PART II

Core HTML and XHTML

CHAPTER 3
Core Elements

CHAPTER 4
Links and Addressing

3
CHAPTER

Core Elements

This chapter introduces the basic HTML tags common to nearly every browser, as defined by the HTML 4.01 and XHTML 1.0 transitional specifications. The tags fall primarily into three distinct groups: document structure elements, block elements, and inline elements. In addition to these elements are character entities for inserting special characters into a document. The elements are presented for the most part in a top-down manner: from larger, document and block-oriented structures (such as paragraphs), to smaller units (such as the actual character entities). To speed up the discussion, common attributes to nearly all HTML elements are discussed as a group. Before proceeding, we quickly revisit the structure of HTML documents.

Document Structure Redux

Recall from Chapter 1 that HTML and XHTML are structured languages. All documents should start with a document type definition indicating the type of markup in use and then have a root **html** element that contains a **<head>** and **<body>** tag, or in some situations a **<head>** and **<frameset>** tag. Within the head of the document, information about the document such as its title, character set, style sheets, and scripts is indicated, while the **<body>** tag encloses the actual content of the page. An example structure for an XHTML 1.0 transitional document is shown here:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Title here</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />

<!-- other head information here -->
</head>
<body>
<!-- other body information here -->
</body>
</html>
```

Given the basic XHTML template, the italicized areas would be modified appropriately per document. All the following examples will continue on with this XHTML 1.0 transitional template except where noted. However, if you prefer to write markup to another specification, all the examples presented in the chapter can be found at the support site (www.htmlref.com) in HTML 4.0 transitional, HTML 4.01 strict with CSS, and XHTML 1 strict with CSS. Now that we have reminded ourselves of the overall document structure, it is time to explore the tags within the body of the document. But before we do that, we should talk about the aspects of these tags that are always similar: the core attributes.

Core HTML Attributes

To accommodate new technologies such as style sheets and scripting languages, some important changes have been made to HTML and XHTML. Since HTML 4.0, a set of four core attributes—**id**, **class**, **style**, and **title**—have been added to nearly all HTML and XHTML elements. At this stage, the purpose of these attributes might not be obvious, but it is important to address their existence before discussing the various HTML elements to avoid redundancy.

id Attribute

The **id** attribute is used to set a unique name for an element in a document. For example, using **id** with the paragraph tag, **<p>**,

```
<p id="FirstParagraph">
This is the first paragraph of text.
</p>
```

names the bound element **"FirstParagraph"**. Naming an element is useful for manipulating the enclosed contents with a style sheet or script. For example, a style sheet rule such as

```
<style type="text/css">
  #FirstParagraph {color: red;}
</style>
```

could be put in the **head** of a document. This style rule says to make an element named "FirstParagraph" red. Naming is key to associating style or interactivity to particular elements. Of course, document authors must make sure objects are named uniquely, as having elements with the same **id** attribute value might cause significant bugs. The uses of the **id** attribute for style sheets and scripting are discussed in Chapter 10 and Chapter 14, respectively.

class Attribute

The **class** attribute is used to indicate the class or classes that a tag might belong to. Like **id**, **class** is used to associate a tag with a name, so

```
<p id="FirstParagraph" class="important">
   This is the first paragraph of text.
</p>
```

not only names the paragraph uniquely as **FirstParagraph**, but also indicates that this paragraph belongs to a class grouping called **important**. The main use of the **class** attribute is to relate a group of elements to various style sheet rules. For example, a style sheet rule such as

```
<style type="text/css">
 .important {background-color: yellow;}
</style>
```

would give all elements with the **class** attribute set to **important** a yellow background. Given that many elements can have the same class values, this may affect a large portion of the document. You can find more examples of the use of **class** and **id** with style sheets in Chapter 10.

style Attribute

The **style** attribute is used to add style sheet information directly to a tag. For example,

```
<p style="font-size: 18pt; color: red;">
   This is the first paragraph of text.
</p>
```

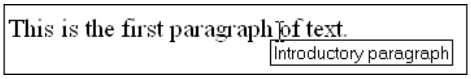
sets the font size of the paragraph to be 18 point, red text. Although the **style** attribute allows CSS rules to be added to an element with ease, it is preferable to use **id** or **class** to relate a document-wide or linked style sheet. The use of CSS is discussed in Chapter 10.

title Attribute

The **title** is used to provide advisory text about an element or its contents. In the case of

```
<p title="Introductory paragraph">
This is the first paragraph of text.
</p>
```

the **title** attribute is set to indicate that this particular paragraph is the introductory paragraph. Browsers can display this advisory text in the form of a *Tooltip*, as shown here:


This is the first paragraph of text.

Tooltips set with **title** values are often found on links, form fields, images, and anywhere where an extra bit of information is required.

The core attributes might not make a great deal of sense at this time because generally they are most useful with scripting and style sheets, but keep in mind that these four attributes are assumed with *every* tag that is introduced for the rest of this chapter.

Core Language Attributes

One major goal of HTML 4 was to provide better support for languages other than English. The use of other languages might require that text direction be changed from left to right across the screen to right to left. Nearly all HTML elements now support the **dir** attribute, which can be used to indicate text direction as either **ltr** (left to right) or **rtl** (right to left). For example,

```
<p dir="rtl">
   This is a right to left paragraph.
</p>
```

Furthermore, mixed-language documents might become more common after support for non-ASCII-based languages is improved within browsers. The use of the **lang** attribute enables document authors to indicate, down to the tag level, the language being used. For example,

```
<p lang="fr">
   C'est Francais.
</p>

<p lang="en">
   This is English.
</p>
```

Although the language attributes should be considered part of nearly every HTML element, in reality, these attributes are not widely supported by all browsers and are rarely used by document authors.

Core Events

The last major aspect of modern markup initially introduced by HTML 4 was the increased possibility of adding scripting to HTML documents. In preparation for a more dynamic Web, a set of core events has been associated with nearly every HTML element. Most of these events are associated with a user doing something. For example, the user clicking an object is associated with an **onclick** event attribute. So,

```
<p onclick="alert('Ouch!');">
Press this paragraph
</p>
```

would associate a small bit of scripting code with the paragraph event, which would be triggered when the user clicks the paragraph. In reality, the event model is not fully supported by all browsers for all tags, so the previous example might not do much of anything. A much more complete discussion of events is presented in Chapter 14, as well as in Appendix A. For now, just remember that any tag can have a multitude of events associated with it, paving the way for a much more dynamic Web experience.

Now that the core attributes have been covered, we can avoid mentioning them for every element presented, and turn to the most common elements used in HTML. The next section begins the discussion with some of the most common elements found in a document— headings.

Headings

The heading elements are used to create "headlines" in documents. Six different levels of headings are supported: **<h1>**, **<h2>**, **<h3>**, **<h4>**, **<h5>**, and **<h6>**. These range in importance from **<h1>**, the most important, to **<h6>**, the least important. Most browsers display headings in larger and/or bolder font than normal text. Many HTML authors erroneously think of heading elements as formatting that makes text bigger or bolder. Actually, heading elements convey logical meaning about a document's structure. However, in most visual browsers, sizing and weight are relative to the importance of the heading. Therefore, **<h1>** level headings are larger and generally bolder than **<h3>** headings, leading developers to interpret these logical tags most often in a physical manner. In addition, as a block element, returns are inserted after the heading unless overridden by a style sheet. The following example markup demonstrates the heading elements:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Heading Test</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>

<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>

</body>
</html>
```

A sample rendering of this heading example is shown in Figure 3-1.

NOTE *The Lynx text browser renders headings very differently from commercial graphical browsers. Lynx can't display larger fonts, so it might attempt to bold them or align them.* ***h1*** *headings are aligned in the center, and each lower-level heading is indented more than the next-highest level heading.*

Besides the core attributes discussed early in the chapter, the primary attribute used with headings is **align**. By default, headings usually are left-aligned, but the value of the **align** attribute of headings can also be set to **right**, **center**, and **justify** under transitional versions of HTML and XHTML. The following example markup shows the common usage of the **align** attribute for headings:

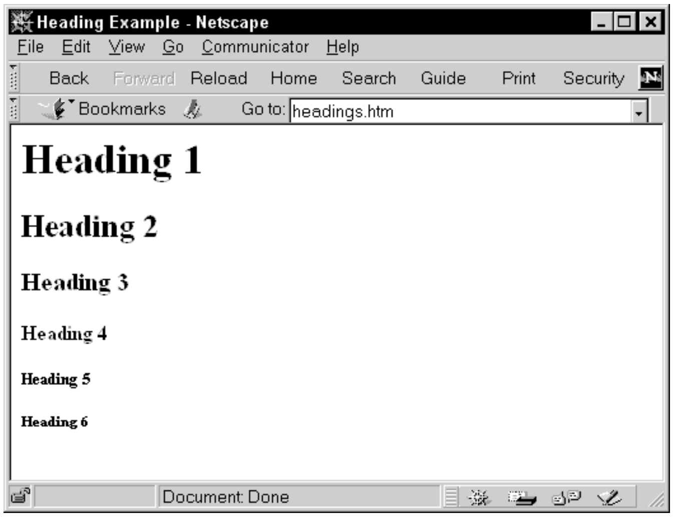
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Heading Alignment Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
```

```
<h1 align="left">Aligned Left</h1>
<h1 align="center">Aligned Center</h1>
<h1 align="right">Aligned Right</h1>

</body>
</html>
```

Under the strict version of HTML, as well as under XHTML, the **align** attribute has been deprecated in favor of using style sheets. The example that follows would validate under strict XHTML 1.0 as it uses CSS to align the headings. A full discussion of CSS begins in Chapter 10.

```
<!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" lang="en">
<head>
<title>XHTML Heading Alignment</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<style type="text/css">
     h1.left {text-align: left;}
     h1.center {text-align: center;}
     h1.right {text-align: right;}
</style>
</head>
<body>

<h1 class="left">Aligned Left</h1>
<h1 class="center">Aligned Center</h1>
<h1 class="right">Aligned Right</h1>

</body>
</html>
```

FIGURE 3-1 Rendering of heading style example


Heading Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security
Bookmarks Go to: headings.htm
Document: Done

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Regardless of appropriateness, given the rise of CSS, many HTML authors often use headings to make text large. As with all HTML elements viewed in a presentational manner, size is a relative concept, not an absolute concept. The actual size of the heading depends on the browser, the browser's settings, and the platform on which it is running. The size of an **<h1>** header under Netscape on a UNIX system may be different from the same **<h1>** header on a Windows machine running Internet Explorer. The headlines are relatively bigger, but the exact size is unknown, making consistent layout difficult.

NOTE *A quick survey of heading use on the Web should reveal that headings beyond* ***<h3>*** *rarely are used. Why? Certainly this is because people use headings in a visual fashion and the look of* ***<h4>****,* ***<h5>*** *and* ***<h6>*** *tags can be easily accomplished with other tags.*

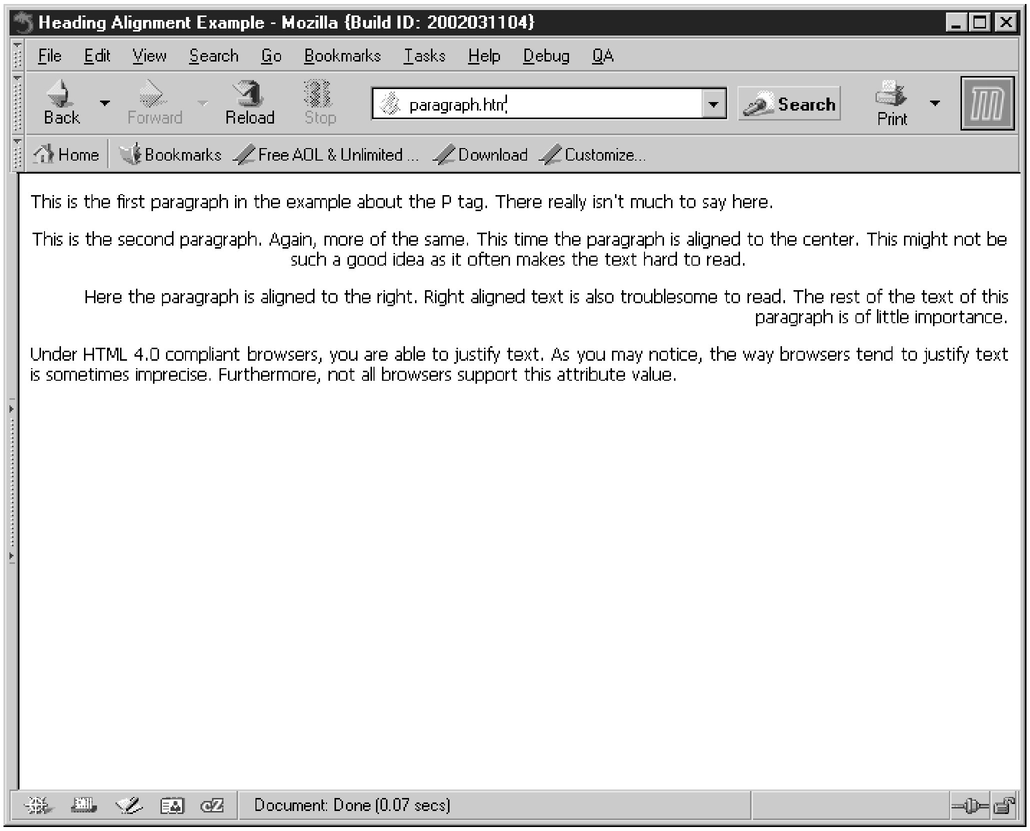
Paragraphs and Breaks

Recall that white space handling rules of HTML suggest that multiple spaces, tabs, and carriage returns are often ignored. Word wrapping can occur at any point in your source file, and multiple white space characters are collapsed into a single space. Instead of relying solely on white space, structural elements are used to section a document. One of the most important structuring elements is the paragraph element. Surrounding text with the **<p>** and **</p>** tags indicates that the text is a logical paragraph unit. In general, the browser places a blank line or two before the paragraph, but the exact rendering of the text depends on the browser and any applied style sheet. Text within the **<p>** tags generally is rendered flush left, with a ragged right margin. Like headings, the **align** attribute makes it possible to specify a **left**, **right**, or **center** alignment. Since HTML 4.0, you also can set an **align** value of **justify**, to justify all text in the paragraph. Due to the poor quality of justification in some browsers, this value is used only rarely. The following example in XHTML 1.0 transitional shows four paragraphs with alignment, the rendering of which is shown in Figure 3-2:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html> xmlns="http://www.w3.org/1999/shtml" lang="en">
<head>
<title>Heading Alignment Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>

<p>This is the first paragraph in the example about the P tag.
There really isn't much to say here.</p>

<p align="center">This is the second paragraph. Again, more of the
same. This time the paragraph is aligned to the center. This might
not be such a good idea as it often makes the text hard to read.</p>

<p align="right">Here the paragraph is aligned to the right. Right
aligned text is also troublesome to read. The rest of the text of this
paragraph is of little importance.</p>
```

```
<p align="justify">Under HTML 4.0 compliant browsers, you are
able to justify text. As you may notice, the way browsers tend to
justify text is sometimes imprecise. Furthermore, some older
browsers do not support this attribute value.</p>

</body>
</html>
```

Because under its default rendering the paragraph element causes a blank line, some HTML authors attempt to insert blank lines into a document by using multiple **<p>** tags. This rarely results in the desired outcome. The browser will collapse empty **<p>** tags because they have no contents and thus desired formatting may not be achieved. To get around this problem, many WYSIWYG HTML editors and even some by-hand HTML authors use a nonbreaking space character within a paragraph, to keep the element from collapsing, as shown here: **<p>&nbsp;</p>**. This approach isn't recommended because it doesn't reduce markup used and further obscures the meaning of the document. A break should be used instead.


Heading Alignment Example - Mozilla {Build ID: 2002031104}
File Edit View Search Go Bookmarks Tasks Help Debug QA
Back Forward Reload Stop
paragraph.htm
Search
Print
Home Bookmarks Free AOL & Unlimited ... Download Customize...
Document: Done (0.07 secs)

This is the first paragraph in the example about the P tag. There really isn't much to say here.

This is the second paragraph. Again, more of the same. This time the paragraph is aligned to the center. This might not be such a good idea as it often makes the text hard to read.

Here the paragraph is aligned to the right. Right aligned text is also troublesome to read. The rest of the text of this paragraph is of little importance.

Under HTML 4.0 compliant browsers, you are able to justify text. As you may notice, the way browsers tend to justify text is sometimes imprecise. Furthermore, not all browsers support this attribute value.

FIGURE 3-2 Rendering of the paragraph example

To insert returns or blank lines in an HTML or XHTML document, the **<br />** tag is used. This tag inserts a single carriage return or line break into a document. It is an empty element—thus, it has no close tag. Because of this, under XHTML you would use **<br />** instead of just plain **<br>**. In addition to the core attributes, the one attribute commonly used with a **<br />** tag is **clear**. This attribute controls how text flows around images or embedded objects. The use of **<br />** in this fashion is discussed in Chapter 5.

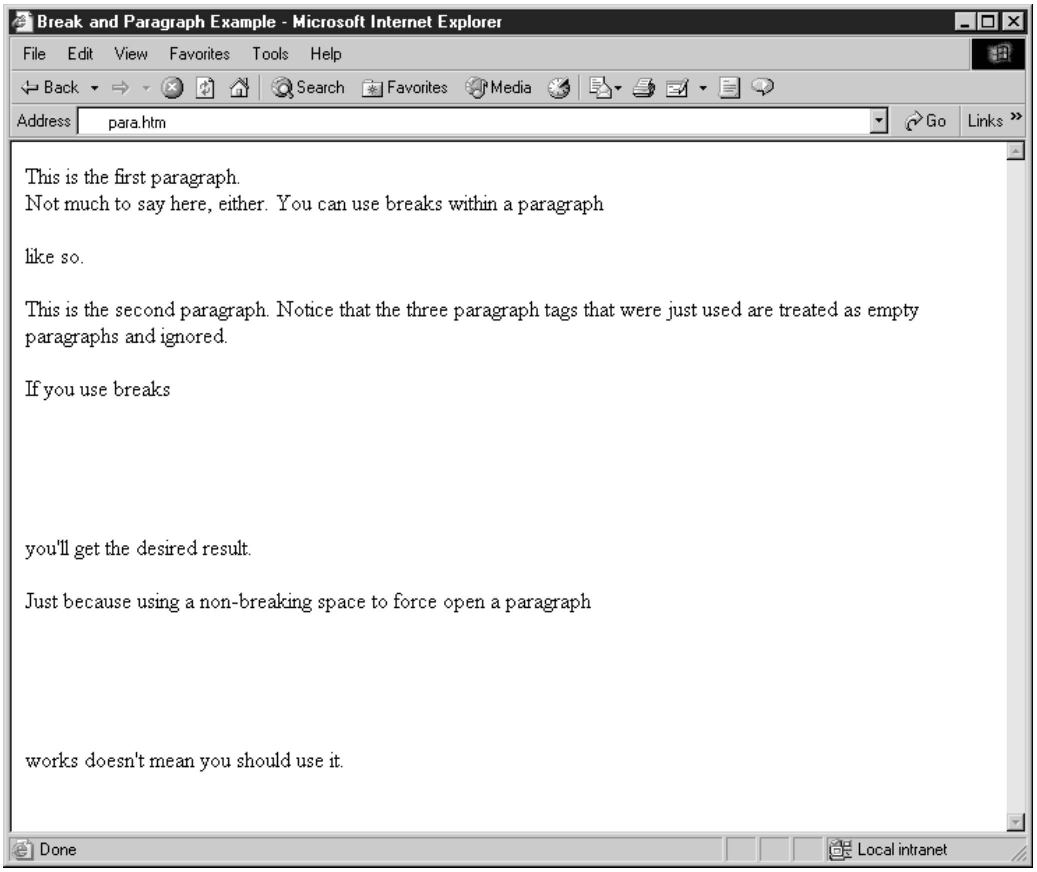
The following code fragment shows the basic uses of **p** and **br**, and also shows that the two elements are really not equivalent, despite their physical rendering similarities (a screen rendering appears in Figure 3-3):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Break and Paragraph Example</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>

<p>This is the first paragraph.<br />
Not much to say here, either. You can use
breaks within a paragraph<br /><br />
like so.</p>

<p></p><p></p><p></p>

<p>This is the second paragraph. Notice that the three paragraph
tags that were just used are treated as empty paragraphs and ignored.</p>

<p>If you use breaks</p>

<br /><br /><br /><br />
<p>you'll get the desired result.</p>
<p>Just because using a non-breaking space to force open a paragraph</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>works doesn't mean you should use it.</p>
</body>
</html>
```

TIP *Users looking for blank lines have to insert multiple **<br>** tags into their documents. A single **<br>** tag merely goes to the next line rather than inserting a blank line.*

It should be noted that under strict DTDs, the **br** element is not allowed directly in the body as in the previous example and must be present within a heading (**<h1>** - **<h6>**), **<p>**, **<pre>**, **<div>**, or **<address>** tag. Moving the **<br />** tags within the nearest **<p>** tag like so

```
<p>If you use breaks
<br /><br /><br /><br /></p>
```


Break and Paragraph Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address para.htm
Go Links
Done
Local intranet

This is the first paragraph.
Not much to say here, either. You can use breaks within a paragraph

like so.

This is the second paragraph. Notice that the three paragraph tags that were just used are treated as empty paragraphs and ignored.

If you use breaks

you'll get the desired result.

Just because using a non-breaking space to force open a paragraph

works doesn't mean you should use it.

FIGURE 3-3 Rendering of break and paragraph example

would allow the document to validate under a strict doctype. Given that many HTML authors still do not use strict HTML or XHTML, the tag continues to be commonly found outside block tags.

Divisions and Centering

The **<div>** tag is used to structure HTML documents into sections or divisions. A **<div>** is a logical block tag that has no predefined meaning or rendering. Under traditional HTML, the only major value of the **<div>** tag is to align sections of content by setting the **align** attribute to **left**, **right**, or **center**. By default, content within the **<div>** tag is left-aligned. Divisions are more significantly useful when used in conjunction with style sheets (see Chapter 10).

Aside from using the **<div>** tag to align blocks of text, it is possible to center text using a difficult-to-characterize proprietary tag: **<center>**. Under HTML 2.0-based browsers, centering text was impossible. One of the major additions introduced by Netscape was the **<center>** tag. HTML 3.2 adopted this element because of its widespread use, which continues today. To center text or embedded objects (such as images), simply enclose the content within **<center>** and **</center>**. In this sense, **<center>** appears to be a text-formatting style element, but under the HTML 3.2 and transitional 4.0 specification (and beyond), **<center>** is defined as an alias for a block-level structuring element and eventually will be deprecated under strict versions of HTML. Under the HTML 4.01 DTD, **<center>** is supposed to be simply an alias for **<div align="center">** and is treated exactly the same way. Specification or not, the **<center>** tag is unlikely to go away, considering its simplicity and widespread use. The following example shows the use of **<center>** and **<div>.** (Figure 3-4 shows their screen rendering.)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Center and Division Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<center>
<h1>This heading is centered.</h1>
<p>This paragraph is also centered.</p>
</center>

<br /><br />

<div align="right">
<h1>Division Heading</h1>
<p>Many paragraphs and other block elements
   can be affected by a DIV at once.</p>
<p>Notice all the paragraphs are right aligned.</p>
</div>
</body>
</html>
```

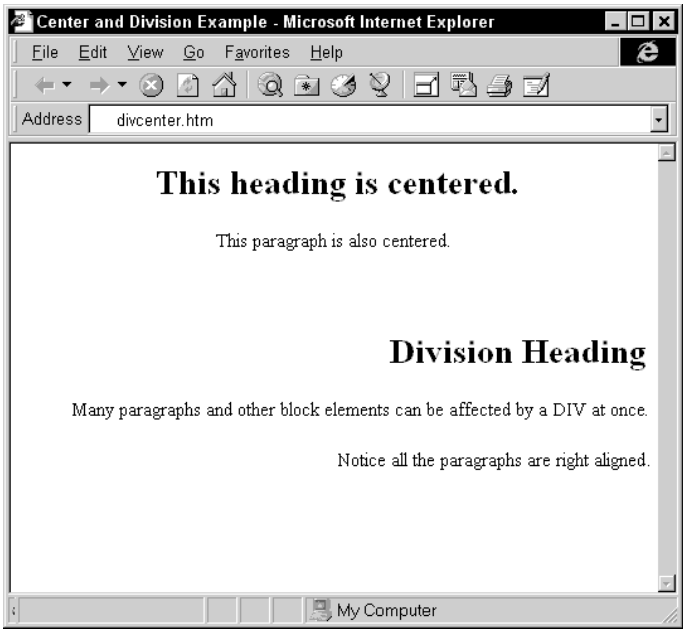
Spans

Although the **<div>** tag can be used to group large sections of a document for later application of a style sheet or various other formatting, it is not appropriate to put everything within a division. Because **div** is a block element, it will induce a return. If you want to group text without using a block element, use a **<span>** tag, as it provides logical grouping inline with no predefined look. Consider the following markup:

```
<p>In this sentence <span class="important">some of the text is
important!</span></p>
```

In this markup fragment, the **<span>** tag wouldn't necessarily cause any particular presentation under plain HTML. However, as shown, using the **class** attribute, it could be related to a style sheet to make the enclosed text look different, while at the same time providing no particular logical meaning. We could do something directly, like so:

```
<p>In this sentence <span style="color: red; font-size: xx-large;">some of the text
is big and red</span>!</p>
```


Center and Division Example - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address divcenter.htm
My Computer

This heading is centered.

This paragraph is also centered.

Division Heading

Many paragraphs and other block elements can be affected by a DIV at once.

Notice all the paragraphs are right aligned.

FIGURE 3-4 Example rendering of <div> and <center>

but that would defeat the value of separating presentation from document structure with styles. It would seem at this point the use of **<div>** and **<span>** tags might not make a great deal of sense, but they are some of the most useful of the core elements of XHTML. Their use with style sheets is discussed in Chapter 10.

Quotations

Occasionally, you might want to quote a large body of text to make it stand out from the other text. The **<blockquote>** tag provides a facility to enclose large block quotations from other works within a document. Although the element is logical in nature, enclosing text within **<blockquote>** and **</blockquote>** usually indents the blocked information from both the left and right. In keeping with its meaning, the **<blockquote>** tag supports the cite attribute, which can be set to the Web address of the document or site from which the quotation was pulled, or a brief message describing the quote or its source.

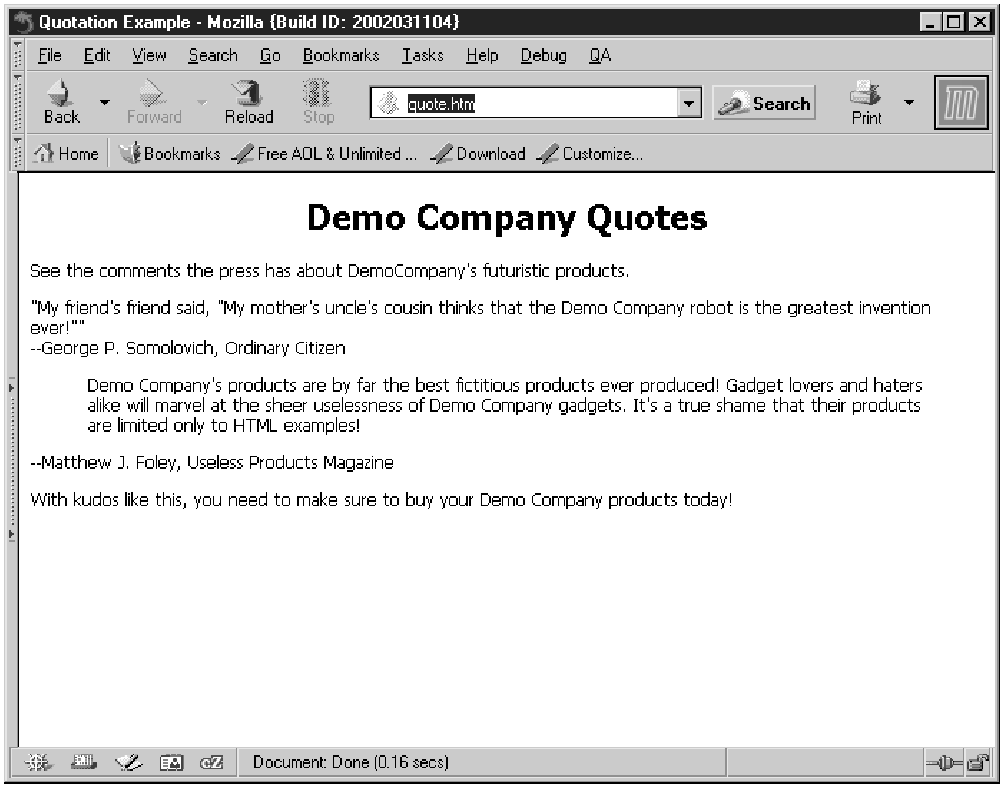
Whereas a **<blockquote>** element will cause a return like other block elements, it is possible to create an inline quotation using the logical **<q>** element. The tag should result in quotation marks around the enclosed text. The tag also should address the rules for switching quotes within quotes. Note that older browsers and even some modern ones such as Internet Explorer 6 do not properly support the **<q>** tag, despite its inclusion in the HTML 4.0/4.01 and XHTML 1.0 specifications. Like the **<blockquote>** tag, **<q>** also supports a **cite** attribute. The following shows an example of **<blockquote>** and **<q>** (rendered in Figure 3-5):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Quotation Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h1 align="center">Demo Company Quotes</h1>

<p>See the comments the press has about Demo Company's
futuristic products.</p>

<q>My friend's friend said, <q cite="sounds fishy">My mother's
uncle's cousin thinks that the Demo Company robot is the
greatest invention ever!</q></q>
<br />--George P. Somolovich, Ordinary Citizen

<blockquote cite="http://www.democompany.com">
Demo Company's products are by far the best fictitious products
ever produced! Gadget lovers and haters alike will marvel at the
sheer uselessness of Demo Company gadgets. It's a true shame that
their products are limited only to HTML examples!
</blockquote>
--Matthew J. Foley, Useless Products Magazine

<p>With kudos like this, you need to make sure to buy your
Demo Company products today!</p>
</body>
</html>
```

NOTE *The first Web browsers did not provide any indentation or layout control in regular text. Many HTML authors and HTML editors use* **<blockquote>** *to provide indentation. List elements, particularly the unordered list, are also commonly used for quick indentation in Web pages. While it is poor practice, until style sheets become more common, these workarounds will continue.*

Preformatted Text

Occasionally, spacing, tabs, and returns are so important in text that HTML's default behavior of disregarding them would ruin the text's meaning. In such cases, you might want to preserve the intended formatting by specifying the text to be preformatted. Imagine that programming source code or poetry needs to be inserted into a Web page. In both cases, the spacing, returns, and tabs in the document must be preserved to ensure proper meaning. This situation requires an HTML directive that indicates the preservation of format. The **<pre>** tag can be used to indicate text that shouldn't be formatted by the browser. The text enclosed within a **<pre>** tag retains all spacing and returns, and doesn't reflow when the browser is resized. Scrollbars and horizontal scrolling are required if the lines are longer than the width of the window. The browser generally renders the preformatted text in a monospaced font, typically Courier. Simple forms of text formatting, such as bold, italics,


Quotation Example - Mozilla {Build ID: 2002031104}
File Edit View Search Go Bookmarks Tasks Help Debug QA
Back Forward Reload Stop
quote.htm
Search
Print
Home Bookmarks Free AOL & Unlimited ... Download Customize...
Document: Done (0.16 secs)

Demo Company Quotes

See the comments the press has about DemoCompany's futuristic products.

"My friend's friend said, "My mother's uncle's cousin thinks that the Demo Company robot is the greatest invention ever!""
--George P. Somolovich, Ordinary Citizen

Demo Company's products are by far the best fictitious products ever produced! Gadget lovers and haters alike will marvel at the sheer uselessness of Demo Company gadgets. It's a true shame that their products are limited only to HTML examples!

--Matthew J. Foley, Useless Products Magazine

With kudos like this, you need to make sure to buy your Demo Company products today!

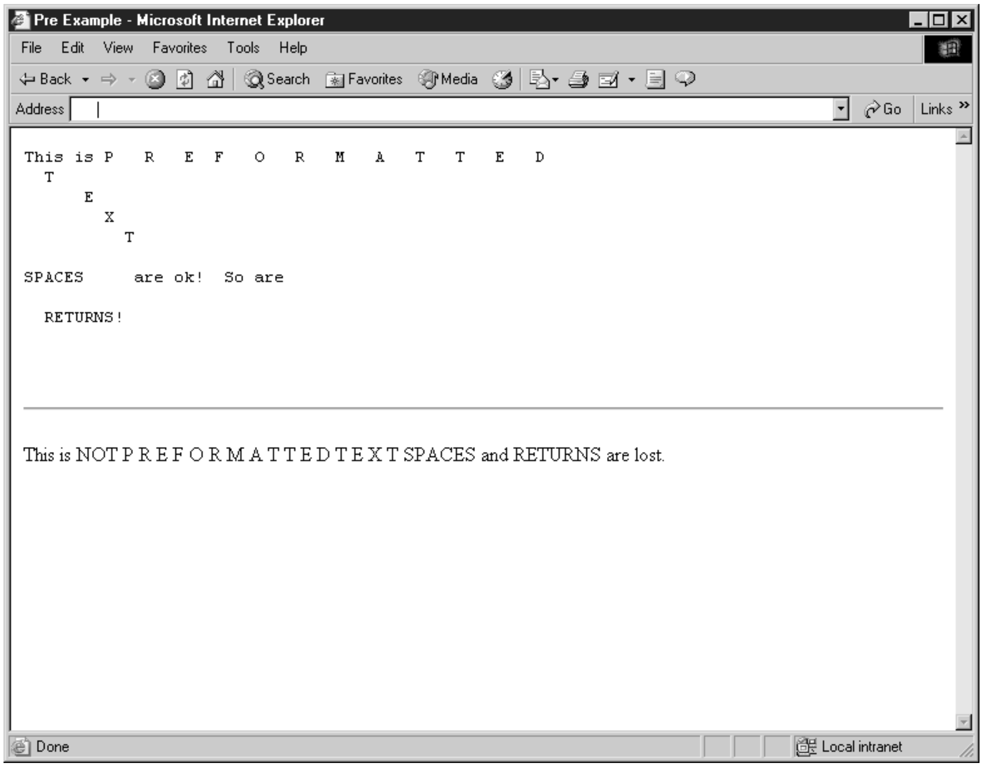
FIGURE 3-5 Rendering of quotations example

or links, can be used within **<pre>** tags. The following example, displayed in Figure 3-6, uses the **<pre>** tag and compares it to regular paragraph text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Pre Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<pre>
This is P   R   E  F   O   R   M   A   T   T   E   D
  T
      E
        X
          T

SPACES     are ok!  So are

  RETURNS!
```

```
</pre>
<hr />
<p>
This is NOT P    R    E  F    O    R    M    A    T    T    E    D
    T
      E
        X
          T

SPACES     and
RETURNS are lost.
</p>
</body>
</html>
```

NOTE *According to the HTML and XHTML specifications, only certain HTML elements are allowed within the* ***<pre>*** *tag, and some elements, such as* ***<img>****, are excluded. Most browsers allow any elements, even those beyond the stated specification, to appear within* ***<pre>****, and render these as expected. Authors should not, however, rely on this. See Appendix A for the* ***<pre>*** *tag's content model.*


Pre Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Done
Local intranet

```
This is P    R    E  F    O    R    M    A    T    T    E    D
  T
      E
         X
            T

SPACES     are ok!  So are

  RETURNS!
```

This is NOT P R E F O R M A T T E D T E X T SPACES and RETURNS are lost.

FIGURE 3-6 Rendering of preformatted and regular text

HTML authors should be careful about using **<pre>** to create simple tables or preserve spacing. Unpredictable differences in browser window sizes could introduce horizontal scrolling for wide preformatted content. In these cases, other elements, particularly **<table>,** will provide better formatting control.

Lists

Modern HTML has three basic forms of lists: ordered lists (**<ol>**), unordered lists (**<ul>**), and definition lists (**<dl>**). Lists are block-level, although they can be nested, and their items can contain other block-level structures, such as paragraphs.

Ordered Lists

An ordered list, as enclosed by **<ol>** and **</ol>**, defines a list in which order matters. Ordering typically is rendered by a numbering scheme, using Arabic numbers, letters, or Roman numerals. Ordered lists are suitable for creating simple outlines or step-by-step instructions because the list items are automatically numbered by the browser. List items in ordered and other lists are defined by using the list item tag, **<li>**, which doesn't require an end tag under traditional HTML. For XHTML compliance, however, the use of the closing **</li>** tag is required. List items usually are indented by the browser. Numbering starts from one. A generic ordered list looks like this:

```
<ol>
   <li>Item 1</li>
   <li>Item 2</li>
     . . .
   <li>Item n</li>
</ol>
```

NOTE *In many browsers, the* ***<li>*** *tag has a rendering outside a list. It often renders as a nonindented bullet. Some books recommend using* ***<li>*** *in this way. This isn't correct practice given the HTML/XHTML content model. It should be avoided.*

The **<ol>** tag has three basic attributes, none of which is required: **compact**, **start**, and **type**. The **compact** attribute requires no value under traditional HTML but under XHTML, which allows no attributes without values, it is set to a value of compact, like so:

```
<ol compact="compact">
```

It simply suggests that the browser attempt to compact the list, to use less space onscreen. In reality, most browsers ignore the **compact** attribute.

The **type** attribute of **<ol>** can be set to **a** for lowercase letters, **A** for uppercase letters, **i** for lowercase Roman numerals, **I** for uppercase Roman numerals, or **1** for regular numerals. The numeral **1** is the default value. Remember that the **type** attribute within the **<ol>** tag sets the numbering scheme for the whole list, unless it is overridden by a **type** value in an **<li>** tag. Each **<li>** tag can have a local **type** attribute set to **a**, **A**, **i**, **I**, or **1**. Once an **<li>** element is set with a new type, it overrides the numbering style for the rest of the list, unless another **<li>** sets the **type** attribute.

The **<ol>** element also has a **start** attribute that takes a numeric value to begin the list numbering. Whether the **type** attribute is a letter or a numeral, the **start** value must be a number. To start ordering from the letter *j*, you would use **<ol type="a" start="10">** because *j* is the tenth letter. An **<li>** tag within an ordered list can override the current numbering with the **value** attribute, which also is set to a numeric value. Numbering of the list should continue from the value set.

NOTE *Numbering lists to count backward from 10 to 1 or to count by twos or other values is not directly possible in HTML. I believe that a single addition of a step attribute could address this, but it appears that the few remaining holes in HTML have been left unfilled. Instead, CSS is left to address this, which unfortunately is taking far too long.*

Lists can be nested, but the syntax is widely misunderstood. Commonly, on the Web, markup such as

```
<ol>
   <li>Item 1</li>
    <ol>
      <li>Item a</li>
        . . .
      <li>Item z</li>
    </ol>
     ...
   <li>Item n</li>
</ol>
```

is used. However, the content model of HTML and XHTML lists only allows list items inside of lists, so the correct syntax is actually

```
<ol>
   <li>Item 1
     <ol>
       <li>Item a</li>
. . .
       <li>Item z</li>
       </ol>
   </li>
...
   <li>Item n</li>
</ol>
```

A complete example of ordered lists and their attributes is shown next, the rendering of which is shown in Figure 3-7:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Ordered List Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
```

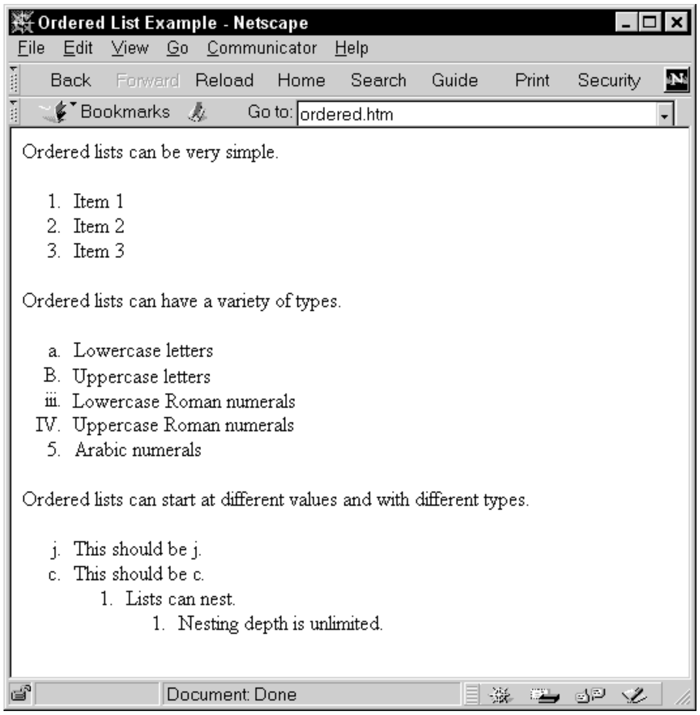
```
</head>
<body>
<p>Ordered lists can be very simple.</p>

<ol>
     <li>Item 1</li>
     <li>Item 2</li>
     <li>Item 3</li>
</ol>

<p>Ordered lists can have a variety of types.</p>

<ol>
     <li type="a">Lowercase letters</li>
     <li type="A">Uppercase letters</li>
     <li type="i">Lowercase Roman numerals</li>
     <li type="I">Uppercase Roman numerals</li>
     <li type="1">Arabic numerals</li>
</ol>

<p>Ordered lists can start at different values
and with different types.</p>

<ol start="10" type="a">
<li>This should be j</li>
<li value="3">This should be c
   <ol>
      <li>Lists can nest
          <ol>
             <li>Nesting depth is unlimited</li>
          </ol>
      </li>
   </ol>
</li>
</ol>
</body>
</html>
```

NOTE *When dealing with extremes, use numbering with caution. Negative values or very large values produce unpredictable results. Whereas Navigator ignores negative numbers, Internet Explorer numbers up toward zero. Browsers can allocate a fixed width to the left of a list item to display its number. Under Navigator, a list not embedded in another block structure can accommodate only about four digits; larger numbers can overwrite list elements. A list indented by nesting in another block structure could have more space. Numbering in both Navigator and Internet Explorer loses meaning with large integer values, most likely because of limitations within the operating environment.*

Unordered Lists

An unordered list, signified by **<ul>** and **</ul>**, is used for lists of items in which the ordering is not specific. This can be useful in a list of features and benefits for a product.

FIGURE 3-7 Rendering of ordered list example


Ordered List Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security
Bookmarks Go to: ordered.htm
Document: Done

Ordered lists can be very simple.

1. Item 1
2. Item 2
3. Item 3

Ordered lists can have a variety of types.

a. Lowercase letters
B. Uppercase letters
iii. Lowercase Roman numerals
IV. Uppercase Roman numerals
5. Arabic numerals

Ordered lists can start at different values and with different types.

j. This should be j.
c. This should be c.
    1. Lists can nest.
        1. Nesting depth is unlimited.

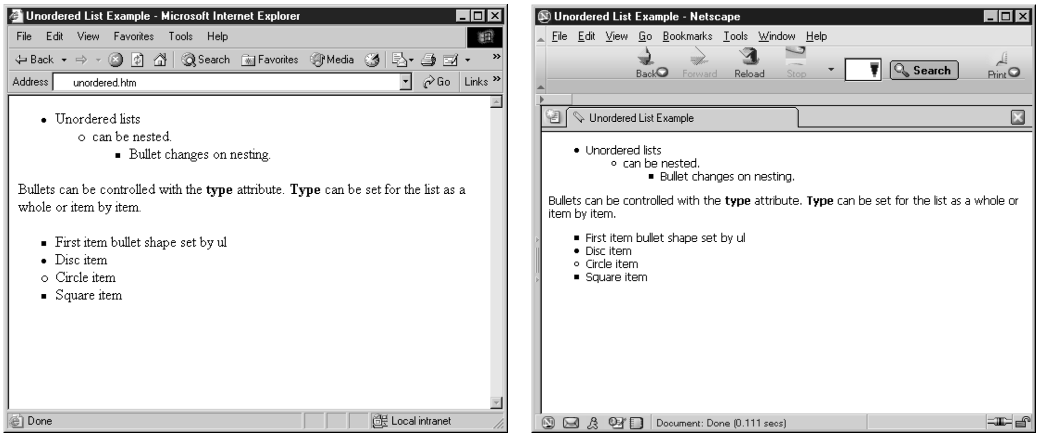
A browser typically adds a bullet of some sort (a filled circle, a square, or an empty circle) for each item and indents the list.

Like ordered lists, unordered lists can be nested. However, in this case each level of nesting indents the list farther, and the bullet changes accordingly. Generally, a filled circle or solid round bullet is used on the first level of lists. An empty circle is used for the second-level list. Third-level nested lists generally use a square. These renderings for bullets are common to browsers, but shouldn't be counted on. Under transitional forms of HTML and XHTML, the **type** attribute can be used to set the bullet type for a list. Under strict variants, CSS should be used instead. The **type** attribute can appear within the **<ul>** tag and set the type for the whole list, or it can appear within each **<li>**. A **type** value in an **<li>** tag overrides the value for the rest of the list, unless it is overridden by another **type** specification. The allowed values for **type** are **disc**, **circle**, or **square**. This is commonly supported in browsers, but small variations exist. For example, in the case of MSN TV (formerly WebTV), a triangle bullet type also is available, because on a television a circle and square generally look the same due to limited resolution. For the greatest level of cross-browser compatibility, authors are encouraged to set the bullet type only for the list as a whole.

NOTE *Internet Explorer 3.0–level browsers under Windows don't render **type** settings for unordered lists. This has been fixed under Internet Explorer 4.0 and beyond.*

The following is an example of unordered lists, various renderings of which are shown in Figure 3-8:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Unordered List Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<ul>
    <li>Unordered lists
        <ul>
            <li>can be nested.
                <ul>
                    <li>Bullet changes on nesting.</li>
                </ul>
            </li>
        </ul>
     </li>
</ul>

<p>Bullets can be controlled with the <b>type</b> attribute.
<b>Type</b> can be set for the list as a whole or item by item.</p>

<ul type="square">
    <li>First item bullet shape set by ul</li>
    <li type="disc">Disc item</li>
    <li type="circle">Circle item</li>
    <li type="square">Square item</li>
</ul>

</body>
</html>
```


Unordered List Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address unordered.htm
Go Links
Done
Local intranet
Unordered List Example - Netscape
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop Search Print
Unordered List Example
Document: Done (0.111 secs)

- Unordered lists
  - can be nested.
    - Bullet changes on nesting.

Bullets can be controlled with the **type** attribute. **Type** can be set for the list as a whole or item by item.

- First item bullet shape set by ul
- Disc item
- Circle item
- Square item

FIGURE 3-8 Rendering of unordered list example

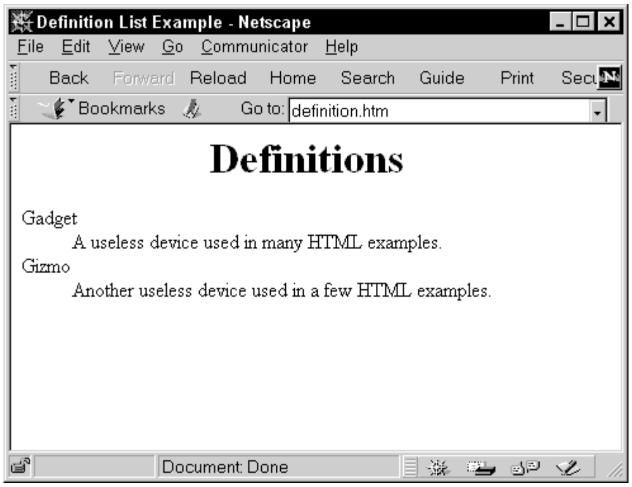
Definition List

A definition list is a list of terms paired with associated definitions—in other words, a glossary. Definition lists are enclosed within **<dl>** and **</dl>**. Each term being defined is indicated by a **<dt>** element, which is derived from "definition term." Each definition itself is defined by **<dd>**. Neither the **dt** nor the **dd** elements require a close tag under traditional HTML, yet given potentially long definitions and conformance to XHTML, the close tags should never be omitted. It is interesting to note that **<dt>** and **<dd>** tags are not required to be used in pairs, although they usually are. The following is a basic example using **<dl>**, the rendering of which is shown in Figure 3-9:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Definition List Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h1 align="center">Definitions</h1>
<dl>
   <dt>Gadget</dt>
   <dd>A useless device used in many HTML examples.</dd>

   <dt>Gizmo</dt>
   <dd>Another useless device used in a few HTML examples.</dd>
</dl>
</body>
</html>
```

Vestigial Lists: <dir> and <menu>

Beyond basic ordered, unordered, and definition lists, two other lists are specified under traditional HTML: **<menu>** and **<dir>**. They were supposed to specify short menus of



Definitions

Gadget
    A useless device used in many HTML examples.
Gizmo
    Another useless device used in a few HTML examples.

Definition List Example - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Secu
Bookmarks Go to: definition.htm
Document: Done

FIGURE 3-9 Rendering of definition list example

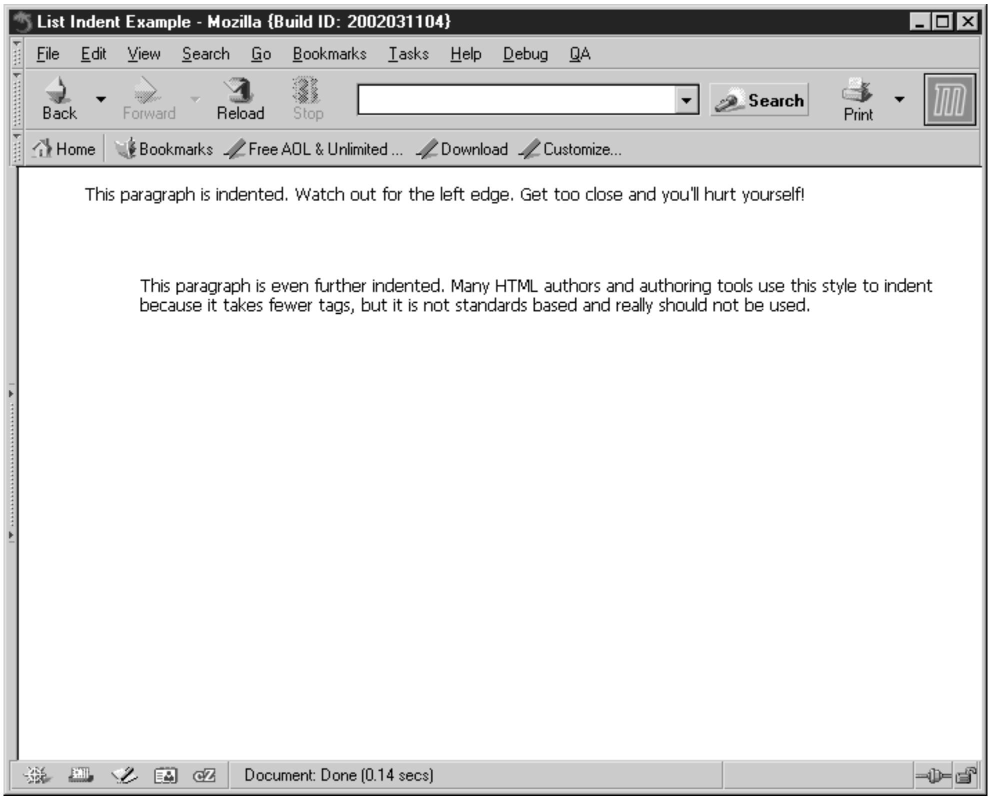
information and directory listings respectively, yet these rarely used elements generally appear as unordered lists in most browsers. Presented here solely for completeness, Web developers are warned to avoid using **<menu>** or **<dir>** because they have been dropped from the strict versions of HTML and XHTML.

Using Lists for Presentation

Because definition lists don't add numbering or bullets, many HTML writers have used this element to indent text. Although functionally this is the most appropriate way other than **<blockquote>** to achieve some rudimentary indentation without CSS, the unordered list often is used instead. The use of **<ul>** instead of **<dl>** to indent text quickly is unfortunately very common despite the fact that it is not valid markup. The most likely reason for this preference for **<ul>** is that it requires fewer elements to achieve indentation. A simple example of indenting with lists is shown next, with its rendering shown in Figure 3-10:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>List Indent Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<dl>
  <dd><p>This paragraph is indented. Watch out for
         the left edge. Get too close and you'll hurt yourself!
  </p></dd>
</dl>

<br /><br />

<!-- Warning: markup below is not valid XHTML  -->
<ul><ul>
  <p>This paragraph is even further indented. Many HTML authors
     and authoring tools use this style to indent because
     it takes fewer tags, but it is not standards based and
     really should not be used.</p>
</ul></ul>
<!-- End invalid markup -->

</body>
</html>
```

NOTE *Most HTML purists are offended by the use of* **<ul>** *to indent. If you really must use HTML for presentation, consider using the definition list or tables, if possible, to indent text. However, with WYSIWYG editors spitting out* **ul** *elements in mass numbers, this might be more of a fine point than a real issue. The rise of style sheets and other technologies should, finally in time, put an end to this question.*


List Indent Example - Mozilla {Build ID: 2002031104}
File Edit View Search Go Bookmarks Tasks Help Debug QA
Back Forward Reload Stop Search Print
Home Bookmarks Free AOL & Unlimited ... Download Customize...
Document: Done (0.14 secs)

This paragraph is indented. Watch out for the left edge. Get too close and you'll hurt yourself!

This paragraph is even further indented. Many HTML authors and authoring tools use this style to indent because it takes fewer tags, but it is not standards based and really should not be used.

FIGURE 3-10 Example of lists for presentation

Horizontal Rules

As sections are added to an HTML document, it is often useful to break up the document into visually distinct regions. A horizontal rule, indicated by the <**hr**> tag, is an empty block-level element that serves this purpose. As an empty element under XHTML, horizontal rules are written as **<hr />**.

Historically, under HTML 2.0, horizontal rules generally were rendered as an etched bar or line across a browser window. With HTML 3.2 and beyond, more control over the horizontal rule's look and size was added. For example, the **size** attribute sets the bar's thickness (height) in pixels. The **width** attribute sets the bar's width in pixels or percentage. The **align** attribute sets its alignment to left, right, or center. The **noshade** attribute renders the bar without a surrounding shadow. Additional browser-specific attributes (such as **color**) are described in the element reference in Appendix A. Under strict HTML 4.0 and XHTML, the various presentation attributes for the horizontal rule have been removed in favor of CSS, leaving the exact look of the line to the browser rendering the page. An example of horizontal rules and their basic attributes is shown next. A browser rendering is shown in Figure 3-11.

Note *Although it looks like a physical element,* ***hr*** *can have some logical meaning as a section break. For example, under an alternative browser, such as a speech-based browser, a horizontal rule theoretically could be interpreted as a pause. A handheld browser with limited resolution might use it as a device to limit scrolling of the text.*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Horizontal Rule Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>


<p>HR size of 10</p>
<hr size="10" />

<p>HR with width of 50% and no shading</p>
<hr width="50%" noshade="noshade" />

<p>HR with width of 200 pixels, size of 3 pixels, and no shading</p>
<hr width="200" size="3" noshade="noshade" />

<p>HR with width of 100 pixels, aligned right</p>
<hr align="right" width="100" />

<p>HR with width of 100 pixels, aligned left</p>
<hr align="left" width="100" />

<p>HR with width of 100 pixels, aligned center</p>
<hr align="center" width="100" />

</body>
</html>
```

Other Block-Level Elements

HTML has many other large tag structures, most notably tables and forms. Many other elements are available under Navigator and Internet Explorer, including frames, layers, and a variety of other formatting and structuring features. Because of their complexity, it makes sense to discuss these tags in later chapters. Tables are discussed in depth in Chapter 7, and forms are discussed in Chapter 12. Before moving on to inline elements, let's cover one element that's somewhat difficult to characterize: **<address>.**

address

The **<address>** tag is used to surround information, such as the signature of the person who created the page, or the address of the organization the page is about. For example,

```
<address>
Demo Company, Inc.<br />
```

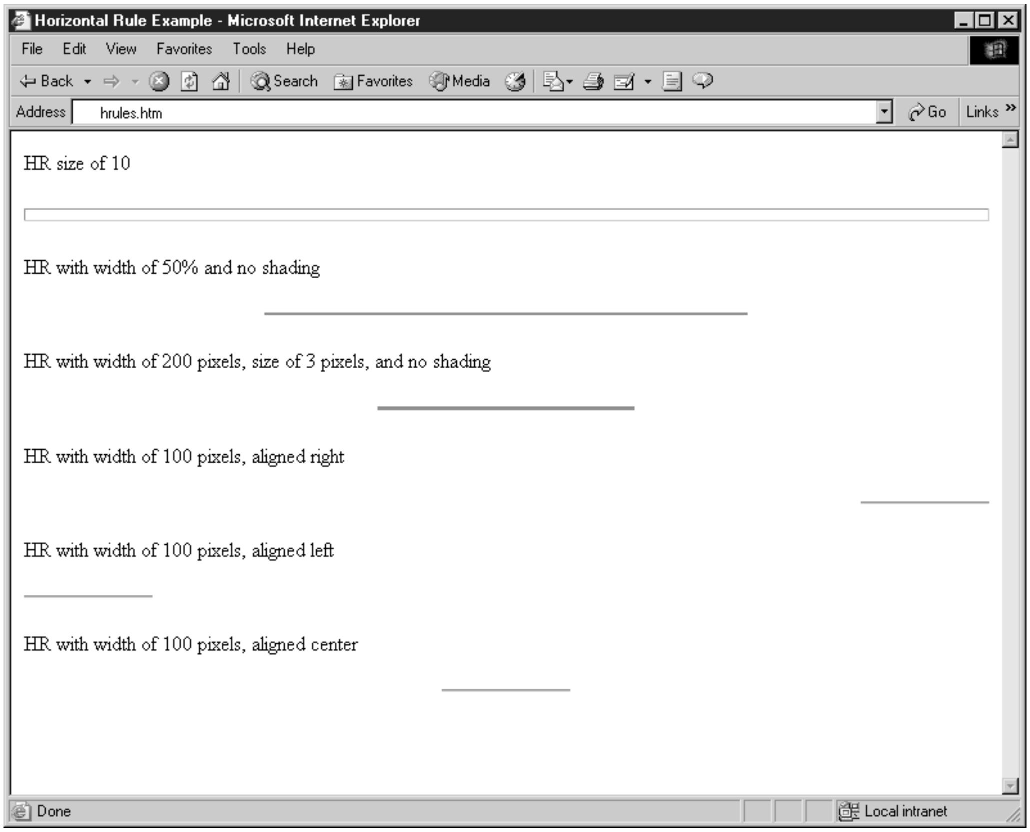
```
1122 Fake Street<br />
San Diego, CA 92109<br />
858.555.2086<br />
info@democompany.com<br />
</address>
```

can be inserted toward the bottom of every page throughout a Web site.

The **<address>** tag should be considered logical, although its physical rendering is italicized text. The HTML specification treats **<address>** as an idiosyncratic block-level element. Like other block-level elements, it inserts a blank before and after the block. It can enclose many lines of text, formatting elements to change the font characteristics and even images. However, according to the specification, it isn't supposed to enclose other block-level elements, although browsers generally allow this.

Text-Level Elements

Text-level elements in HTML come in two basic flavors: physical and logical. *Physical elements*, such as **<b>** for bold and **<i>** for italic, are used to specify how text should be


Horizontal Rule Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address hrules.htm
Go Links
Done
Local intranet

HR size of 10

HR with width of 50% and no shading

HR with width of 200 pixels, size of 3 pixels, and no shading

HR with width of 100 pixels, aligned right

HR with width of 100 pixels, aligned left

HR with width of 100 pixels, aligned center

FIGURE 3-11 Rendering of horizontal rule example

rendered. *Logical elements*, such as **<strong>** and **<em>**, indicate what text is, but not necessarily how it should look. Although common renderings exist for logical text elements, the ambiguity of these elements and the limited knowledge of this type of document structuring have reduced their use. However, the acceptance of style sheets and the growing diversity of user agents mean using logical elements makes more sense than ever.

Physical Character-Formatting Elements

Sometimes you might want to use bold, italics, or other font attributes to set off certain text, such as computer code. HTML and XHTML support various elements that can be used to influence physical formatting. The elements have no meaning other than to make text render in a particular way. Any other meaning is assigned by the reader.

The common physical elements are listed in Table 3-1. Note, as shown in the table, under the strict variants of HTML and XHTML, the **<s>**, **<strike>**, and **<u>** tags are deprecated and should not be used.

The following example code shows the basic use of the physical text-formatting elements:

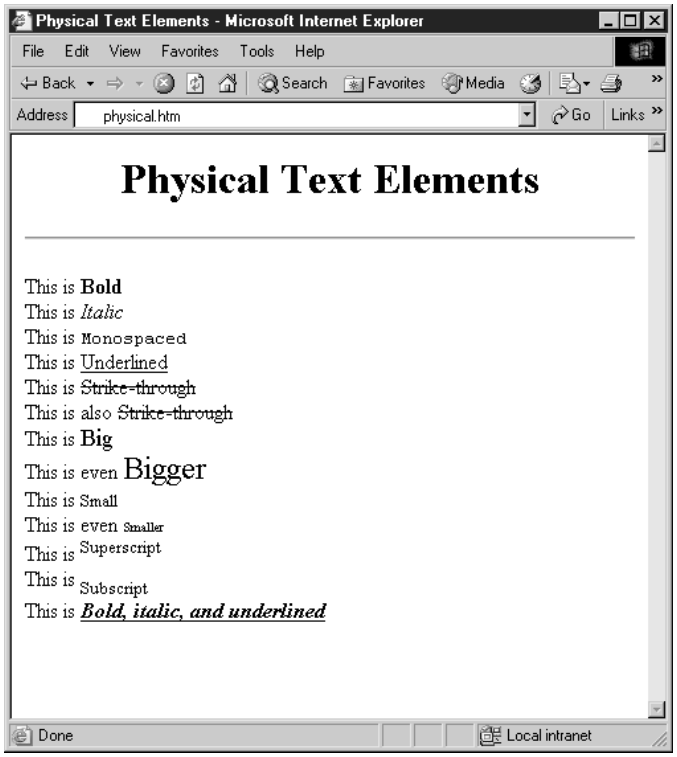
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Physical Text Elements</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h1 align="center">Physical Text Elements</h1>
<hr />
<p>
This is <b>Bold</b>                             <br />
This is <i>Italic</i>                           <br />
This is <tt>Monospaced</tt>                     <br />
This is <u>Underlined</u>                       <br />
This is <strike>Strike-through</strike>         <br />
This is also <s>Strike-through</s>              <br />
This is <big>Big</big>                          <br />
This is even <big><big>Bigger</big></big>       <br />
This is <small>Small</small>                    <br />
This is even <small><small>Smaller</small></small><br />
This is <sup>Superscript</sup>                  <br />
This is <sub>Subscript</sub>                    <br />
This is <b><i><u>Bold, italic, and underlined</u></i></b>
</p>
</body>
</html>
```

Physical elements can be combined in arbitrary ways, as shown in the final part of the example. However, just because text *can* be made monospaced, bold, italic, and superscript doesn't mean that various types of formatting *should* be applied to text. Figure 3-12 shows the rendering of the physical text elements under popular browsers.

Element	Meaning	Notes
<i> … </i>	Italics	
<b> … </b>	Bold	
<tt> … </tt>	Teletype (monospaced)	
<u> … </u>	Underline	Deprecated in HTML and XHTML strict variants.
<s> … </s>	Strikethrough	Deprecated in HTML and XHTML strict variants.
<strike> … </strike>	Strikethrough	Deprecated in HTML and XHTML strict variants.
<sub> … </sub>	Subscript	
<sup> … </sup>	Superscript	
<big> … </big>	Bigger font (one size bigger)	
<small> … </small>	Smaller font (one size smaller)	

TABLE 3-1 Inline Physical Tags

FIGURE 3-12 Rendering of physical text formatting elements


Physical Text Elements - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address physical.htm Go Links
Done
Local intranet

Physical Text Elements

This is **Bold**
This is *Italic*
This is `Monospaced`
This is Underlined
This is ~~Strike-through~~
This is also ~~Strike-through~~
This is **Big**
This is even Bigger
This is Small
This is even Smaller
This is $^{\text{Superscript}}$
This is $_{\text{Subscript}}$
This is ***Bold, italic, and underlined***

Several physical text formatting elements—particularly **<u>**, **<big>**, and **<small>**—present certain problems that warrant extra discussion.

Confusion Caused by Underlining

Today most browsers support the **<u>** tag, which underlines text. It was not initially defined under HTML 2.0, but was later introduced by Microsoft Internet Explorer and soon after all browsers began to support it. The trouble with this tag is that the meaning of underlined text can be unclear to people who use the Web. In most graphical browsers, clickable hypertext links are represented as blue underlined text. (Link color might vary.) Users instinctively think of underlined text as something that can be clicked. Some feel that the link color sufficiently distinguishes links from text that is underlined purely for stylistic purposes. However, this doesn't take into consideration monochrome monitors or people who are colorblind. Because the underline element could introduce more trouble than it is worth, it should be avoided even if it's allowed in the variant of HTML or XHTML you are using. Fortunately, under strict variants of HTML and XHTML, the tag is removed.

Using <big> and <small>

What do the **big** and **small** elements actually do? On the face of it, enclosing content within a **<big>** tag pair makes it bigger. Putting the **<small>** tag around something makes it smaller. What about when multiple **<big>** and **<small>** tags are nested? HTML has relative fonts ranging from size 1, very small, to size 7, very large. Every application of **<big>** bumps up the font one notch to the next level. The default font for a document usually is relative size 3, so two applications of **<big>** would raise the font size to 5. Multiple occurrences of **<small>** do the opposite—they make things one size smaller. HTML authors familiar with the **<font>** tag discussed in Chapter 6 should note that **<big>** is equivalent to **<font size="+1">** and **<small>** is equivalent to **<font size="-1">**.

Logical Elements

Logical elements indicate the type of content that they enclose. The browser is relatively free to determine the presentation of that content, although there are expected renderings for these elements that are followed by nearly all browsers. Although this practice conforms to the design of HTML, there are issues about designer acceptance. Plain and simple, will a designer think **<strong>** or **<b>**? As mentioned previously, HTML purists push for **<strong>** because a browser for the blind could read strong text properly. For the majority of people coding Web pages, however, HTML is used as a visual language, despite its design intentions. Even when logical elements are used, many developers assume their default rendering in browsers to be static. **<h1>** tags always make something large in their minds. Little encourages Web page authors to think in any other way. Consider that until recently, it was almost impossible to insert a logical tag using a WYSIWYG HTML editor.

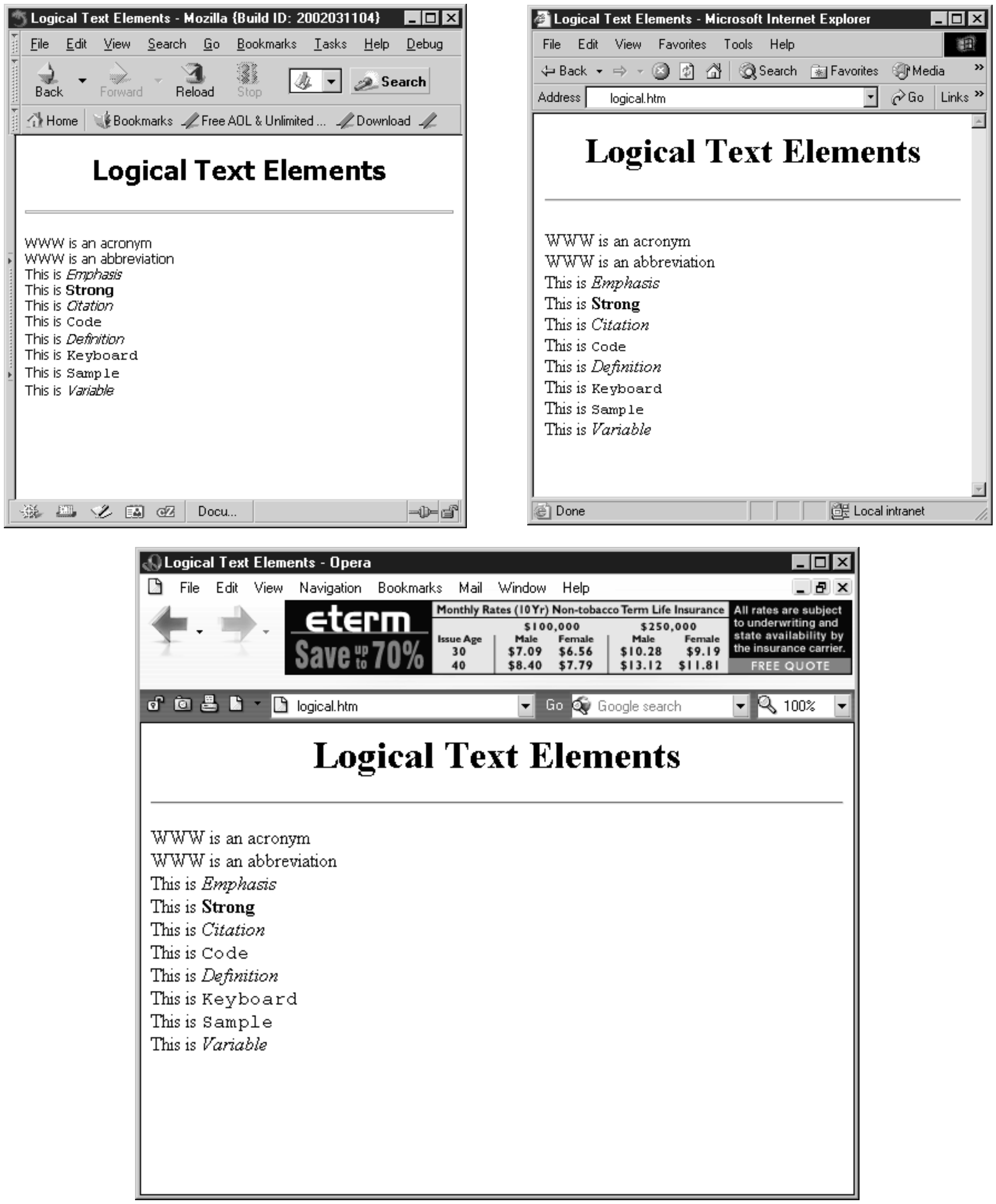
Seasoned experts know the beauty and intentions behind logical elements, and with style sheets logical elements will continue to catch on and eventually become the dominant form of page design. Even at the time of this writing, a quick survey of large sites shows that logical text elements are relatively rare. However, to embrace the future and style sheets, HTML authors should strongly reexamine their use of these elements. Table 3-2 illustrates the logical text-formatting elements supported by browsers.

Element	Meaning	Common Rendering
<abbr> … </abbr>	Abbreviation (for example, Mr.)	Plain
<acronym> … </acronym>	Acronym (for example, WWW)	Plain
<cite> … </cite>	Citation	Italics
<code> … </code>	Code listing	Fixed Width
<dfn> … </dfn>	Definition	Italics
<em> … </em>	Emphasis	Italics
<kbd> .. </kbd>	Keystrokes	Fixed Width
<q> … </q>	Inline quotation	Quoted (not in IE 6)
<samp> … </samp>	Sample text (example)	Fixed Width
<strong> … </strong>	Strong emphasis	Bold
<var> … </var>	Programming variable	Italics

TABLE 3-2 Logical Text Formatting Elements

The following example uses all of the logical elements in a test document (shown in Figure 3-13 under common browsers):

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

<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Logical Text Elements</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h1 align="center">Logical Text Elements</h1>
<hr />
<p>
  <acronym>WWW</acronym> is an acronym   <br />
  <abbr>Mr.</abbr> is an abbreviation    <br />
  This is <em>Emphasis</em>              <br />
  This is <strong>Strong</strong>       <br />
  This is a <cite>Citation</cite>        <br />
  This is <code>Code</code>              <br />
  This is <dfn>Definition</dfn>          <br />
  This is <kbd>Keyboard</kbd>            <br />
  This is a <q>Quotation</q>             <br />
  This is <samp>Sample</samp>            <br />
  This is <var>Variable</var>            <br />
</p>
</body>
</html>
```


Logical Text Elements - Mozilla {Build ID: 2002031104}
File Edit View Search Go Bookmarks Tasks Help Debug
Back Forward Reload Stop Search
Home Bookmarks Free AOL & Unlimited ... Download
Docu...
Logical Text Elements - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address logical.htm Go Links
Done Local intranet
Logical Text Elements - Opera
File Edit View Navigation Bookmarks Mail Window Help
logical.htm Go Google search 100%

Logical Text Elements

WWW is an acronym
WWW is an abbreviation
This is *Emphasis*
This is **Strong**
This is *Citation*
This is `Code`
This is *Definition*
This is `Keyboard`
This is `Sample`
This is *Variable*

FIGURE 3-13 Rendering of logical text formation under Mozilla, Internet Explorer, and Opera

Subtle differences might occur in the rendering of logical elements. For example, **<dfn>** results in Roman text under Netscape, but yields italicized text under Internet Explorer. **<q>** wraps quotes around content, but does not change rendering in Internet Explorer 6 or earlier. You should also note that the **<abbr>** and **<acronym>** tags lack default physical presentation in browsers. Without CSS, they have no practical meaning, save potentially using the **title** attribute to display the meaning of the enclosed text. In short, there is no guarantee of rendering, and older versions of browsers in particular can vary on inline logical elements, including common ones such as **<em>**. Consult Appendix A for any special browser support details you may encounter.

Inserted and Deleted Text

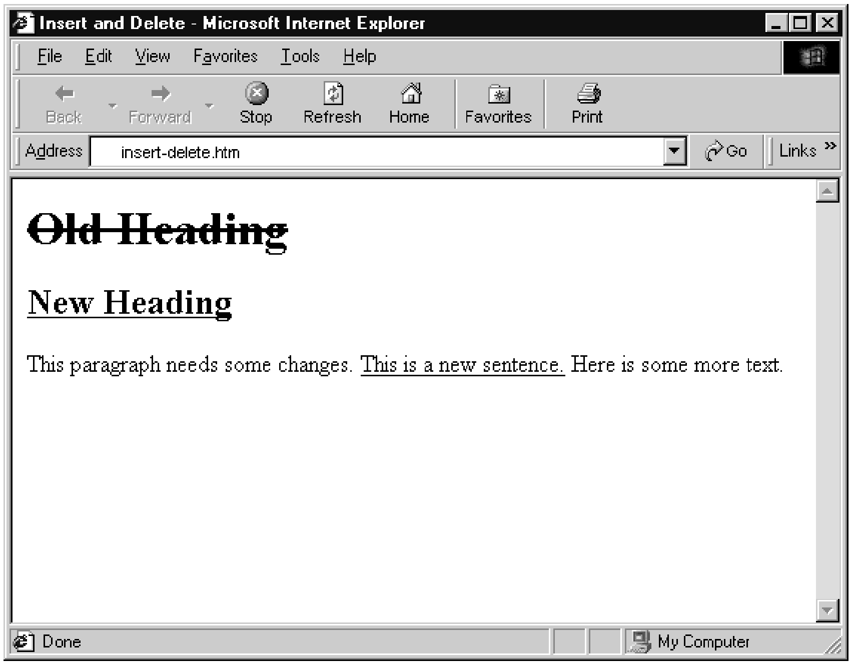
HTML 4.0 introduced elements to indicate inserted and deleted text. The **<ins>** tag is used to show inserted text and might appear underlined in a browser, whereas the **<del>** tag is used to indicate deleted text and generally appears as struck text. For example, the markup here

```
<p>There are <del>6</del><ins>5</ins> robot models</p>
```

shows how a small modification could be made to content using these two elements. It is possible to provide more information about the insert or delete through the use of the core attribute **title**, which could provide advisory information about the text change as well as the **datetime** attribute, which could be used to indicate when the change happened. Through the use of scripting, it should be possible to hide various revisions made with this element.

A complete example of the use of **<ins>** and **<del>** is shown here; a rendering is given in Figure 3-14. It is important to note that these elements are very difficult to characterize because they can contain any amount of block or inline elements.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Insert and Delete</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<del><h1>Old Heading</h1></del>
<ins><h2>New Heading</h2></ins>
<p>This paragraph needs some changes.
<ins datetime="1999-01-05T09:15:30-05:00"
        title="New info inserted by TAP.">
This is a new sentence.</ins>
Here is some more text.</p>

</body>
</html>
```


Insert and Delete - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Favorites Print
Address insert-delete.htm
Go Links
Done
My Computer

~~Old Heading~~

New Heading

This paragraph needs some changes. This is a new sentence. Here is some more text.

FIGURE 3-14 Rendering of Insert-Delete example

Character Entities

After covering most of the block elements and the basic inline text formatting elements, you might think that nothing remains to talk about—but there is one more level to HTML and XHTML documents: the characters themselves.

Sometimes, you need to put special characters within a document, such as accented letters, copyright symbols, or even the angle brackets used to enclose HTML elements. To use such characters in an HTML document, they must be "escaped" by using a special code. All character codes take the form ***&code;***, in which ***code*** is a word or numeric code indicating the actual character that you want to put onscreen. Some of the more commonly used characters are shown in Table 3-3.

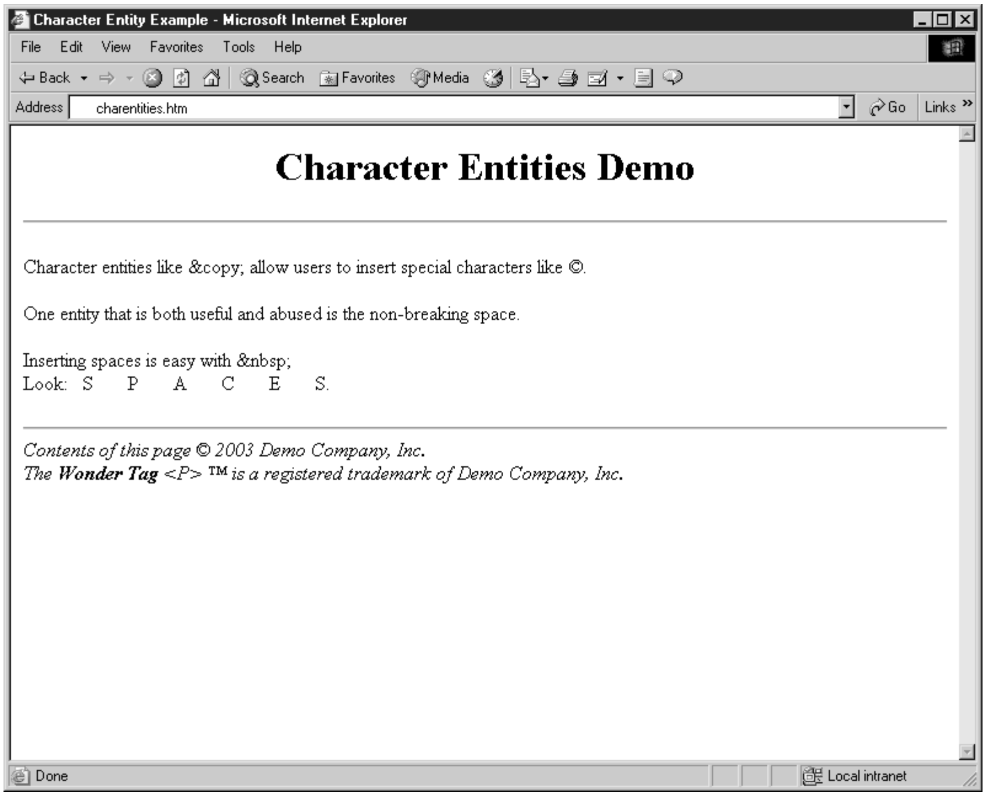
NOTE *The character entity **&#153;** might not always be acceptable as trademark. On many UNIX platforms, and potentially on Macs or Windows systems using various "other" character sets, this entity doesn't render as trademark. **&153;** can be undefined and you may even find trouble with the **&trade;** named entity. However, trademarks are important legally, so they often are needed. The commonly used workaround is **<sup><small>**TM**</small></sup>**. This markup creates a superscript trademark symbol (™) in a slightly smaller font. Because it's standard HTML, it works on nearly every platform.*

Numeric Value	Named Value	Symbol	Description
&#034;	&quot;	“	Quotation mark
&#038;	&amp;	&	Ampersand
&#060;	&lt;	<	Less than
&#062;	&gt;	>	Greater than
&#153;	&trade;	™	Trademark
&#160;	&nbsp;		Non-breaking space
&#169;	&copy;	©	Copyright symbol
&#174;	&reg;	®	Registered trademark

TABLE 3-3 Commonly Used Character Entities

The following example shows some basic uses of HTML character entities. Figure 3-15 shows how the example might render.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Character Entity Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h1 align="center">Character Entities Demo</h1>
<hr />
<p>Character entities like &amp;copy; allow users to insert
special characters like &copy;.</p>

<p>One entity that is both useful and abused is the
non-breaking space.</p>

<p>Inserting spaces is easy with &amp;nbsp;<br />
Look: &nbsp; S &nbsp; &nbsp; &nbsp; P &nbsp; &nbsp; &nbsp;
A &nbsp; &nbsp; &nbsp; C &nbsp; &nbsp; &nbsp; E
&nbsp; &nbsp; &nbsp; S.<br />
</p>

<hr />
<address>
Contents of this page &copy; 2003 Demo Company, Inc.<br />
The <strong>Wonder Tag</strong> &lt;P&gt; &#153; is a registered
 trademark of Demo Company, Inc.
</address>

</body>
</html>
```


Character Entity Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address charentities.htm
Go Links
Done
Local intranet

Character Entities Demo

Character entities like &copy; allow users to insert special characters like ©.

One entity that is both useful and abused is the non-breaking space.

Inserting spaces is easy with &nbsp;
Look: S P A C E S.

Contents of this page © 2003 Demo Company, Inc.
*The **Wonder Tag** <P> ™ is a registered trademark of Demo Company, Inc.*

FIGURE 3-15 Rendering of character entities example

NOTE *The use of the nonbreaking space to push text or elements around the screen is an overused crutch. Many HTML editors overuse this technique in an attempt to preserve look and feel.*

While entities are easy to add, excessive use can make markup difficult to read, particularly if the character entities aren't well spaced.

The character set currently supported by HTML is the ISO Latin-1 character set. Many of its characters, such as accents and special symbols, cannot be typed on all keyboards. They must be entered into HTML documents by using the appropriate code. Even if the character in question is supported on the keyboard (for example, the copyright symbol), simply typing the symbol into the document directly may not produce the correct encoding. Of course, many HTML editors make the appropriate insertion for you. A complete list of the character entities is presented in Appendix C.

NOTE *HTML is capable of representing the standard ASCII characters and all the extended characters defined by the ISO Latin-1 character set. However, for non-Western characters, such as Japanese, Russian, or Arabic alphabets, special encoding extensions to a browser or operating system may be required.*

Comments

The last topic that should be considered a core aspect of HTML is the use of comments in an HTML document. The contents of HTML comments are not displayed within a browser window. Comments are denoted by a start value of **<!--** and an end value of **-->**. Comments can be many lines long. For example,

```
<!--
Document Name: Sample HTML Document
Author: Thomas A. Powell
Creation Date: 1/5/00
Last Modified: 5/15/03

Copyright 2000-2003 Demo Company, Inc.
-->
```

is a valid comment. Be careful to avoid putting spaces between the two dashes or any additional exclamation or dashes points in the comment. Comments are useful in the **<head>** of a document to describe information about a document, as the previous example suggested. Comments might also be useful when trying to explain complex HTML markup.

Comments also can include HTML elements, which is very useful for hiding new HTML elements from older browsers, and is commonly used with **<style>** and **<script>** elements, discussed in Chapters 10 and 14, respectively. Recall that a typical browser will output the contents of any tag it doesn't understand onscreen. Thus, we may want to hide the contents of certain tags from nonsupporting browsers. For example, consider trying to hide a style sheet's contents from an older browser. The **<style>** element might occur in the **<head>** of the document and contain various style rules, as shown here:

```
<style type="text/css">
  h1   {font-size: 48pt; color: red;}
</style>
```

A simple use of an HTML comment could mask the content from nonsupporting browsers like so:

```
<style type="text/css">
<!--
h1   {font-size: 48pt; color: red;}
-->
</style>
```

In this case, style sheet-aware browsers are smart enough to know to look within a comment found directly inside the **style** element, whereas older browsers would just skip over the comment, and nothing would happen. The use of comments to mask script code is slightly different as the closing --> symbols are valid code under JavaScript. In this case, hidden script code is wrapped in <!-- //-->. The close comment is preceded with a // because // is a JavaScript comment that hides the --> from the script interpreter.

```
<style type="text/javascript">
<!--
```

PART II

```
 alert("Hello I am JavaScript");
//-->
</style>
```

Unfortunately, this comment-out trick does have some problems. Under strict interpretation of XHTML, you should use a CDATA section to mask the content, as shown here:

```
<script type="text/javascript">
<![cdata[
  alert("Hello I am JavaScript");
]]>
</script>
```

It's interesting that this is not supported in the 6 and 7.*x* generation browsers so far; thus, the traditional hiding approach should be used if scripts are embedded in XHTML documents. However, in reality, linked scripts should be used wherever possible to more cleanly separate markup from programming logic.

Summary

The HTML and XHTML elements presented so far are common across nearly all systems. Whether or not they are used, they are simple and widely understood. Yet, despite their simplicity, many of these basic elements are still abused to achieve a particular look within a document, which continues the struggle between the logical and physical nature of HTML. Despite some manipulation, these elements generally are used in a reasonable manner. More complex formatting elements and programming elements are introduced in later chapters. The simplicity of this chapter should provide you with some assurance that HTML rests on a stable core.

A great number of elements have been left out of this discussion. No mention was made of layout-oriented elements, and so far I've completely avoided graphics. These topics and others are covered in upcoming chapters. First, in the next chapter, we'll cover the "H" in HTML, namely hypertext, and explore the concept of linking documents and objects.