses, outlines, poems, and other types of text whose spacing you want to tightly control. Don't use them for bulleted or numbered lists, either—you'll learn about elements designed just for these lists on page 123.

In some cases, you want to *prevent* a line break, like when you want to keep the longish name of a company or a product on a single line. The solution is to use the nonbreaking space code (which looks like `&nbsp;`) instead of just hitting the space bar. The browser still displays a space when it gets to the `&nbsp;` code, but it won't wrap the words on either side of it (see Figure 5-5).



**Figure 5-5:**

*Paragraphs 2 and 3 in this figure show how the `&nbsp;` code affects line breaks. Paragraph 3 is actually coded as `Microsoft Office 2007`. As a result, the browser won't split this term.*

#### HOW'D THEY DO THAT?

### The Mystery of Empty Paragraphs

In Web authoring tools like Dreamweaver and Expression Web, if you're in Design view and you press Enter, the program creates a new paragraph. This seems a little counter-intuitive, as you've seen that browsers normally ignore line breaks (see Figure 5-5).

The trick is that when you hit the Enter key, both programs insert a paragraph that contains a nonbreaking space. Here's what that creation looks like:

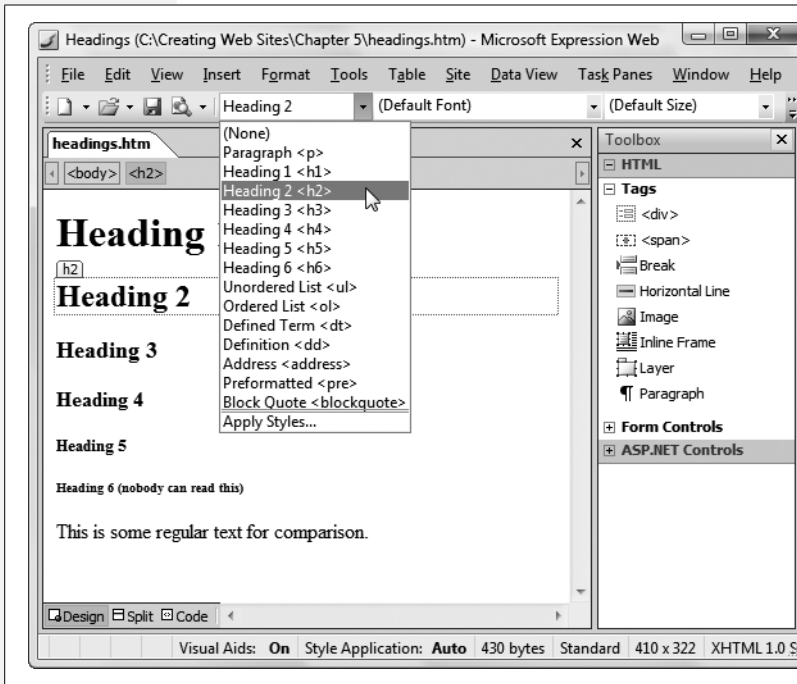
```
<p>&nbsp;</p>
```

This paragraph is still empty, but the browser won't ignore it because it includes the `&nbsp;` code. Therefore, the browser gives it the same space as a single-line paragraph and bumps down the content underneath.

Incidentally, Dreamweaver and Expression Web do let you use more ordinary `<br />` line break elements instead of empty paragraphs, even in Design view. To do this, press Shift+Enter instead of Enter.

## Headings

Headings are section titles—for example, the word “Headings” just above this paragraph. Browsers display them in boldface at various sizes. The size depends on the *heading level*. XHTML supports six heading levels, starting at <h1> (the biggest) and dwindling down to <h6> (the smallest). Both <h5> and <h6> are actually smaller than regularly sized text, and Web developers don’t use them too often. Figure 5-6 shows all the heading levels you can use.



**Figure 5-6:** Many Web page editors let you apply headings with a single click. In Expression Web, you can find a drop-down list that lets you choose whether to make the currently selected text a paragraph or one of the various headings, as shown here. In Dreamweaver, you can use the handy buttons in the Text tab of the Insert toolbar.

Headings aren’t just useful for formatting—they also help define the hierarchy of your document. Big headings identify important topics, while smaller ones denote lesser issues related to that larger topic. To make sure your document makes sense, start with the largest headings (level 1) and work your way down. For instance, don’t jump straight to a level-3 heading just because you like the way it looks.

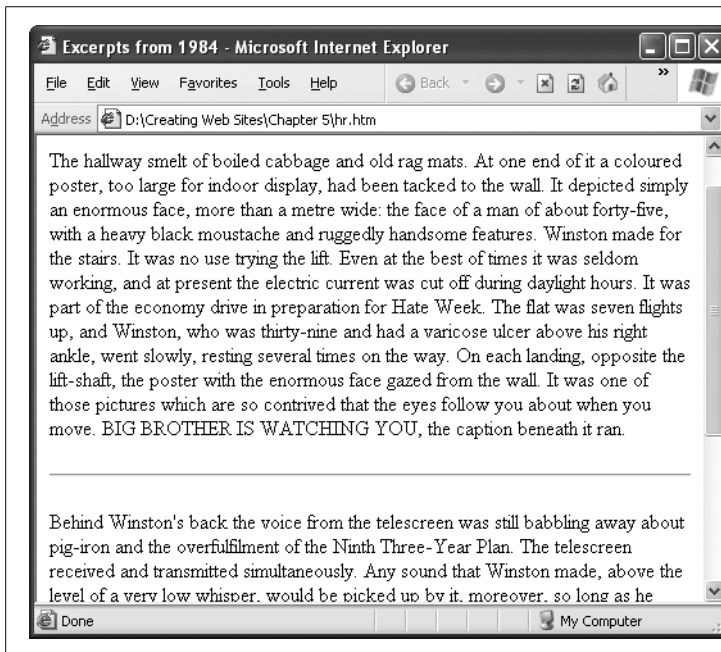
**Note:** It’s probably occurred to you that if everyone uses the same heading levels in the same order, the Web will become as bland as a bagel in a chain supermarket. Don’t panic—it’s not as bad as it seems. When you add style sheets into the mix, you’ll see that you can completely change the look of any and every heading you use. So for now, stick to using the right levels in the correct order.

## Horizontal Lines

Paragraphs and line breaks aren't the only way to separate sections of text. Another neat trick is the standalone `<hr>` element, which translates to "horizontal rule." A horizontal rule element adds a line that stretches from one side of its container to the other, separating everything above and below it.

**Tip:** Usually, you'll position a horizontal break between paragraphs, which means it will stretch from one side of a page to the other. However, you can also put a horizontal rule in a smaller container, like a single cell in a table, in which case it won't turn out nearly as big.

Horizontal rules are block elements, so you can stick them in between paragraphs (see Figure 5-7).



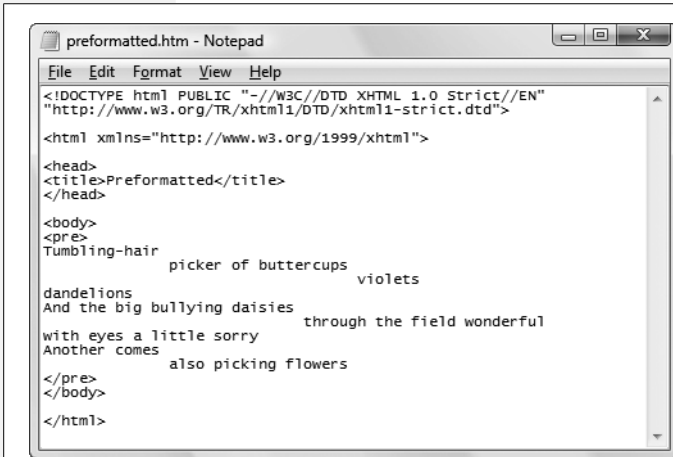
**Figure 5-7:**  
In this example, two paragraphs have an `<hr>` element between them. The `<hr>` element inserts the solid line you see.

## Preformatted Text

Preformatted text is a unique concept in XHTML that breaks the rules you've read about so far. As you've seen, Web browsers ignore multiple spaces and flow your text to fit the width of a browser window. Although you can change this to a certain extent by using line breaks and nonbreaking spaces, some types of content are still hard to deal with.

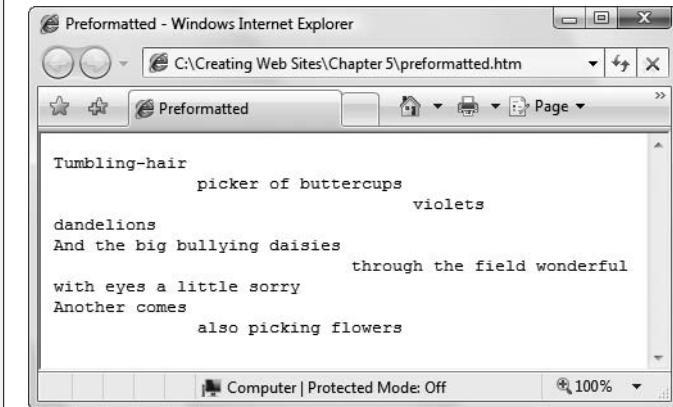
For example, imagine you want to display a bit of poetry. Using nonbreaking spaces to align the text is time-consuming and makes your XHTML markup difficult to read. The `<pre>` element gives you a better option. It tells your browser to re-create the text just as you entered it, including every space and line break, and it displays these details on-screen. Additionally, the browser puts all that text into a monospaced font (typically Courier). Figure 5-8 shows an example.

**Note:** In a *monospaced* font, every letter occupies the same amount of space. XHTML documents and books like this one use proportional fonts, where letters like W and M are much wider than l and i. Monospaced fonts are useful in preformatted text, because it lets you line up rows of text exactly. However, it doesn't look as polished.



```
preformatted.htm - Notepad
File Edit Format View Help
<!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>
<title>Preformatted</title>
</head>
<body>
<pre>
Tumbling-hair
    picker of buttercups      violets
dandelions
And the big bullying daisies      through the field wonderful
with eyes a little sorry
Another comes      also picking flowers
</pre>
</body>
</html>
```

**Figure 5-8:** There's no mystery as to how this e. e. cummings poem will turn out. Because it's in a `<pre>` block, you get the exact spacing and line breaks that appear in your XHTML file. The `<pre>` element also works well for blocks of programming code.





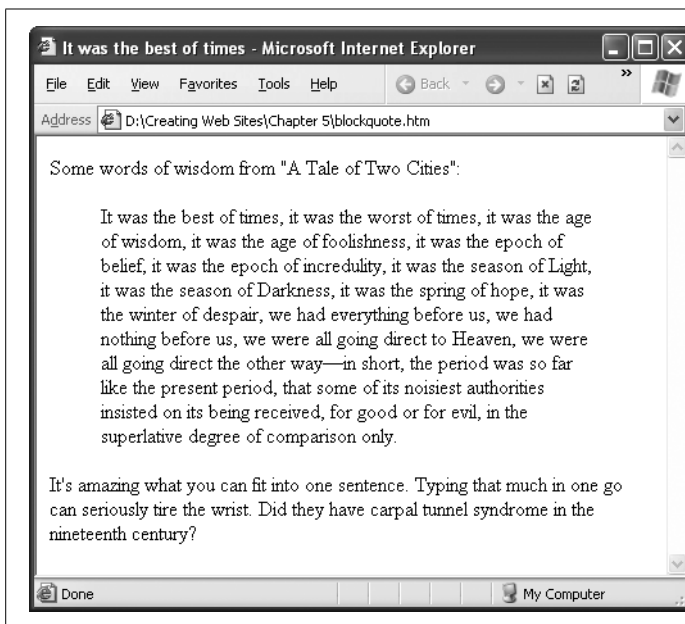
## Quotes

It may be a rare Web page that spouts literary quotes, but the architects of XHTML created a block element named `<blockquote>` especially for long quotations. When you use this element, your browser indents text on the left and right edges.

Here's an example:

```
<p>Some words of wisdom from "A Tale of Two Cities":</p>
<blockquote>
<p>It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness, it was the epoch of belief, it was
the epoch of incredulity, it was the season of Light, it was the season of
Darkness, it was the spring of hope, it was the winter of despair, we had
everything before us, we had nothing before us, we were all going direct to
Heaven, we were all going direct the other way—in short, the period was so
far like the present period, that some of its noisiest authorities insisted
on its being received, for good or for evil, in the superlative degree of
comparison only.</p>
</blockquote>
<p>It's amazing what you can fit into one sentence.</p>
```

Figure 5-9 shows how this appears in the browser.



**Figure 5-9:**  
Here, the `<blockquote>` element indents the middle paragraph.

Occasionally, people use the `<blockquote>` element purely for its formatting capability—they like the way it sets off text. Of course, this compromises the spirit of the element, and you'd be better off to use style sheets to achieve a similar effect. However, it's a fairly common technique, so it's more or less accepted.

The `<blockquote>` element is a block element, which means it always appears separately from other block elements, like paragraphs and headings. The `<blockquote>` has one further restriction—it can hold only other block elements, which means you need to put your content into paragraphs rather than simply type it in between the `blockquote` start and end tags.

If, instead of using a quote that runs a paragraph or longer, you want to include a simple one-line quote, XHTML's got you covered. It defines an inline element for short quotes that you can nest inside a block element. It's the `<q>` element, which stands for quotation:

```
<p>As Charles Dickens once wrote, <q>It was the best of times, it was the  
worst of times</q>.</p>
```

Some browsers, like Firefox, add quotation marks around the text in a `<q>` element. Other browsers, like Internet Explorer, do nothing. If you want your quotation to stand out from the text around it in every browser, you might want to add some different formatting, like italics. You can do this by applying a style sheet rule (see Chapter 6).

And if you're dreaming of the semantic Web (see the box on page 111), you can add a URL that points to the source of your quote (assuming it's on the Web) using the `cite` attribute:

```
<p>As Charles Dickens once wrote, <q cite="http://www.literature.org/  
authors/dickens-charles/two-cities">It was the best of times, it was the  
worst of times</q>.</p>
```

Looking at this example, you might expect your browser to provide some sort of feature that takes you to the referenced Web site (for example, when you click the paragraph). But it doesn't. If you want your text to link to a reference, you need to investigate the anchor element in Chapter 8.

In fact, the information in the `cite` attribute won't appear on your page at all. It *is* available to programs that analyze your Web page—for example, automated programs that scan pages and compile a list of references, or a search engine that uses this information to provide better search results. But most of the time, the reference has little benefit, except that it stores an extra piece of information that you, the Web site creator, might need later to double-check your sources.

## Divisions and Spans

The last block element you'll learn about—`<div>`—is one of the least interesting, at least at first glance. That's because, on its own, it doesn't actually *do* anything.

You use `<div>` to group together one or more block elements. That means you could group together several paragraphs, or a paragraph and a heading, and so on. Here's an example:

```
<div>
  <h1>...</h1>
  <p>...</p>
  <p>...</p>
</div>
<p>...</p>
```

Given the fact that `<div>` doesn't do anything, you're probably wondering why it exists. It turns out that the lowly `<div>` tag becomes a lot more interesting when you combine it with style sheets. That's because you can apply formatting commands directly to a `<div>` element. For example, if a `<div>` element contains three paragraphs, you can format all three paragraphs at once simply by formatting the `<div>` element.

The `<div>` element has an important relative—the `<span>` element. Like its cousin, the `<span>` element doesn't do anything on its own, but when you place it *inside* a block element and define its attributes in a style sheet, you can use it to format just a portion of a paragraph, which is very handy. Here's an example:

```
<p>In this paragraph, some of the text is wrapped in a span element. That
<span>gives you the ability</span> to format it in some fancy way later on.
</p>
```

You'll put the `<div>` and `<span>` elements to good use in later chapters.

## XHTML Elements for Lists

Once you master XHTML's basic text elements, it's time to move on to XHTML's other set of elements for organizing text—list elements. XHTML lets you create three types of list:

- **Ordered lists** give each item in a list a sequential number (as in 1, 2, 3). They're handy when sequence is important, like when you list a series of steps that tell your relatives how to drive to your house.
- **Unordered lists** are also known as bulleted lists, because a bullet appears before each item in the list. To some degree, you can control what the bullet looks like. You're reading a bulleted list right now.
- **Definition lists** are handy for displaying terms followed by definitions or descriptions. For example, the dictionary is one huge definition list. In a definition list on a Web page, your browser left-aligns the terms and indents the definitions underneath them.

In the following sections you'll learn how to create all three types of list.

## Webifying Your Text

As you learned earlier in this chapter, text on the Web isn't like text in print. But sometimes it's hard to shake old habits. Here are some unwritten rules that can help make sure you're making good use of text in your Web pages:

- **Split your text into small sections.** Web pages (and the viewers who read them) don't take kindly to long paragraphs.
- **Create short pages.** If a page is longer than two screenfuls, split it into two pages. Not only does this make your pages easier to read, it gives you more Web pages, which helps with the next point.
- **Divide your content into several pages.** The next step is to link these pages together (see Chapter 8). This gives readers the flexibility to choose what they want to read, and in what order.
- **Put your most important information in the first screenful.** This technique is called designing *above the fold*. The basic idea is to make sure there's something eye-catching or interesting for visitors to read without having to scroll down. (In the same way, well-designed newspapers give newsstand visitors something interesting to read without them having to flip over the folded broadsheet, hence the term "above the fold".)
- **Proofread, proofread, proofread.** Typos and bad grammar shatter your site's veneer of professionalism and Web-coolness.
- **Don't go wild with formatting until you understand style sheets.** If you break this rule, you'll leave a big mess that you'll only need to clean up later on.

## Ordered Lists

In an ordered list, XHTML numbers each item consecutively, starting at some value (usually 1). The neat part about ordered lists in XHTML is that you don't need to supply the numbers. Instead, the browser automatically adds the appropriate number next to each list item (sort of like the autonumber feature in Microsoft Word). This is handy for two reasons. First, it lets you insert and remove list items without screwing up your numbering. Second, XHTML carefully aligns the numbers and list items, which isn't as easy if you do it on your own.

To create an ordered list, use `<ol>`, a block element (`<ol>` stands for "ordered list"). Then, inside the `<ol>` element, you place an `<li>` element for each item in the list (`<li>` stands for "list item").

For example, here's an ordered lists with three items:

```
<p>To wake up in the morning:</p>
<ol>
  <li>Rub eyes.</li>
  <li>Assume erect position.</li>
  <li>Turn on light.</li>
</ol>
```

In a browser, you'd see this:

To wake up in the morning:

1. Rub eyes.
2. Assume erect position.
3. Turn on light.

XHTML inserts some space between the paragraph preceding the list and the list itself, as with all block elements. Next, it gives each list item a number.

Ordered lists get more interesting when you mix in the *start* and *type* attributes. The *start* attribute lets you start the list at a value other than 1. Here's an example that starts the counting at 5:

```
<p>To wake up in the morning:</p>
<ol start="5">
...
</ol>
```

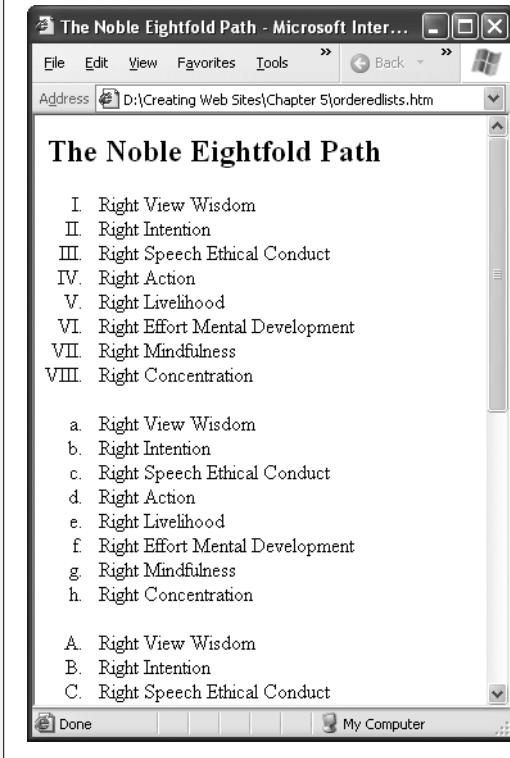
This list will include the numbers 5, 6, and 7. Unfortunately, there's no way to count backward, or to automatically continue counting from a previous list elsewhere on a page.

You aren't limited to numbers in your ordered list, either. The *type* attribute lets you choose the style of numbering. You can use sequential letters and roman numerals, as described in Table 5-1. Figure 5-10 shows a few examples.

**Table 5-1.** Types of ordered lists

type Attribute	Description	Example
1	Numbers	1, 2, 3, 4...
a	Lowercase letters	a, b, c, d...
A	Uppercase letters	A, B, C, D...
i	Lowercase roman numerals	i, ii, iii, iv...
I	Uppercase roman numerals	I, II, III, IV...

Strict XHTML forbids both the *type* and *start* attributes. If you want to use these attributes, you need to stick an XHTML transitional doctype at the top of your Web page. Another option is to switch to styles (using CSS, the standard you'll pick up in Chapter 6). This is a partial solution, because CSS provides an alternative for the *type* attribute but not for the *start* attribute. When you use CSS, you can keep the XHTML strict doctype. Just remove the *style* attribute and replace it with the *list-style-type* CSS property (explained in Chapter 6).



**Figure 5-10:**  
The *type* attribute in action. For example, the code to start off the first list would be: `<ol type="I">`.

## Unordered Lists

Unordered lists are similar to ordered lists except that they aren't consecutively numbered or lettered. The outer element is `<ul>`, and you wrap each item inside an `<li>` element. The browser indents each item in the list, and automatically draws the bullets.

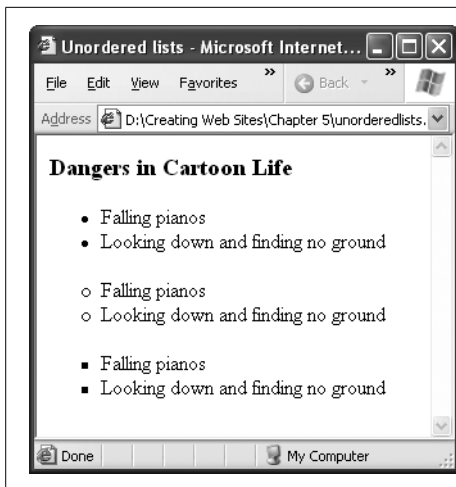
The most interesting frill that comes with unordered lists is the *type* attribute, which lets you change the style of bullet. You can use *disc* (a black dot, which is automatic), *circle* (an empty circle), or *square* (a filled-in square). Figure 5-11 shows the different styles.

Once again, the *type* attribute is only suitable for XHTML 1.0 transitional—if you're going strict, you need to switch to the *list-style-type* style property.

---

**Tip:** Most Web page editors have handy links for quickly creating the different types of lists. In Dreamweaver, look for the "ul" and "ol" icons in the Text tab of the Insert toolbar.

---



**Figure 5-11:**  
Three flavors of the same list.

## Definition Lists

Definition lists are perfect for creating your own online glossary. Each list item actually has two parts—a term (which the browser doesn't indent) and a definition (which the browser indents underneath the term).

Definition lists use a slightly different tagging system than ordered and unordered lists. First, you wrap the whole list in a dictionary list element (`<dl>`). Then you wrap each term in a `<dt>` element (dictionary term), and each definition in a `<dd>` element (dictionary definition).

Here's an example:

```
<dl>
<dt>eat</dt>
<dd>To perform successively (and successfully) the functions of mastication,
humectation, and deglutition.</dd>
<dt>eavesdrop</dt>
<dd>Secretly to overhear a catalogue of the crimes and vices of another or
yourself.</dd>
<dt>economy</dt>
<dd>Purchasing the barrel of whiskey that you do not need for the price of
the cow that you cannot afford.</dd>
</dl>
```

In a browser you'd see this:

*eat*

To perform successively (and successfully) the functions of mastication, humectation, and deglutition.

*eavesdrop*

Secretly to overhear a catalogue of the crimes and vices of another or yourself.

*economy*

Purchasing the barrel of whiskey that you do not need for the price of the cow that you cannot afford.

## Nesting Lists

Lists work well on their own, but you can get even fancier by placing one complete list inside another. This technique is called *nesting* lists, and it lets you build multi-layered outlines and detailed sequences of instructions.

To nest a list, declare a new list inside an `<li>` element in an existing list. For example, the following daily to-do list has three levels:

```
<ul>
  <li>Monday
    <ol>
      <li>Plan schedule for week</li>
      <li>Complete Project X
        <ul style="square">
          <li>Preliminary Interview</li>
          <li>Wild Hypothesis</li>
          <li>Final Report</li>
        </ul>
      </li>
      <li>Abuse underlings</li>
    </ol>
  </li>
  <li>Tuesday
    <ol>
      <li>Revise schedule</li>
      <li>Procrastinate (time permitting). If necessary, put off
        procrastination until another day.</li>
    </ol>
  </li>
  <li>Wednesday
    ...
</ul>
```

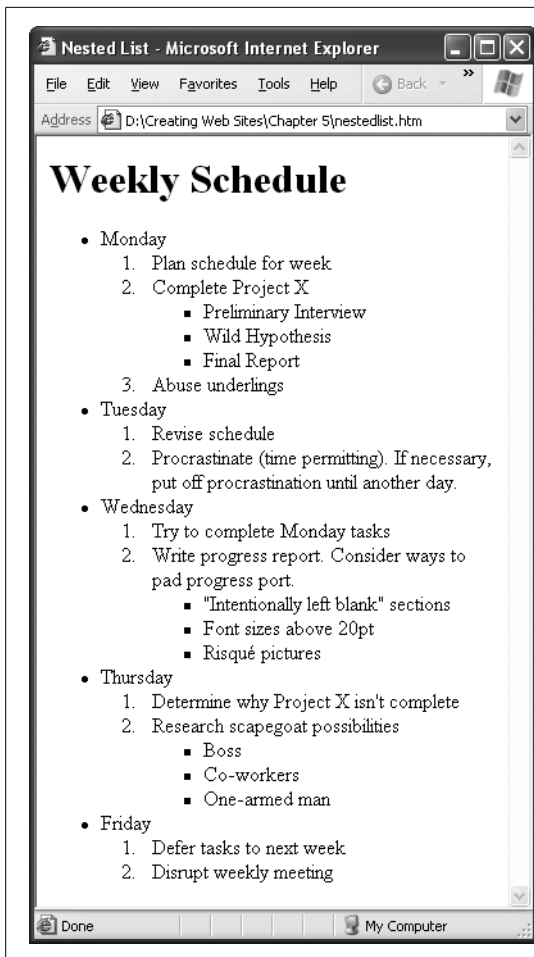
---

**Tip:** When using nested lists, it's a good idea to use indents in your XHTML document so you can see the different levels at a glance. Otherwise, you'll find it difficult to determine where each list item belongs.

---

In a nested list, the different list styles really start to become useful for distinguishing each level. Figure 5-12 shows the result of this example.



**Figure 5-12:**

*In a nested list, browsers indent each subsequent list. Although you aren't limited in the number of levels you can use, you'll eventually run out of room and force your text up against the right side of the page.*

## Inline Formatting

As you learned earlier in this chapter, it's best not to format XHTML too heavily. To get maximum control and make it easy to update your Web site's look later on, you should head straight to style sheets (as described in the next chapter). However, a few basic formatting elements are truly useful. You're certain to come across them, and you'll probably want to use them in your own pages. These elements are all inline elements, so you use them inside a block element, like a paragraph, a heading, or a list.

### Italics, Bold, and Underline

You've already seen the elements for bold (`<b>`) and italic (`<i>`) formatting in Chapter 2. They're staples in XHTML, letting you quickly format snippets of text.

XHTML also has a `<u>` element for underlining text, but you can only use it in XHTML 1.0 transitional (page 30). Here's an example that uses all three elements—`<i>` for italics, `<b>` for bold, and `<u>` for underline:

```
<p>
<b>Stop!</b> The mattress label says <u>do not remove under penalty
of law</u> and you <i>don't</i> want to mess with mattress companies.
</p>
```

A browser displays it like this:

**Stop!** The mattress label says do not remove under penalty of law and you *don't* want to mess with mattress companies.

If you keep your pages clean with XHTML 1.0 strict, you can't use the `<u>` element. However, you can get exactly the same effect using text decorations in a style sheet. Page 155 shows you how.

## Emphasis and Strong

The `<em>` element (for emphasized text) is the logical-element equivalent of the physical element `<i>`. These two elements have the same effect—they both italicize text. Philosophically, the `<em>` element is a better choice, because it's more generic. When you use `<em>`, you're simply indicating that you want to emphasize a piece of text, but you aren't saying *how* to emphasize it. Later on, you can use a style sheet to define just how browsers should emphasize it. Possibilities include making it a different color, a different font, or a different size. If you don't use a style sheet, the text inside the `<em>` element is set in italics, just as with the `<i>` element.

---

**Note:** Technically, you can use style sheets to redefine the `<i>` element in the same way. However, it seems confusing to have the `<i>` element do anything except apply italics. After all, that's its name.

---

The `<strong>` element is the logical-element equivalent of the physical element `<b>`. If you aren't using style sheets, this simply applies bold formatting to a piece of text. Overall, Web developers more commonly use the `<i>` and `<b>` elements over `<em>` and `<strong>`, but XHTML experts prefer the latter because they're more flexible.

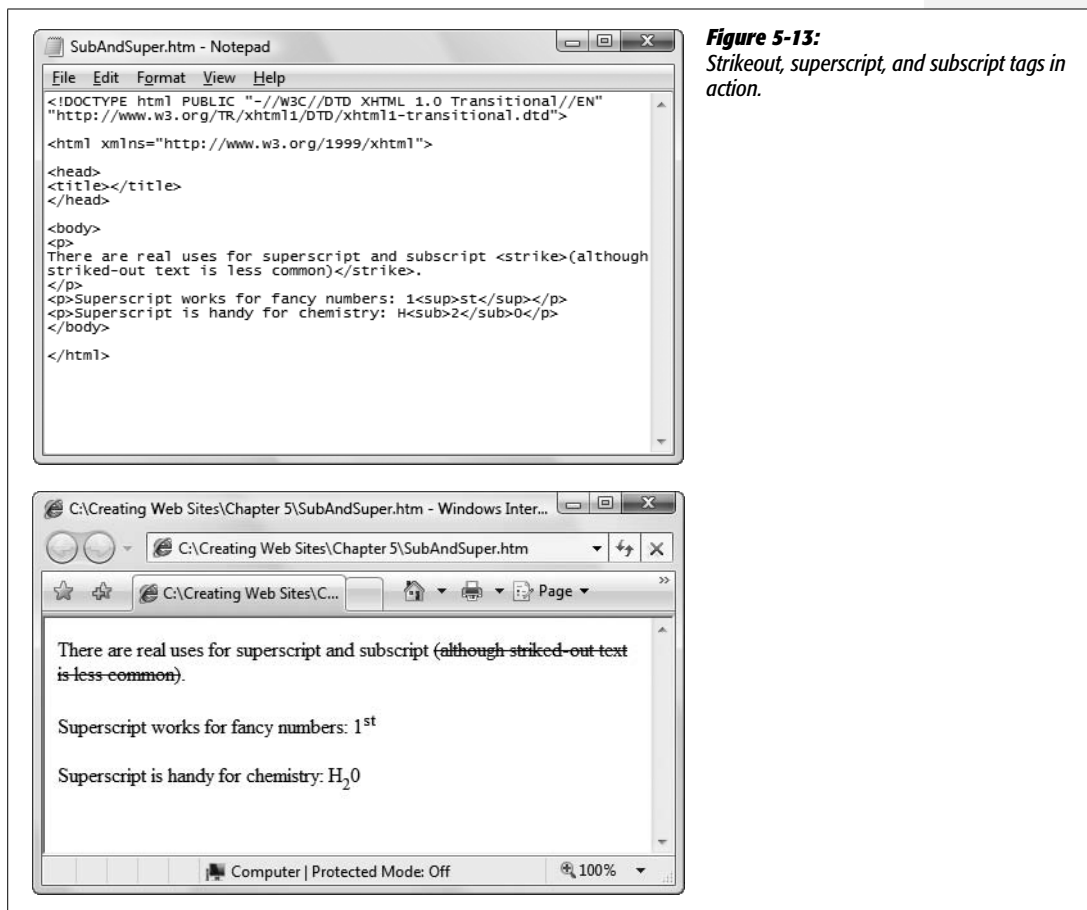
Here's the previous example rewritten to use the `<em>` and `<strong>` elements:

```
<p>
<strong>Stop!</strong> The mattress label says <u>do not remove under penalty
of law</u> and you <em>don't</em> want to mess with mattress companies.
</p>
```

There's no logical-element equivalent for the `<u>` underline element, although you can always use one of the generic elements discussed earlier, like `<span>` in conjunction with the *text-decoration* style property (see page 155).

## Subscript, Superscript, and Strikethrough

You can use the `<sub>` element for *subscript*—text that’s smaller and placed at the bottom of the current line. The `<sup>` element is for *superscript*—smaller text at the top of the current line. Finally, wrapping text in a `<strike>` element tells a browser to cross it out, but you can use it only in XHTML 1.0 transitional. Figure 5-13 shows an example of all three.



**Figure 5-13:** Strikeout, superscript, and subscript tags in action.

Web designers who want to stay on the right side of XHTML law can still create crossed-out text. One alternative is to use the rare `<del>` element (which is meant to represent deleted text in a revised document). However, you can’t trust that all browsers will format `<del>` the same way, and you really shouldn’t use it for anything other than highlighting changes. A better approach is to use a style rule that applies the right text decoration, as explained on page 155.

## Teletype

Text within a `<tt>` element appears in a fixed-width (monospaced) font, such as Courier. Programmers sometimes use it for snippets of code in a paragraph.

```
<p>To solve your problem, use the <tt>Fizzle()</tt> function.</p>
```

Which shows up like this:

To solve your problem, use the `Fizzle()` function.

Teletype text (or typewriter text) looks exactly like the text in a `<pre>` block (see page 119), but you should place `<tt>` text inside another block element. Unlike preformatted text, browsers ignore spaces and line breaks in `<tt>` text, as they do in every other XHTML element.

## Special Characters

Not all characters are available directly on your keyboard. For example, what if you want to add a copyright symbol (©), a paragraph mark (¶), or an accented e (é)? Good news: XHTML supports them all, along with about 250 relatives, including mathematical symbols and Icelandic letters. To add them, however, you need to use some sleight of hand. The trick is to use *XHTML character entities*—special codes that browsers recognize as requests for unusual characters. Table 5-2 has some common options, with a sprinkling of accent characters.

**Table 5-2.** Common special characters

Character	Name of Character	What to Type
©	Copyright	&copy;
®	Registered trademark	&reg;
¢	Cent sign	&cent;
£	Pound sterling	&pound;
¥	Yen sign	&yen;
€	Euro sign	&euro; (but &#8364; is better supported)
°	Degree sign	&deg;
±	Plus or minus	&plusmn;
÷	Division sign	&divide;
×	Multiply sign	&times;
μ	Micro sign	&micro;
¼	Fraction one-fourth	&frac14;
½	Fraction one-half	&frac12;
¾	Fraction three-fourths	&frac34;
¶	Paragraph sign	&para;
§	Section sign	&sect;

**Table 5-2.** Common special characters (continued)

Character	Name of Character	What to Type
«	Left angle quote, guillemot left	&laquo;
»	Right angle quote, guillemot right	&raquo;
¡	Inverted exclamation	&iexcl;
¿	Inverted question mark	&iquest;
æ	Small ae diphthong (ligature)	&aelig;
ç	Small c, cedilla	&ccedil;
è	Small e, grave accent	&egrave;
é	Small e, acute accent	&eacute;
ê	Small e, circumflex accent	&ecirc;
ë	Small e, dieresis or umlaut mark	&euml;
ö	Small o, dieresis or umlaut mark	&ouml;
É	Capital E, acute accent	&Eacute;

**Tip:** The euro symbol is a relative newcomer to XHTML. Although you can use the character entity &euro; you'll have the best support using the numeric code &#8364; because it works with older browsers.

XHTML character entities aren't just for non-English letters and exotic symbols. You also need them to deal with characters that have a special meaning according to the XHTML standard—namely angle brackets (< >) and the ampersand (&). You shouldn't enter these characters directly into a Web page because the browser will assume you're trying to give it a super-special instruction. Instead, you need to replace these characters with their equivalent character entity, as shown in Table 5-3.

**Table 5-3.** XHTML character entities

Character	Name of Character	What To Type
<	Left angle bracket	&lt;
>	Right angle bracket	&gt;
&	Ampersand	&amp;
"	Double quotation mark	&quot;

Strictly speaking, you don't need all these entities all of the time. For example, it's safe to insert ordinary quotation marks by typing them in from your keyboard—just don't put them inside attribute names. Similarly, browsers are usually intelligent enough to handle the ampersand (&) character appropriately, but it's better style to use the &amp; code, so that there's no chance a browser will confuse the ampersand with another character entity. Finally, the character entities for the angle brackets are absolutely, utterly necessary.

Here's some flawed text that won't display correctly:

I love the greater than (>) and less than (<) symbols. Problem is, when I type them my browser thinks I'm trying to use a tag.

And here's the corrected version, with XHTML character entities. When a browser processes and displays this text, it replaces the entities with the characters you really want.

I love the greater than (&gt;) and less than (&lt;) symbols. Problem is, when I type them my browser thinks I'm trying to use a tag.

Most Web design tools insert the correct character entities as you type, as long as you're in Design view and not Code view.

---

**Tip:** To get a more comprehensive list of special characters and see how they look in your browser, check out [www.webmonkey.com/reference/Special\\_Characters](http://www.webmonkey.com/reference/Special_Characters).

---

## Non-English Languages

Although character entities work perfectly well, they can be a bit clumsy if you need to rely on them all the time. For example, consider the famous French phrase "We were given speech to hide our thoughts," shown here:

La parole nous a été donnée pour déguiser notre pensée.

Here's what it looks like with character entities replacing all the accented characters:

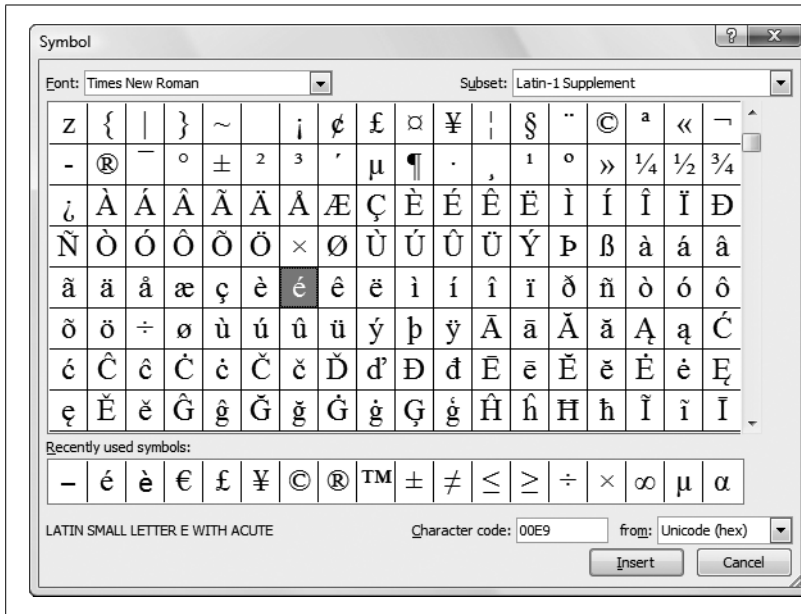
La parole nous a &eacute;&eacute; donn&eacute;e pour d&eacute;guiser notre pens&eacute;e.

French speakers would be unlikely to put up with this for long. Fortunately, there's a solution called *Unicode encoding*. Essentially, Unicode is a system that converts characters into the bytes that computers understand and can properly render. By using Unicode encoding, you can create accented characters just as easily as if they were keys on your keyboard.

So how does it work? First, you need a way to get the accented characters into your Web page. Here are some options:

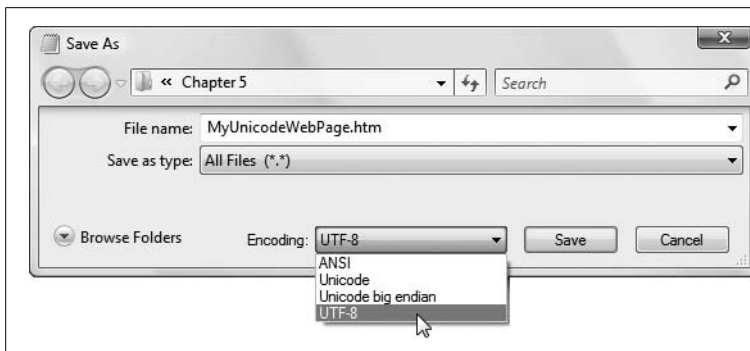
- **Type it in.** Many non-English speakers will have the benefit of keyboards that include accented characters.
- **Use a utility.** In Windows, you can run a little utility called *charmap* (short for Character Map) that lets you pick from a range of special characters and copy your selected character to the clipboard so it's ready for pasting into another program. To run charmap, click Start → Run, type in charmap, and then hit Enter (in Windows Vista, click Start, and then type charmap into the search box).

- **Use your Web page editor.** Some Web page editors include their own built-in symbol pickers. In Expression Web, choose Insert → Symbol (see Figure 5-14). In Dreamweaver, you can use Insert → HTML → Special Characters → Other, but this process only inserts character entities, not Unicode characters. Though the end result is the same, your XHTML markup will still include a clutter of codes.



**Figure 5-14:** Choose Insert → Symbol to see Expression Web’s comprehensive list of special characters. When you pick one, Expression Web inserts the actual character, Unicode-style, not the cryptic character entity.

When using Unicode encoding, you need to make sure you save your Web page correctly. This won’t be a problem if you use a professional Web page editor, which is smart enough to get it right the first time. But Unicode can trip up text editors. For example, in Windows Notepad, you need to choose File → Save As, and then pick UTF-8 from the Encoding list (see Figure 5-15). For the Mac’s TextEdit, select Format → Make Plain Text, go to Preferences → Open and Save → Plain Text File Encoding → Saving Files, and then select Unicode (UTF-8) from the drop-down list. Every time you re-save your file thereafter, Notepad and TextEdit will encode it correctly.



**Figure 5-15:** UTF-8 is a slimmed down version of Unicode that saves space for normal characters. It’s the overwhelming standard of the Web. However, you need to explicitly tell Notepad to use UTF-8 encoding when you save a Web page that includes special characters like accented letters.