Cross-Ref

More information on using styles to influence backgrounds and colors can be found in Chapter 33. ■

Also, keep in mind that using a color property on one element nested within another will change only the color of the nested element, and perhaps elements nested within it (its children). For example, the table shown in Figure 12-1 has the following style added to its opening tag:

```
style="background-color: yellow;"
```

As you can see, the background of the table has been changed. However, the document background remains unaffected.

Image Formats for the Web

Most user agents support, to some degree, three graphics file formats: GIF, JPEG, and PNG. The GIF and JPEG formats have been supported for quite some time (since the origin of the Web), whereas PNG is relatively new. This section covers the basics of these image formats.

Image compression

All three of these graphics file formats use some form of compression to store your image. Why is compression important? Uncompressed images can be large — consider Table 12-3, which compares image dimensions, number of colors, and file size for some sample, uncompressed images.

TABLE 12-3

Uncompressed Image File Size Comparison by Dimensions and Number of Colors

Dimensions (in Inches)	Colors	File Size
1 × 1	2	9K
1 × 1	256	9K
1 × 1	16.7 million	18K
2 × 2	2	16K
2 × 2	256	24K
2 × 2	16.7 million	63K
3 × 3	2	16K
3 × 3	256	49K
3 × 3	16.7 million	139K

As you can see, with file sizes like this, you would have to limit yourself to mighty tiny images, or two-color, black-and-white images. Or, you could compress the files.

Note

With the predominance of broadband in the workplace and in homes worldwide, keeping documents and images small has become an afterthought for designers, not a primary concern. However, the number of graphics on a page has also increased, in some cases exponentially. The net result is the same: bulky, graphical Web documents that can make browsing the Web a tedious prospect.

When designing pages, you should also consider the new wave of user agents — deployed on handheld devices such as cell phones. These devices do not possess speedy processors and blazing fast connections. If you don't keep your pages slim, or offer specialized content for such devices, you risk alienating this relatively new, but growing, audience.

Although it might seem "old school," consider using text instead of graphics wherever possible. ■

Compression options

When you implement file compression, you either have to throw away some information about the image or find a way to store the existing information about the image in a more intelligent manner. GIF files throw away some color information. JPEG files throw away some information about the image itself. PNG files store the information using a more intelligent algorithm.

GIF

GIF was the earliest format in use in inline images on the Web. Version 1 browsers could open GIF images inline, but required that JPEG images be opened out-of-line. GIF uses a compression scheme — called *LZW compression* — that predates CompuServe, even though you might see it called CompuServe GIF. CompuServe implemented LZW compression thinking it was in the public sphere and then found out it was proprietary. A lot of lawyers sorted it out.

How does GIF work? Simply put, GIF indexes images to an 8-bit palette. The system palette is 256 colors. Before you can save your file in GIF format, the utility you are using simply makes its best guess at mapping all your colors to one of the 256 colors in an 8-bit palette.

Is a reduction in color depth a problem? That depends. GIF uses dithering to achieve colors between two colors on the palette. Even with dithering, however, GIF images of a sunset have stripes of color where a smooth gradation would be more natural. GIF images also tend to have more cartoonish colors because flesh tones aren't part of the palette. A GIF image of a drawing of something like a checkerboard, however, will look just fine.

One distinct advantage the GIF format offers is transparency. This feature enables part of the feature to appear transparent when rendered, revealing the elements below the figure through the transparent areas of the figure. As a result, the figure can be more seamlessly

incorporated into the page's design because there is no obvious rectangular area encapsulating the figure.

Note

Transparency in images is covered in the section, "Using transparency" later in this chapter. ■

JPEG

JPEG takes a different approach than that of the GIF format. JPEG stands for the *Joint Photographic Experts Group*, the name of the group that created the standard. With JPEG, you get to keep all your colors, but you don't get to keep all the data about the image. What kinds of images lend themselves to being compressed with JPEG? Most any image that doesn't require absolute detail works well in a JPEG format. Keep in mind that the loss is typically very minor and unnoticeable to all but those trained in photography or graphic arts. Of course, there are some images for which you should avoid using JPEG, including text, schematic drawings, and any line art.

Note

Most graphics packages give the user the option of choosing the level of compression when saving an image in JPEG format. A good rule of thumb is that the more compression selected, the more detail that is lost in the final image. Therefore, try to save your JPEG images with a compression level of 30 percent or lower, unless absolutely necessary. ■

Every user agent, version 3 and later, can handle inline JPEGs. JPEGs are also ideal for showing gradient-filled graphics (when the color changes gradually from one color to another). The same graphic would suffer enormously under the GIF format because the color depth wouldn't support all the in-between colors.

PNG

The *Portable Network Graphics*, or *PNG* format, was developed exclusively for the Web and is in the public domain. The PNG format takes advantage of a clever way to store information about the image so you don't lose as much color or image quality. As a lossless format, images in PNG format tend to be larger than those in the JPEG format.

The adoption of PNG graphics got off to a rocky start because of slow adoption by user agents and incomplete support for its features, such as levels of transparency. However, the latest crop of desktop browsers fully support the PNG standard and all its advantages. Keep in mind that mobile browsers, kiosks, and older browsers still lack adequate PNG support, if they have it at all.

The PNG format supports a variety of transparency options, but does not have any animation features.

Note

Fireworks, a graphics editing package from Adobe, uses the PNG format for its natively saved files. Fireworks embeds unique metadata in the saved PNG file to keep track of objects and features used in the image, such as tweening. However, the raw image saved by Fireworks should not be used in your Web documents. Use the export feature of Fireworks to save a Web-suitable file with the advanced metadata stripped out. ■

Creating Graphics

If you need to create top-notch graphics, the tool of choice among professionals is Adobe Photoshop, available for the Mac and the PC. Freeware and shareware software programs also are available that perform subsets of the functions performed by Photoshop. Photoshop LE, the "lite" version of Photoshop, ships with many scanners. Photoshop Elements — primarily used for photo editing — ships with many digital cameras.

Essential functions

What should your graphics package be able to do? For existing images, such as photographs, you want to sharpen, blur, and perform some special effects on the image (for example, posterize, swirl, and mosaic). For images you create on the screen, you want to create your own custom palette (so you can send as few colors as you need). You also need some basic artist tools, such as a paintbrush, a pencil, a spray can, and a magnifying glass to enlarge parts of the image to see it better.

Keep in mind that it takes a bit of skill and training to be able to use such features effectively. If your needs or skill level are meager, consider a lower-end, cheaper, less complicated package to begin with. However, be sure to pick a package that supports any graphic images you wish to edit, and one that is able to export to the major Web formats — GIF, JPEG, and PNG.

Tip

If you aren't ready to commit to a $500 software package to get all these great functions, you can work with a number of small, free software packages and services that perform many of the tasks previously listed. On the Web, you can find sites that turn your TIF file into a GIF, or make your GIF an interlaced GIF. The trade-off is the time. Finding, learning, and using a variety of small packages to solve all your imaging needs obviously takes longer than learning one package and using it on your desktop. ■

Capturing Images from Other Sites

As you build your documents you may be tempted to borrow images from other sites. The temptation is common; the Web is rich with content that seems "just perfect for your use."

However, this is not a good practice. In general, unless you clearly own the image in question, you cannot use it for *any* purpose. Using an image from another source requires express (usually written)

permission from the image's owner. Moreover, keep in mind that the owner of the image and the owner of the Web page on which it appears may be two different individuals. When it comes down to a suit in copyright court, you would bear the burden of proving you had clear rights to use the image, and to use it in the way you did in your documents.

Instead, create your own images or buy suitably licensed images from one of the stock photography houses, including the following:

- Fotolia (`www.fotolia.com`)
- Getty Images (`www.gettyimages.com`)
- iStockphoto (`www.istockphoto.com/index.php`)

Progressive JPEGs and interlaced GIFs

There was a time on the Web when you had to wait for an image to finish loading before you knew what it was. Today, you can save your files using the progressive JPEG format or the interlaced GIF format and watch the image come into focus as it loads.

The advantage to this approach is that a visitor to your site knows roughly what an image is before the entire image has downloaded. If download times are long because of a poor Internet connection, for example, the site visitor can actually take a link off the page before the image has finished loading without missing anything.

Finally, these two image formats are good because the visitor participates in the download time. Watching the images become clearer as the page downloads gives visitors to the site a sense of reward for waiting.

Note

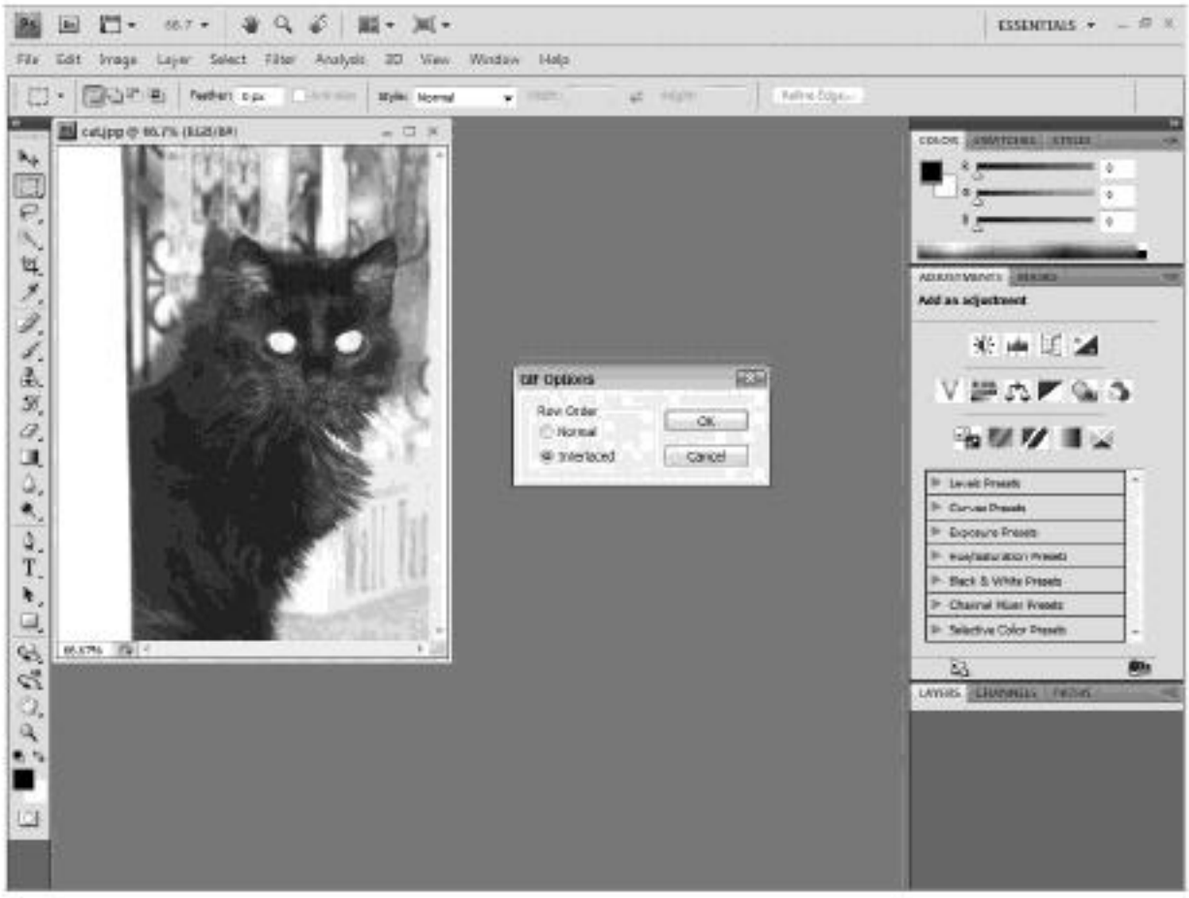
Specifying the size of the image in the image tag can also speed up the display of your Web pages, as it enables the user agent to reserve space for the image and keep rendering, instead of waiting for the entire image to load before progressing. See the "Sizing an image" section later in this chapter for more information. ■

The sense of "coming into focus" that these types of images provide is the result of the way the images are stored. Progressive JPEGs and interlaced GIFs download only every eighth line at first, then every fourth line, then every second line, and then, finally, the odd-numbered lines. As a result, the image goes from blurry to focused.

You create a progressive JPEG or an interlaced GIF by saving it into this format. In Adobe Photoshop, when you save a file as a GIF, you can choose whether you want the file to be normal or interlaced (see Figure 12-2). Freeware packages that convert your regular JPEGs and GIFs into progressive JPEGs and interlaced GIFs are also available.

FIGURE 12-2

Adobe Photoshop enables you to choose whether you want your GIF to be interlaced or not.



Using transparency

Two of the Web-supported graphics formats, GIF and PNG, support transparency, which enables parts of images to be completely transparent. Typically, transparency is used to soften the edges of images, creating an illusion that the image is not rectangular. For example, Figure 12-3 shows an image with a standard opaque background and the same figure with a transparent background. The image with transparency allows the page background to show through.

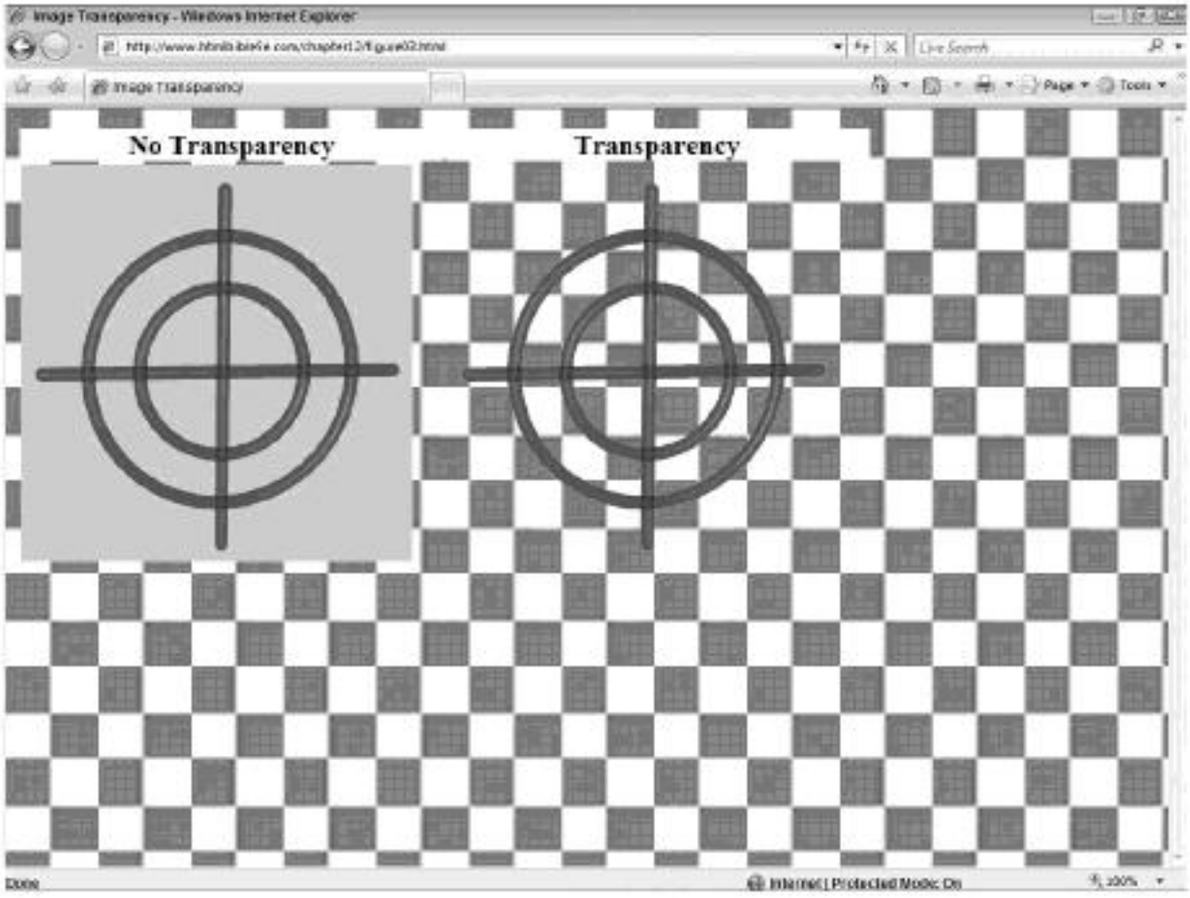
Using transparency can open up a document's design, making it more airy and less "blocky." It gives the document a more professional appearance, looking more like a published document than a Web page of the 1980s.

Different graphics editing programs handle transparency differently. Some assign transparency to the background layer; some allow you to pick one color that should be transparent; some

programs allow multiple colors to be transparent. Check the Help file for your editor to determine how to accomplish transparency.

FIGURE 12-3

Transparency can soften an image, creating the appearance that the image is not rectangular.



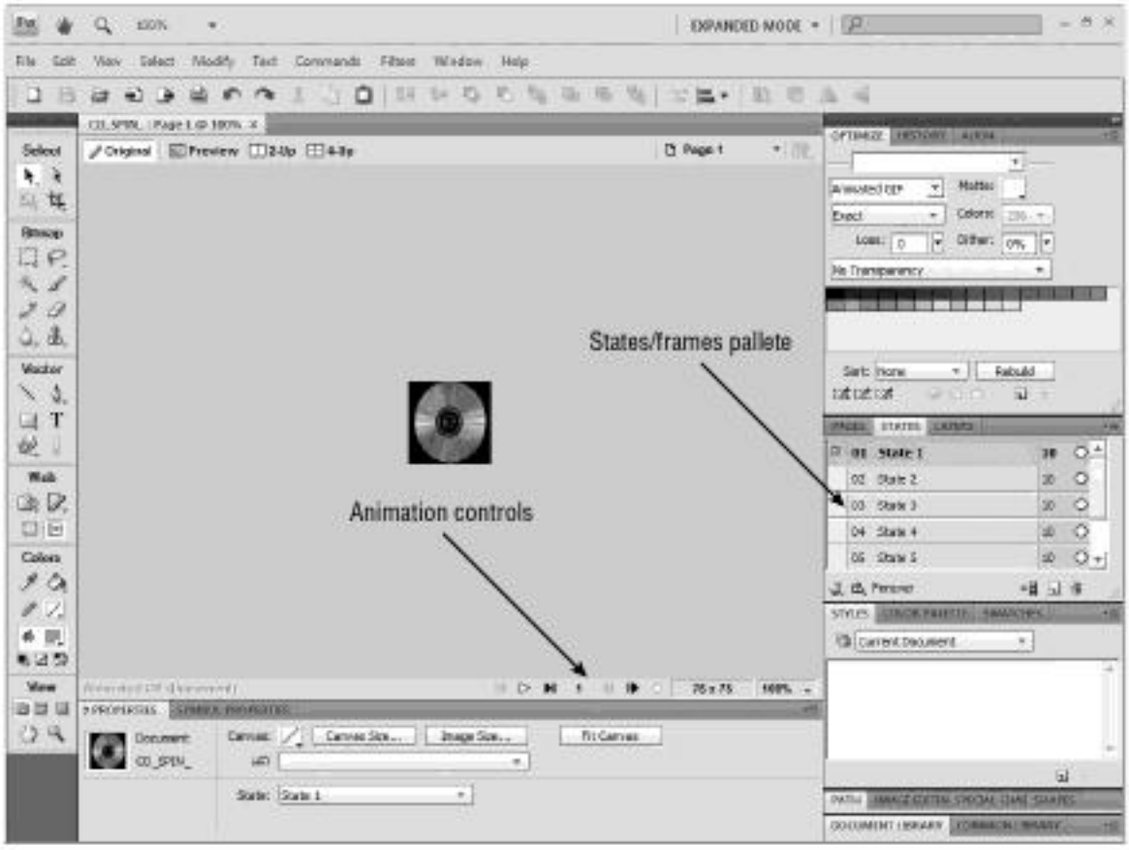
Animated images

The GIF format also supports rudimentary animation by showing different frames of an image one after another. The effect is similar to drawing individual frames of animation on different pages of a sketchbook and rapidly flipping the pages. Animated GIF images are not supported by all user agents and should be used sparingly due to their size — the image must store all the frames of the animation, increasing the size of the image with each new frame.

Some image editors, such as Adobe Fireworks, shown in Figure 12-4, include tools to help create animated GIF images.

FIGURE 12-4

Programs such as Fireworks can help you create animated GIFs, in this case the animation of a spinning CD-ROM.



Inserting an Image

Images are inserted into HTML documents using the image tag (`<img>`). This tag, at a minimum, takes two attributes, `alt` and `src`.

The `alt` attribute specifies text that should be displayed in lieu of the image in nongraphical browsers (see the section "Specifying Text to Display for Nongraphical Browsers" later in this chapter). The `src` attribute tells the user agent what image file should be displayed. For example, if you wanted to include the graphic `gunsight.jpg` in your document, you could use code similar to the following:

```
<img alt="A gunsight" src="gunsight.jpg" />
```

The `<img>` tag has no closing tag. As such, you should include the slash at the end of the tag itself.

Cross-Ref

For more information about absolute and relative URLs, see Chapter 8. ■

The src attribute's value can be any valid URL of an image on the Web — local or remote. Just as with the anchor tag, you can use absolute or relative URLs to specify the location of the image to display. The reasons for using either URL are the same as the reasons for using absolute or relative URLs in anchor tags.

Note

The src attribute of an image tag can also be a server-side program that produces a graphic. For example, some chart-producing PHP libraries work by calling the PHP script directly from an image tag, similar to:

```
<img alt="A pie chart" src="piechart.php" />
```

This allows dynamic means to produce graphics without first having to store the dynamic graphic on the server. Note that the graphic producing script or program must supply the full header and graphic for the user agent to properly render the graphic. ■

Image Alignment

Most user agents will attempt to display the image exactly where the <img> tag is inserted in the document. For example, consider the following HTML code and the resulting display shown in Figure 12-5:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
 <html>
 <head>
   <title>Default Image Placement</title>
 </head>
 <body >
 <p><img src="images/smsight.gif" height="100px" width="100px"
 alt="OTG Logo" />Rodent Studios has recently announced that their
 highly anticipated game, "Gopher Hunt," will miss the Christmas
 season and is now slated for release in early 2008. Gopher Hunt is
 the semi-sequel to "Badger Brigade," Rodent Studios smash hit of
 last year.<img src="images/smsight.gif" height="100px" width="100px"
 alt="OTG Logo" /></p>
 <p>Company president Samuel Perry could not be reached for comment,
 but all release dates for the game have been removed from the
 company website. As you may recall, "Badger Brigade" is one of the
 few titles to which On Target Games awarded 5-stars to and the only
 game to stay on the best seller list for a whopping 24 week.</p>
 <p>We hope this slip isn't a sign of bigger problem at RS, but will
 keep you in the loop.</p>
 </body>
 </html>
```

If the user agent cannot fit the image on the current line, it will wrap it to the next line and follow the paragraph's alignment and formatting.

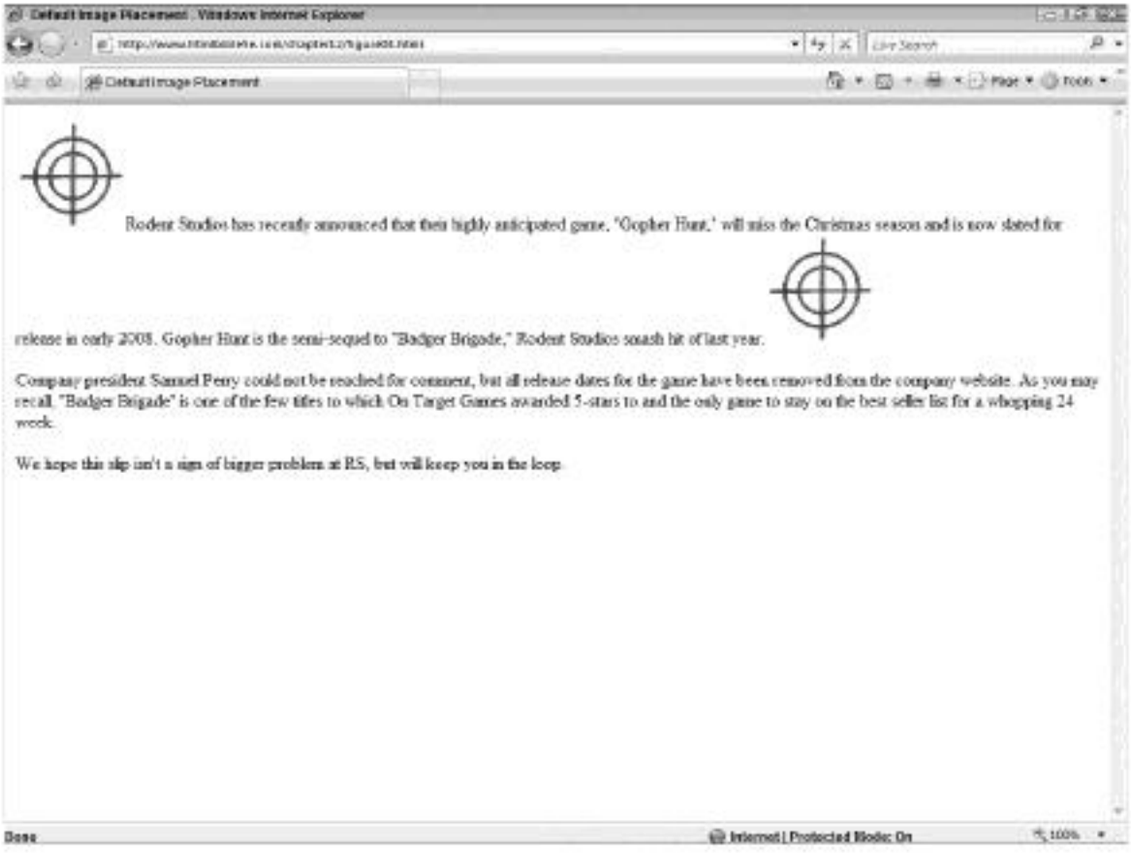
Note how the default formatting (at least for Internet Explorer) of the image is to be aligned with the baseline of neighboring text. This isn't always ideal. Sometimes you will want to specify the

image's alignment as it relates to the text and other objects around it. Image alignment can be controlled by using the `align` attribute with the `<img>` tag. The `align` attribute can be set to the values shown in Table 12-4.

Figure 12-6 shows an example of each of these alignment options.

FIGURE 12-5

The browser displays the image at the beginning and the end of the paragraph where the image tags are located.



Note

Most user agents render items in the order in which they appear in the document. If you are using left-aligned images, they should appear *before* the text that they should be positioned to the left of. ■

However, the `align` attribute has been deprecated in favor of using styles for image alignment. The following CSS properties can be used to help align images:

- `text-align` — Used in surrounding text, this property aligns the text around an image (versus aligning the image itself). See Chapter 8 for more information on using the `text-align` property.
- `float` — Floats the image to the right or left of the user agent. The `float` property allows text and other objects to wrap next to the image.
- `vertical-align` — Aligns the image vertically with neighboring text or objects.

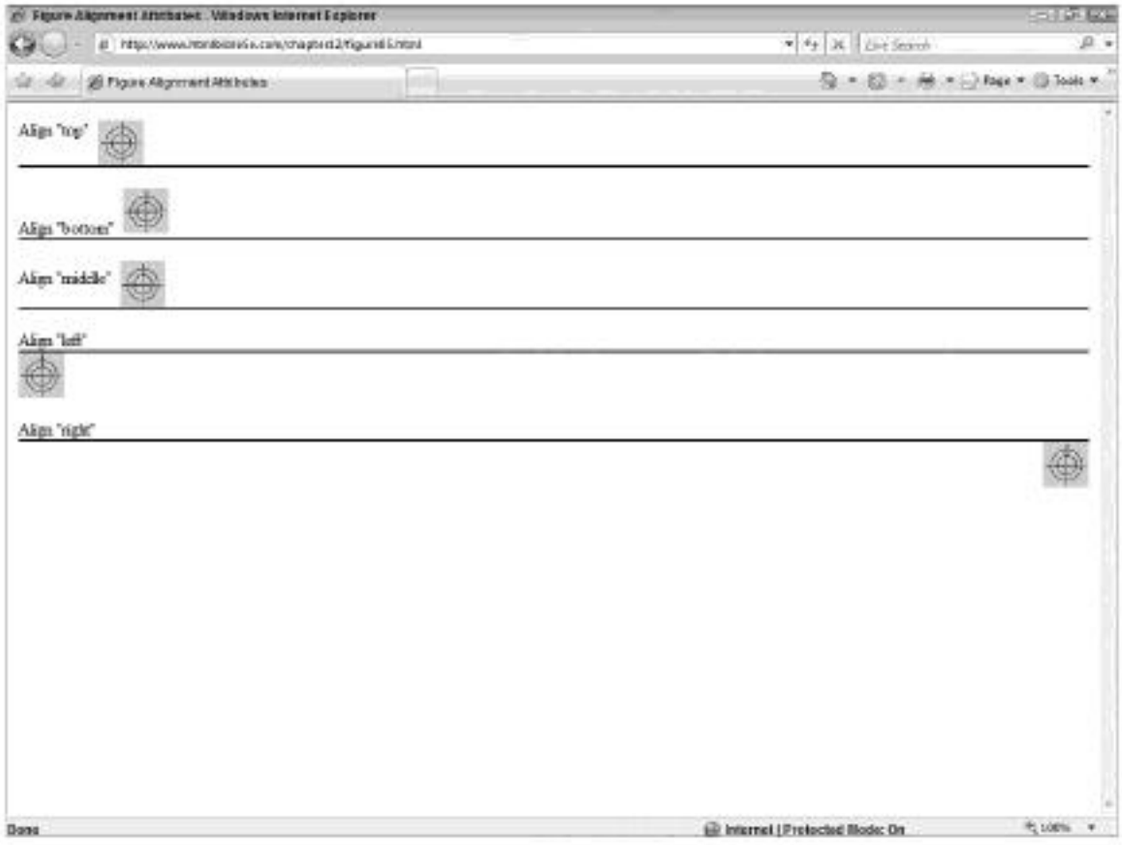
TABLE 12-4

Align Attribute Values

Value	Definition
`top`	Align with the top of nearby text or object
`bottom`	Align with the bottom of nearby text or object
`middle`	Align with the middle of nearby text or object
`left`	Align to the left of nearby text or object
`right`	Align to the right of nearby text or object

FIGURE 12-6

The various alignment options for images



Note that with the `align left` and `align right` attributes, the image is placed after the first line of its containing block, unless the image is placed first in the block. This is evident by the position of the bottom border of the paragraph tag relative to the image. However, this is not consistent behavior across browsers and shouldn't be relied upon. Also, some user agents need

to process the image alignment prior to the text around it. If you are using CSS to position your images, it is usually best to position them before neighboring text in your HTML document.

Specifying Text to Display for Nongraphical Browsers

As mentioned repeatedly in this book, it is important not to get caught up in the graphical nature of the Web, and forget that not all user agents support graphics to the extent that the Web designer would like. You should use the image tag's `alt` attribute to specify text that should be displayed when the image cannot. For example, consider the following text and the display shown in Figure 12-7:

```
<p><img src="images/smsight.gif" height="100px" width="100px"
alt="OTG Logo" style="float: right;"/>Rodent Studios has recently
announced that their highly anticipated game, "Gopher Hunt," will
miss the Christmas season and is now slated for release in early
2008. Gopher Hunt is the semi-sequel to "Badger Brigade," Rodent
Studios smash hit of last year.</p>
```

FIGURE 12-7

The alt attribute specifies text to use (in this case, "TG Logo") when the image cannot be displayed.

Some user agents display the `alt` attribute text when the user mouses over the image. This enables you to use the `alt` attribute to include additional information about an image. If you have a lot of information to convey, consider using the `longdesc` (long description) attribute, as well. The `longdesc` attribute specifies a URL to a document that is to be used as the long description for the figure. Remember that it is up to the user agent to decide how to enable access to the long description, if at all.

Sizing an Image

You can specify the size of an image by using the `height` and `width` attributes of the image tag. These attributes accept pixel and percentage values, enabling you to specify the exact size of an image or a size relative to the current size of the image's containing object.

Tip

Get in the habit of always using the `width` and `height` attributes with your `<img>` tags. These attributes enable the user agent to reserve the correct amount of space for the image while it continues to render the rest of the document. Without these attributes, the user agent must wait for the image to be loaded before continuing to load the rest of the document. ■

For example, suppose you have a large, high-resolution image but want to display a smaller version. Using the pixel values of the sizing attributes, you can specify a custom size of the larger image. Consider the following code and the resulting display in Figure 12-8:

```
<!-- Full image is 200px square -->
<p>Full Size Image<img alt="Full size image"
  src="images/mdsight.gif" width="200px" height="200px" /></p>
<p>Half Size Image<img alt="Half-size image"
  src="images/mdsight.gif" width="100px" height="100px" /></p>
```

Note

It is important to use both the correct height and width when specifying image dimensions in an `<img>` tag. If you change the proportions of the figure (by specifying a wrong width or height), you will end up with a funhouse mirror effect — the image will be stretched or shrunk in one dimension. While this can be used for effect, it is usually accidental. In addition, although pixel values are the default, it is good practice to always be specific in your measurements — that is, always include the `px` when you specify pixel values. ■

You might think that using percentage values in the width and height attributes would be an easier way to scale an image to a percentage of its size. Unfortunately, using percentage values isn't quite that intuitive — the result is an image that is the specified percentage of the available space. For example, for an image in an unconstrained paragraph (user agent screen width), using a width of 50 percent would be scaled to 50 percent of the screen's width.

You can specify only one of the dimensions and have the user agent automatically figure out the other. However, the user agent must then wait for the entire image to load before rendering the rest of the page, so it is best to specify both dimensions.

FIGURE 12-8

Using height and width values, you can display an image at any size other than its normal size, although it is better to use a properly sized image.

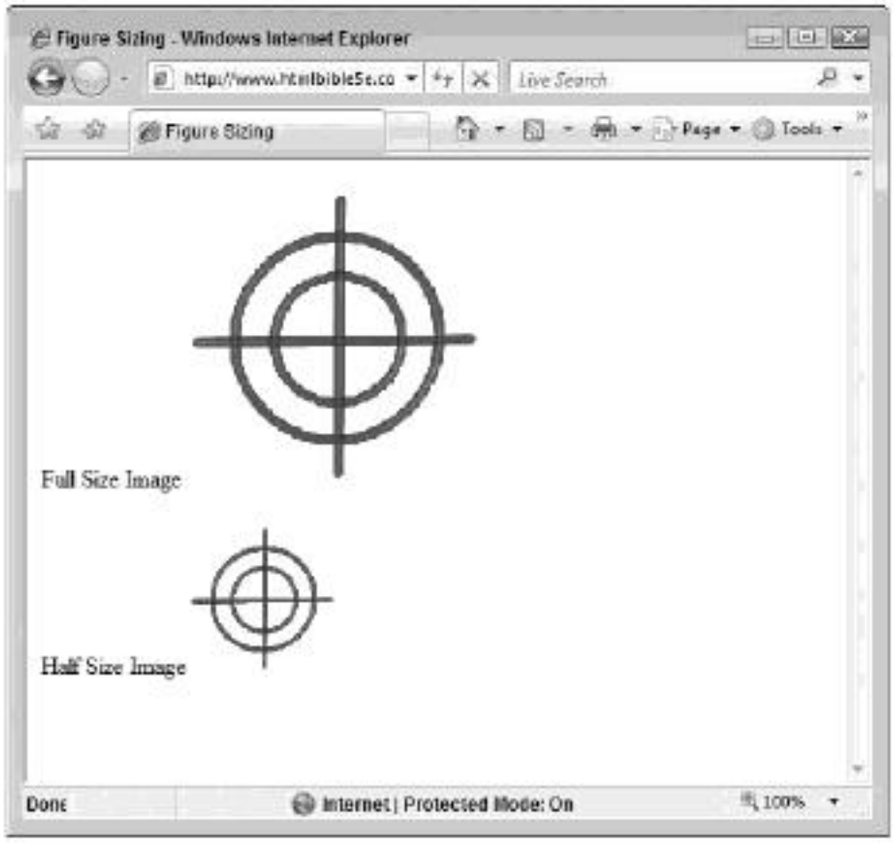


Image size attributes should not be used as a substitute for an appropriately sized graphic. If you need a different sized image, create the appropriate size in an image editor and use the new image instead. Although you can use the width and height attributes to display an image smaller than it actually is, the user agent must still download the entire image and then scale the image accordingly.

Image Borders

You can use CSS styles to create borders around images. Previous versions of HTML supported a `border` attribute for the `<img>` tag, which worked similarly to the table tag's `border` attribute. However, this attribute has been deprecated for use with the image tag. Instead, use styles. CSS supports quite a few border properties, including the following:

- All-inclusive (color, style, width) attribute setting properties:
  - `border`
  - `border-top`
  - `border-right`

    - `border-bottom`
    - `border-left`
- Color-setting properties:
    - `border-color`
    - `border-top-color`
    - `border-right-color`
    - `border-bottom-color`
    - `border-left-color`
- Style-setting (type of line used for border) properties:
    - `border-style`
    - `border-top-style`
    - `border-right-style`
    - `border-bottom-style`
    - `border-left-style`
- Width-setting properties:
    - `border-width`
    - `border-top-width`
    - `border-right-width`
    - `border-bottom-width`
    - `border-left-width`

Cross-Ref

More information on CSS borders can be found in Chapter 32. ■

For example, to define a 4-pixel-wide border around an entire image, you can use the following code:

```
<img alt="A gunsight" src="sight.jpg"
style="border: 4px solid black;" width="50px" height="50px" />
```

To define a border on just the left and right sides of an image, you would use the following:

```
<img alt="A gunsight" src="sight.jpg"
style="border-left: 4px solid black;
border-right: 4px solid black;"
width="50px" height="50px" />
```

To simplify defining a different border on one side of an image, use the `border` property first to define a border on all sides and then the appropriate `border-`*side* property for the side that is the exception, overriding the previous setting for that side. For example, to create a border on all sides of an image except the right, you could specify `border-top`, `border-bottom`, `border-left`, and `border-right` properties individually. Or, you could use just `border` and `border-right`:

```
<img alt="A gunsight" src="sight.jpg"
style="border: 4px solid black;
border-right: none;" width="50px" height="50px" />
```

Note

An image placed inside an anchor tag will display with a border that is the color of the appropriate link status. (See Chapter 8 for more information on links.) The border is designed to highlight the image as a link, but can have undesired effects on the design of your pages. If you wish to remove the border, use styles to specify no border for the image tag (that is, `border: none`). ■

Image Maps

Image maps provide a way to map certain areas of an image to actions. For example, a company might want to provide a map of the United States on its website that enables customers to click a state to find a local office or store.

There are two types of image maps, *client-side* and *server-side*. Client-side image maps rely on the user agent to process the image, the area where the user clicks, and the expected action. Server-side image maps rely on the user agent only to tell the server where the user clicked; all processing is done by an agent on the Web server.

Between the two methods, client-side image maps are generally preferred, as they enable the user agent to offer immediate feedback to the user (like being over a clickable area) and they are supported by most user agents. Server-side agents can bog down a server if the map draws consistent traffic, hides many details necessary to provide immediate feedback to the user, and might not be compatible with some user agents.

Tip

If you want an image to be clickable and take the user to one particular destination, you don't have to use an image map. Instead, embed the image tag (`<img>`)in an appropriate anchor tag (`<a>`) similar to the following:

```
<a href="catpage.html"><img alt="Link to home page"
  src="sight.jpg" /></a> ■
```

Specifying an image map

A client-side image map is generally specified within the contents of a `map` tag and linked to an appropriate `img` tag with the `<img>` tag's `usemap` attribute. For example, to specify a map for an image, `shapes.jpg`, you could use this code:

```
<img alt="Pick a Shape" src="shapes.jpg"
  usemap="#map1" />
<map name="map1">
.........
</map>
```

Inside the `<map>` tags you specify the various clickable regions of the image, as covered in the next section.

Specifying clickable regions

To specify an image map, a list of polygonal regions must be defined on an image and referenced in the HTML document. Three different types of polygons are supported: rectangle, circle, and free-form polygon.

- `rect` — Defines a rectangular area by specifying the coordinates of the upper-left and lower-right corners of the rectangle.
- `circle` — Defines a circular area by specifying the coordinates of the center of the circle and the circle's radius.
- `poly` — Defines a free-form polygon area by specifying the coordinates of each point of the polygon.

Note that all coordinates of the image map are relative to the top-left corner of the image (effectively 0, 0) and are measured in pixels. For example, consider the image shown in Figure 12-9, depicting a polygon (star), circle, and rectangle. Figure 12-10 shows the same image but with callouts — the numbered callouts indicate the coordinates necessary to map each shape.

FIGURE 12-9

An image ready to be used as an image map

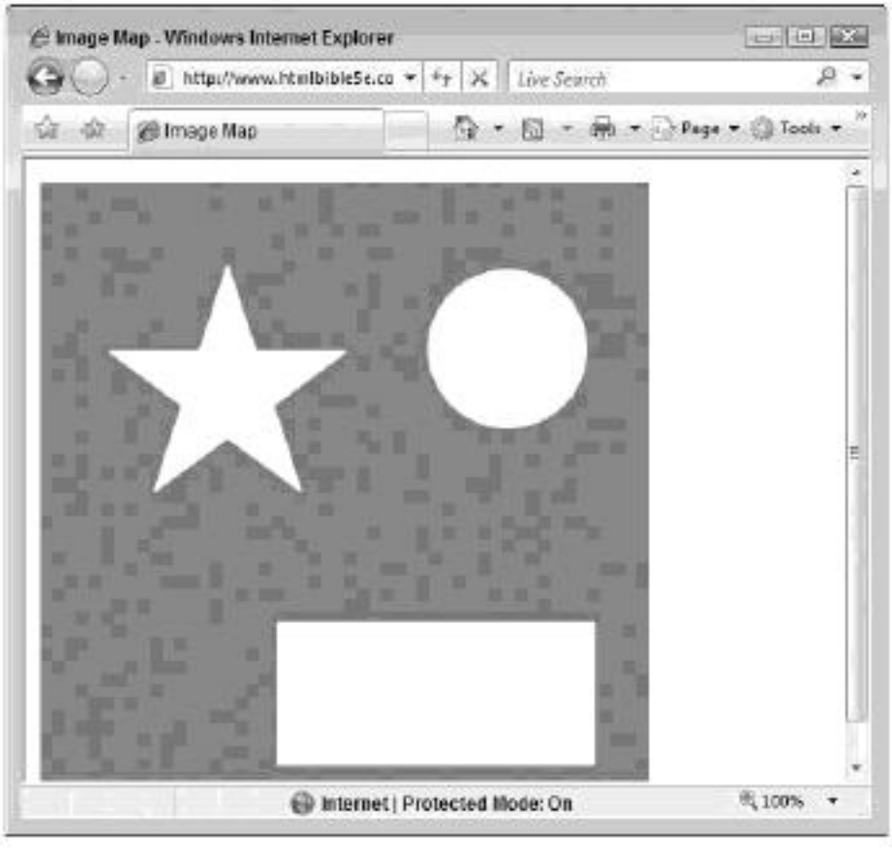
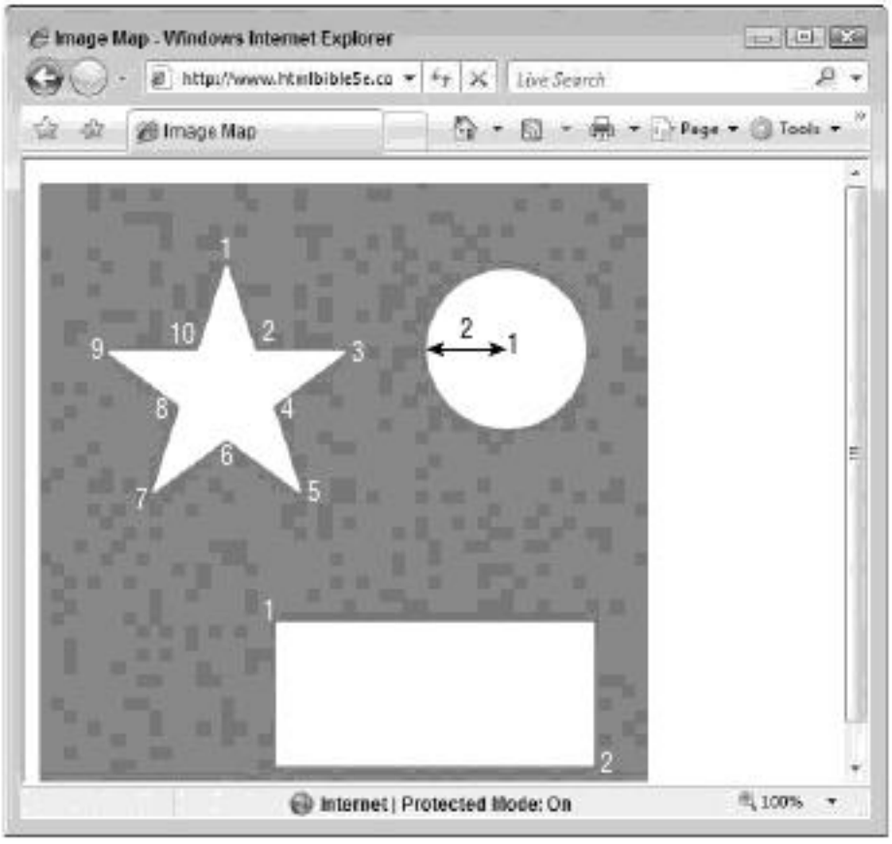


FIGURE 12-10

The points for which coordinates need to be entered for each shape



The regions that will be used for image mapping are the white areas. The coordinates that are necessary for each area are as follows:

- Polygon: 123, 54, 142, 110, 201, 111, 154, 147, 171, 203, 123, 169, 75, 204, 91, 147, 45, 110, 104, 109
- Circle: 307, 109, 53
- Rectangle: 156, 289, 364, 384

Tip

Several tools are available to help create image map coordinates. Use your favorite search engine to find a dedicated piece of software to map regions, or check your graphics program to see if it can create regions for you. ■

In a pinch, you might be able to use your graphics program's status bar. Most graphics programs display the current position of the mouse cursor in their status bar — simply point at the area of the graphic for which you need coordinates and then transpose the numbers in the status bar into your HTML code.

Specifying regions using anchor tags

You can specify regions using anchor tags with `shape` and `coords` attributes. For example, to specify the three regions previously outlined, you could use the following:

```
<map name="map1">
<a href="polygon.html" shape="poly" coords="123,54,142,
110,201,111,154,147,171,203,123,169,75,204,91,147,45,110,
104,109">Polygon Link</a>
<a href="circle.html" shape="circle" coords="307,109,53">
Circle Link</a>
<a href="rectangle.html" shape="rect" coords="156,289,364,384">
Rectangle Link</a>
</map>
```

Note that the link text helps the user determine where the clickable area links to. The user agent will typically provide a tooltip or other visual clue using the text.

Specifying regions using area tags

You can also define regions with the area (`<area>`)tags instead of anchors:

```
<map name="map1">
<area href="polygon.html"
  shape="poly" coords="123,54,142,110,201,
  111,154,147,171,203,123,169,75,204,91,147,45,110,104,109"
  alt="Polygon Link" />
<area href="circle.html"
  shape="circle" coords="307,109,53"
  alt="Circle Link" />
<area href="rectangle.html"
  shape="rect" coords="156,289,364,384"
  alt="Rectangle Link" />
</map>
```

In the case of the `<area>` tag, using the `alt` attribute helps the user determine what the clickable area leads to, usually via a tooltip when the user mouses over the area.

Putting it all together

Code for a document with a working image map (as outlined in this section) would resemble the following:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Pick a Shape</title>
</head>
```

```
<body>
<img alt="Pick a Shape" src="shape.jpg" usemap="#map1" />
<map name="map1">
<area href="polygon.html"
  shape="poly" coords="123,54,142,110,201,
  111,154,147,171,203,123,169,75,204,91,147,45,110,104,109"
  alt="Polygon Link" />
<area href="circle.html"
  shape="circle" coords="307,109,53"
  alt="Circle Link" />
<area href="rectangle.html"
  shape="rect" coords="156,289,364,384"
  alt="Rectangle Link" />
</map>
</body>
</html>
```

Note

Image maps can be used for more complex purposes, such as the clickable U.S. map mentioned earlier in this chapter, or to enable users to obtain more information, such as clicking various buildings on a map, or individual parts of a complex diagram of a machine. ■

Summary

This chapter explained the basics of image formats. You learned the benefits and drawbacks of each supported image type, as well as how to include and format them in an HTML document, bringing the basics of multimedia to your documents.

The next chapter builds on the multimedia concept by showing you how to bring full animation, movies, Flash animations, and other pieces of multimedia to the table. This part of the book wraps up with a few HTML tidbits (special characters in Chapter 14, and internationalization in Chapter 15), and then covers scripting (Chapters 16 and 17).

CHAPTER 13

Multimedia

In the early days of the Web, the word multimedia caused quite a bit of excitement. Hearing digital sounds over a previously text-only protocol was captivating. Today, the word "multimedia" simultaneously brings users excitement and dismay. The difference in the emotion encountered is driven by how much control the user has over coming in contact with the media — live or on-demand broadcast of TV shows is generally a good thing, whereas a multitude of Flash banners that bring your user agent to a standstill is a bad thing.

But that still begs the question, "What is multimedia?"

Wikipedia (`www.wikipedia.org`) defines multimedia as "media that uses multiple forms of information content and information processing (such as text, audio, graphics, animation, video, interactivity) to inform or entertain the (user) audience." So, for the Web, multimedia is anything not otherwise considered text. This includes graphics (both still and animated), audio, video, and combinations thereof. Almost any page you visit on the Web today qualifies as having multimedia content.

I've gone through this exercise for two main reasons:

- To help define what this chapter considers to be multimedia
- To illustrate that multimedia on the Web is not a complex and daunting prospect, but a very easily implemented and common one

The following sections cover the various forms of multimedia along with specific and general means to incorporate them into your documents.

IN THIS CHAPTER

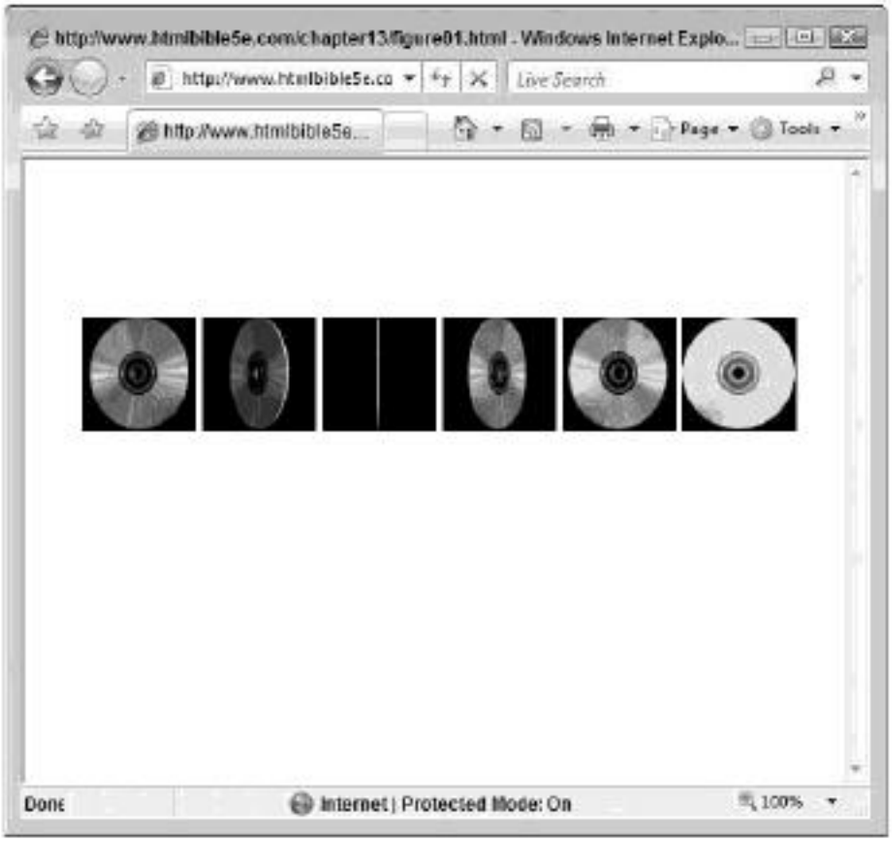
Animated Images

Animated images are a simple type of multimedia that incorporates visual animation but has no audible component. Animated images are built and displayed by frames — individual images that vary slightly, which, when played back in rapid succession, produce simulated motion. Think of the process like the flip books you might have created when you were younger, drawing images on the pages and flipping them rapidly to produce an animated effect.

Figure 13-1 shows an example of an animated image: six frames of a spinning CD animation.

FIGURE 13-1

An animation of a spinning CD, animated over six frames.



Cross-Ref

Non-animated images and the image tag (`<img>`) are covered in Chapter 12. ■

One downside to animated images is their size. Each frame in the animation increases the size of the image file incrementally. For example, if you have a 75 KB image and animate its contents

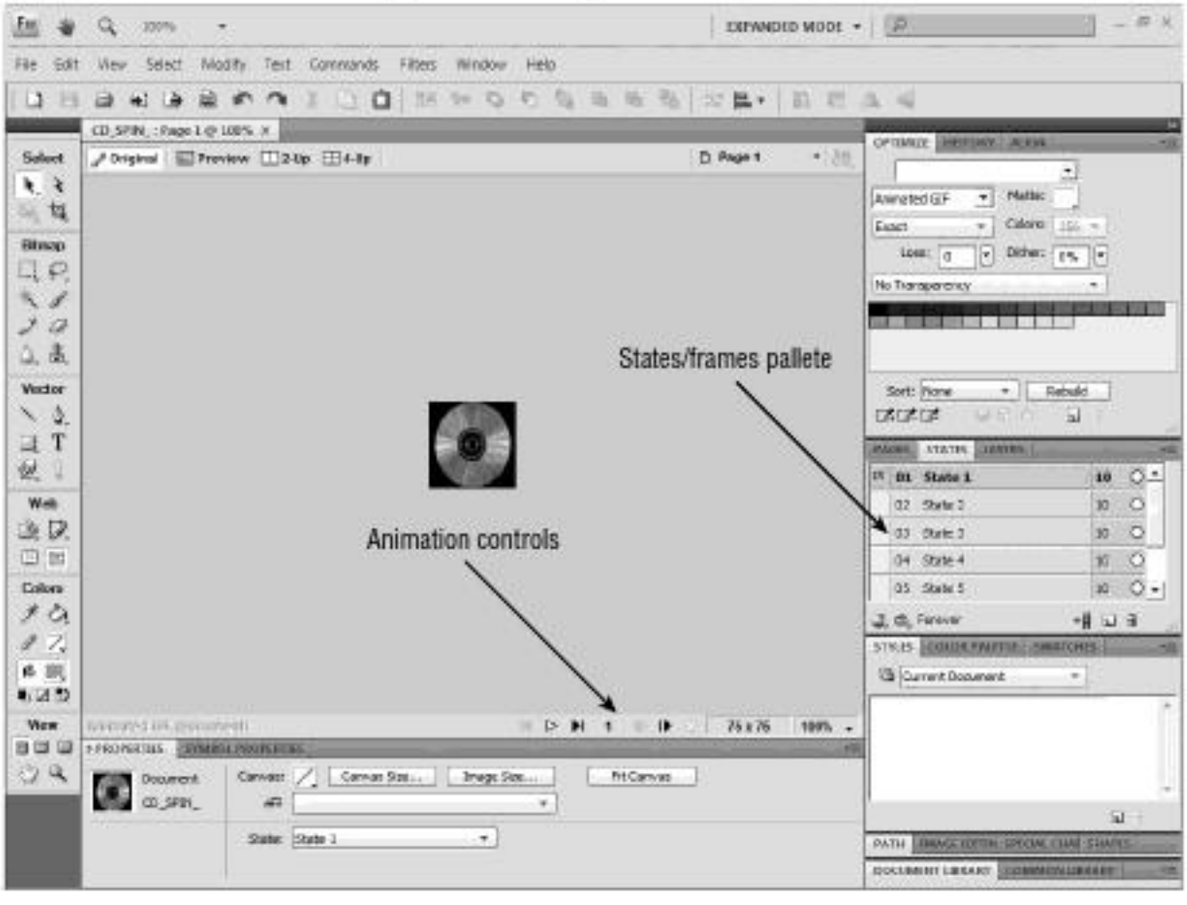
across ten frames, you would create a 750 KB file (75 × 10 = 750). With animations, it is easy to get carried away and produce even more frames, which can increase the file to a cumbersome size.

Animated images need to be stored in a Web-friendly format that supports animation. Today, only the GIF format fits this bill. You also need an image editing program capable of creating animated GIF files. Several high-end image editors are capable of creating animated images, and provide tools that aid in the image's creation.

As an example, Adobe Fireworks enables you to build an animation frame by frame (using the Edit ➤ Insert ➤ Frame menu option) and save the result in the animated GIF format. Figure 13-2 shows the spinning CD animation in Adobe Fireworks; notice the animation controls to move from frame to frame, and the frame palette showing the state of each frame and enabling easy access to each one.

FIGURE 13-2

Adobe Fireworks makes building animated images relatively easy.



Tip

If you don't want to spend the money for an image editor that includes creating animations, consider using a cheaper image editor to build the frames, and a cheap or even free utility to assemble the frames into an animated GIF file. Search utility sites such as Tucows (`www.tucows.com`) or Winfiles (`www.winfiles.com`) for suitable utilities.

You add an animated image to your documents in the same way you add any other image, using the image tag (`<img>`). Image tag specifics are covered in detail in Chapter 12, but the tag's basic syntax is shown here:

```
<img src="url_to_image" alt="alternative_text"
width="width_of_image" height="height_of_image" /> ■
```

Using the spinning CD image as an example, which is 75 pixels square, you could use the following tag:

```
<img src="cdspin.gif" alt="a spinning cd"
width="75px" height="75px" />
```

You can then use other tags and styles to place the image, control the flow of other objects around it, add ornamentation, and more.

Animation and Video Formats, Plug-ins, and Players

The standard user agents do not natively support animation outside of animated images or video. Instead, they rely on add-on programs, typically known as *plug-ins*, to bring the content to the user. The content is embedded in Web documents via tags with URLs pointing to the actual media, just like other objects and content. However, when the page is decoded and rendered by the user agent, an appropriate plug-in is required to play, view, or otherwise do something useful with the content. The process resembles the diagram shown in Figure 13-3.

The process generally proceeds like this:

1. The user agent encounters an `<object>` tag.
2. The user agent tries to determine the type of media encased in the tag via the MIME type — either specified in the `type` attribute or inferred from the media file's extension (`.avi`, `.mpg`, and so on). If the MIME type cannot be determined, then the user agent won't know what to do with the media and cannot play/display it.
3. Once the MIME type has been determined, the user agent compares the type against the players it already knows about. If an appropriate player plug-in is found, it is loaded to play/display the media.
4. If an appropriate player isn't already available, the `<object>` tag is again examined, this time for an appropriate player to acquire for playback of the media. If a URL is supplied,

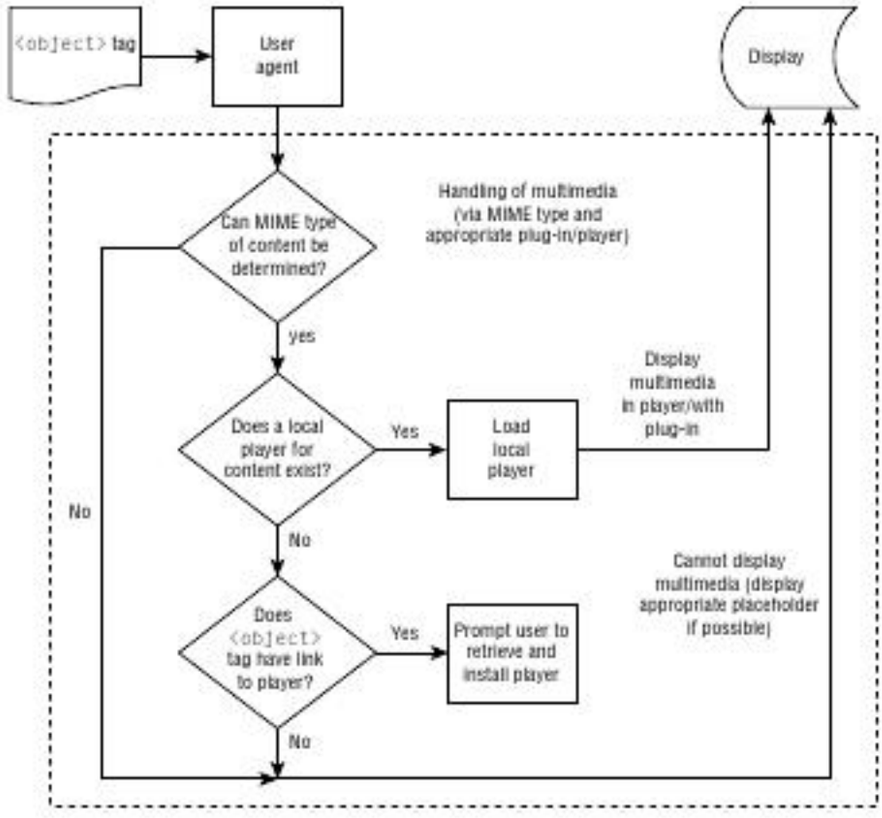
then the user is prompted to download the player. If the user responds in the affirmative, the player is downloaded and used to play/display the media.

5. If for any reason this process fails — invalid or missing MIME types, user refused to download player, and so on — the user agent will generally display a placeholder (as it does for missing or broken graphic links) and will not play the media.

For example, if you haven't installed the Flash plug-in and you load a page with Flash content, you can install the Flash plug-in quickly by following the prompts. However, not every plug-in works as well as Flash.

FIGURE 13-3

The process a user agent goes through when unknown content is encountered.



Tip

More information on inserting objects into your document can be found in the HTML standards at: `www.w3.org/TR/html401/struct/objects.html`. ■

Popular formats and players (plug-ins)

Four animation and video players dominate the Web space today. Windows Media Player's capabilities have also been enhanced over the years to enable it to play a lot of content that would

otherwise have taken a dedicated player. The following sections detail the four dominant multimedia programs used for the Web.

Flash

Flash, which has become arguably the most prevalent multimedia format, began life as a plug-in for something called FuturePlayer. FuturePlayer was purchased by Macromedia, which made significant refinements to the original product. Macromedia had already enjoyed reasonable success with its own Shockwave format, which was quite similar to Flash files but was generated by Macromedia Director. Macromedia did a good job of commingling the two formats, and eventually Shockwave pretty much disappeared in favor of Flash. Today's Flash can display MP3-based video and sound along with vector graphics, and can harness data sources from relational databases and XML.

In fact, Flash has become a serious application platform in its own right, enabling developers to display changing data in real time.

RealOne

RealOne is a media player that reads video and audio files. Real, Inc., the developer of RealOne, was one of the first companies to introduce the concept of streaming audio to desktops. Streaming media (audio and video) is sent in real time through special servers. If you're doing professional-level streaming media, you'll want to check whether your host provider (if you're using one) offers access to a Real Audio server. If you're planning on developing for RealOne, you can find comprehensive software development kits (SDKs) and tutorials at `www.realnetworks.com/support/index.html`.

QuickTime

QuickTime has distinguished itself by consistently raising the bar on video quality. QuickTime has long been a staple in the Apple world, but its quality is so good it has made inroads into the Wintel world too.

YouTube

More of a platform than format or player, YouTube has become the most popular way to embed video in documents. Using Flash, YouTube provides a stable and known platform for displaying and controlling video. Using YouTube to embed video is covered later in this chapter.

Windows Media Player

Windows Media Player has a huge installed base because it is included as part of the Windows operating system. Its functionality is virtually identical to RealOne, offering video and

music playing capabilities. To properly display Windows Media Player files, you should use the ASX markup language, which is an XML-based proprietary language developed by Microsoft.

When a user clicks an ASX link, the browser spawns an instance of the Windows Media Player. Consider the following link:

```
<a href="http://webserver/path/yourfile.asx">Link to Streaming Content</a>
```

This links to the following file and opens up a Media Player:

```
<asx version = "3.0">
<title> ASX Demo</title>
  <entry>
   <title>Gopher Hunt Theme Song</title>
   <author>Rodent Studios</author>
    <copyright>(c)2009 Rodent studios</copyright>
    <ref href="mms://windowsmediaserver/path/mysong.asf" />
  </entry>
</asx>
```

For the specifics of what the various elements mean in an ASX file, go to `http://msdn2.microsoft.com/en-us/library/ms910265.aspx`.

Embedding Media via the Object Tag

The best way to embed multimedia in your documents is via the object tag (`<object>`). This tag embeds a link to media in your document and specifies what a user agent needs in order to play the media (plug-in/player).

An example of an object tag follows; it is configured to embed a Flash file named `myFlashMovie.swf`:

```
<object
 classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase=
"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#
 version=9,0,45,0"
 width="550" height="400" id="myMovieName">
   <param name=movie value="myFlashMovie.swf" />
   <param name=quality value=high />
   <param name=bgcolor value=#FFFFFF />
 </object>
```

The object tag includes a wealth of information for the user agent and plug-in alike. The `classid` and `codebase` attributes specify information about which plug-in is necessary to play the content — the `codebase` attribute includes the player/plug-in necessary to play the content, including the specific (or usually minimum) version of the plug-in.

Where do you get the appropriate information for the `classsid` and `codebase` attributes? Usually from the vendor's website. For example, the following page from Adobe's TechNote site outlines all the attributes for the `<object>` and `<embed>` tags (although the version code is a bit dated): `www.adobe.com/go/tn_12701`.

Note

Embedding media in your document via `<object>` or `<embed>` tags causes the media to display within your document alongside other HTML objects. The other method of including media is simply linking to it via a standard anchor. For example, to link to a media file — causing it to display exclusively in the user agent window — you could use a tag similar to this:

```
<a href="myFlashFile.swf">A Flash File</a>
```

When the user clicks the link, the Flash file (`myFlashFile.swf`) will load in the user agent window and play according to the player's defaults. This technique can be used for most media formats that can be played by the players typically installed on an end user's machine (`wav`, `mpg`, `avi`, `aiff`, and so on).

However, this method relies upon the user agent finding a suitable player based solely on the extension (`wav`, `mpg`, `avi`, `aiff`, and so on) of the media file. To help the user agent, consider adding a `type` attribute that directly specifies the media type, similar to the following:

```
<a href="myFlashFile.swf"
type="application/x-shockwave-flash">
A Flash File</a> ■
```

Table 13-1 shows the other attributes that you can use with the object tag. The specific attributes used depend on your specific needs and the plug-in being used.

For extra functionality and customization of the user agent's handling of the media, you can embed parameter tags (`<param>`) within the object tag. The parameter tags have the following format:

```
<param name="name_of_parameter" value="value_of_parameter" />
```

These tags are used to set the value of parameters that the plug-in/player understands. For example, the Flash Player supports a parameter named `loop`, which controls whether the movie will play only once or multiple times. To set the Flash animation to loop, you would include the following parameter tag:

```
<param name="loop" value="true" />
```

TABLE 13-1

Attributes of the Object Element

Attribute Name	HTML Standard and Description
`archive` (optional)	(HTML 4.01) A space-separated list of URIs for archives of classes and resources to be preloaded. Using this attribute can significantly improve the loading speed of an object.
`classid` (optional)	(HTML 4.01) Specifies the location of the object's implementation by URI. Depending on the type of object involved, it can be used with, or as an alternative to, the `data` attribute.
`codebase` (optional)	(HTML 4.01) Indicates the base URI for the path to the object file. The default is the same base URI as the document.
`codetype` (recommended)	(HTML 4.01) Specifies the content type of data expected. If this is omitted, the default is the same as the `type` attribute.
`data` (optional)	(HTML 4.01) Specifies the location of the object's data. If given as a relative URI, it is relative to the code-based URI.
`height` (optional)	(HTML 4.01) Specifies the initial height in pixels or percentages of the element.
`hspace` (optional)	(HTML 4.01) Defines the number of pixels on the horizontal sides of the element.
`id` (optional)	(HTML 4.01) (CSS enabled) Formats the contents of the tag according to the style ID. Note: IDs must be unique within a document.
`name` (optional)	(HTML 4.01) The `name` attribute assigns the control name to the element.
`standby` (optional)	(HTML 4.01) This specifies a message that is shown to a user while the object is loading.
`style` (optional)	(HTML 4.01) (CSS enabled) Formats the contents of the element according to the listed style.
`type` (recommended)	(HTML 4.01) Indicates the content type at the link target. Specify the type as a MIME-type. This attribute is case insensitive.
`vspace` (optional)	(HTML 4.01) Defines the number of pixels on the vertical sides of the element.
`width` (optional)	(HTML 4.01) Specifies the initial width in pixels or a percentage of the element.

When the <object> Tag Is Not Supported

A handful of user agents do not support the relatively new <object> tag. These user agents typically *do* support an older embedded media tag, embed (<embed>). This deprecated tag has the following syntax when embedding a Flash file named myFlashMovie.swf:

```
<embed src="myFlashMovie.swf"
  quality="high" bgcolor="#FFFFFF" width="550" height="400"
  name="myMovieName" align="" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer">
</embed>
```

The embed tag contains similar information to that in the <object> tag and its <param> tags, but in attribute format — that is, the options are embedded as attributes within the <embed> tag, not as children tags. As with the <object> tag, <embed> supports a variety of attributes and values, depending on the desired use and player/plug-in being utilized. The preceding example uses the attributes listed in the accompanying table.

Attribute	Definition
src	URL to the media file
quality (optional)	Quality setting at which the movie should be played (not supported by all plug-ins)
bgcolor (optional, not supported with all players/plug-ins)	Background color that should be used for the media
width (optional, but should be included)	Width of the media file
height (optional, but should be included)	Height of the media file
name	Name used to identify the particular embed tag
type	MIME type of the media file
pluginspage	URL to a page to download the appropriate player/plug-in

Because the <embed> tag has been deprecated, direct use of this tag will cause your documents not to validate. However, you can use the following technique to include both an <object> tag and an <embed> tag to support a wider range of user agents. Simply encase the <embed> tag within the <object> tag as shown:

```
<object
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase=
"http://download.macromedia.com/pub/shockwave/cabs/flash/
swflash.cab#version=9,0,45,0"
width="550" height="400" id="myMovieName">
```

```
    <param name-movie value-"myFlashMovie.swf" />
    <param name-quality value-high />
    <param name-bgcolor value-#FFFFFF />
    <embed src-"myFlashMovie.swf"
      quality-"high" bgcolor-"#FFFFFF" width-"550" height-"400"
      name-"myMovieName" align-""
      type-"application/x-shockwave-flash"
      pluginspage-"http://www.macromedia.com/go/getflashplayer">
    </embed>
  </object>
```

This method effectively hides the <embed> tag from validation programs — they do not check the children tags embedded within the <object> tag. Older user agents still find and use the <embed> tag, whereas modern user agents will use the <object> tag instead, ignoring the <embed> tag.

Embedding a Windows Media Player Using <object>

Earlier in this chapter, I discussed using ASX for delivering media via Windows Media Player. That method spans an instance of the player to play the media. Using the <object> tag, you can also embed the player in your Web pages.

Caution

To work, this method relies on your entire target audience running Windows, as it utilizes Windows ActiveX controls to integrate with Windows Media Player. No Windows, no ActiveX controls, no interface into Windows Media Player. In addition, if a user agent doesn't fully support ActiveX, it won't be able to take full advantage of the features described in this section. In other words, if your audience isn't running Internet Explorer, you can't bank on the full feature set of the embedded Media Player video.

It's generally more useful to port your media into a Flash-friendly format and wrap it with one of many Flash projector scripts, if necessary, to give it appropriate playback controls. ■

The <object> tag format should be familiar by this point. The following is a sample of a tag supporting an embedded Windows Media Player instance:

```
<object id-"video" width-"320" height-"240"
   classid-"CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
   codebase-"http://activex.microsoft.com/activex/controls/
```

```
mplayer/en/nsmp2inf.cab#Version=6,4,5,715"
  standby="Loading Microsoft Windows Media Player components..."
  type="application/x-oleobject">
  <param name="URL" value="GamePLaySample.avi" />
  <param name="SendPlayStateChangeEvents" value="True" />
  <param name="AutoStart" value="True" />
  <param name="uiMode" value="mini" />
  <param name="Volume" value="20" />
  <param name="CurrentPosition" value="0" />
  <param name="PlayCount" value="1" />
  <embed type="application/x-mplayer2"
    pluginspage=
    "http://www.microsoft.com/Windows/Downloads/Contents/MediaPlayer/"
    width="320" height="240"
    src="GamePlaySample.avi"
    autostart="True"
    showcontrols="True" showstatusbar="False"
    showdisplay="False" autorewind="True">
  </embed>
</object>
```

Note that the `codebase` line is broken into two lines in this listing because of the limitations of the printed page. In your code, it should be on one line.

The `codebase` and `classid` shown in the preceding listing are specific to Windows Media Player 6, a stable player that is a common denominator in older user agents. Additional pairs of parameters for other versions and other useful information can be found on the following reference site: `www.jakeludington.com/project_studio/20051015_embedding_windows_media_player_wma.html`.

The Windows Media Player object takes a wide variety of parameters, some of which appear in the preceding listing, but most are shown in Table 13-2.

Note

Many additional parameters and control settings are available for Windows Media Player, but not all of them are necessarily suitable for delivering media over the Web. Visit Microsoft's Windows Media Developer Center site for more information about Windows Media Player settings and controls: `http://msdn2.microsoft.com/en-us/windowsmedia/default.aspx`. ■

Figure 13-4 shows examples of each of the four user interface settings. Note that the user interface, if enabled, occupies a certain amount of the object's area. This amount is determined largely by the skin being used by Windows Media Player on the user's machine. However, if you intend to enable the interface to any degree, you should compensate by adjusting the object's height to avoid compacting the media.

TABLE 13-2

Parameters for a Windows Media Player Object

Parameter	Value Type	Use/Effect
AutoStart	True/False	Controls whether the media will begin playing automatically upon loading
BaseURL	URL	Specifies the base URL to be used for any scripting commands embedded in the media
CaptioningID	HTML element ID	The ID of the element supplying alternate captioning for the media, if any
CurrentMarker	Numeric	Specifies the marker number where the media should begin playing. Specifying 0 starts the media from the beginning.
CurrentPosition	Numeric (seconds)	Specifies the timecode (in seconds) where the media should begin playing. Specifying 0 starts the media from the beginning.
EnableContextMenu	True/False	Controls whether the right-click context menu will be available to the user to aid in controlling playback, etc.
Enabled	True/False	Controls whether the Windows Media Player controls are enabled at all
FullScreen	True/False	Controls whether playback occurs in full-screen mode
Mute	True/False	Controls whether the volume is muted on initial playback
Playcount	Numeric	Specifies how many times the media will play. The minimum value for this parameter is 1.
Rate	Numeric	The relative rate of speed at which the media will play. The value specified is multiplied by 1, so specifying .5 will result in half-speed, while specifying 2 results in double-speed.
StretchToFit	True/False	Specifies whether the media should be stretched to fit the player window, if/when the player window is larger than the media size
UIMode	invisible, none, mini, full	Specifies the type and size of the player controls to display with the player. See Figure 13-4 for an example of each.

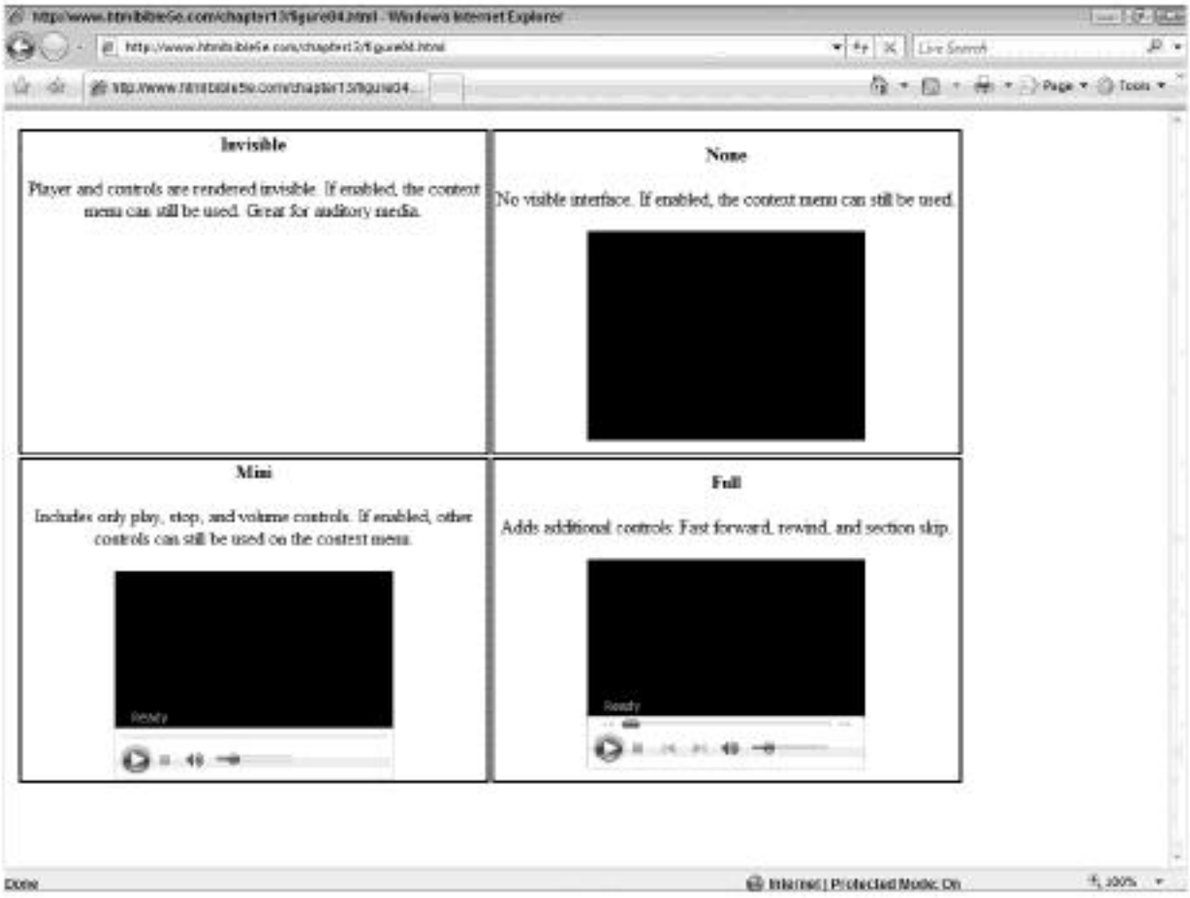
continued

TABLE 13-2 *(continued)*

Parameter	Value Type	Use/Effect
URL	URL	The location of the media file to play
Volume	Numeric (percentage)	The initial setting of the player's volume. The default value is the last setting used by the local user. Values specified with this parameter are treated as a percentage from 0 (no volume) to 100 (full volume).

FIGURE 13-4

The various UIMode settings control how much of the Windows Media Player interface is displayed for the user to control playback of the media.

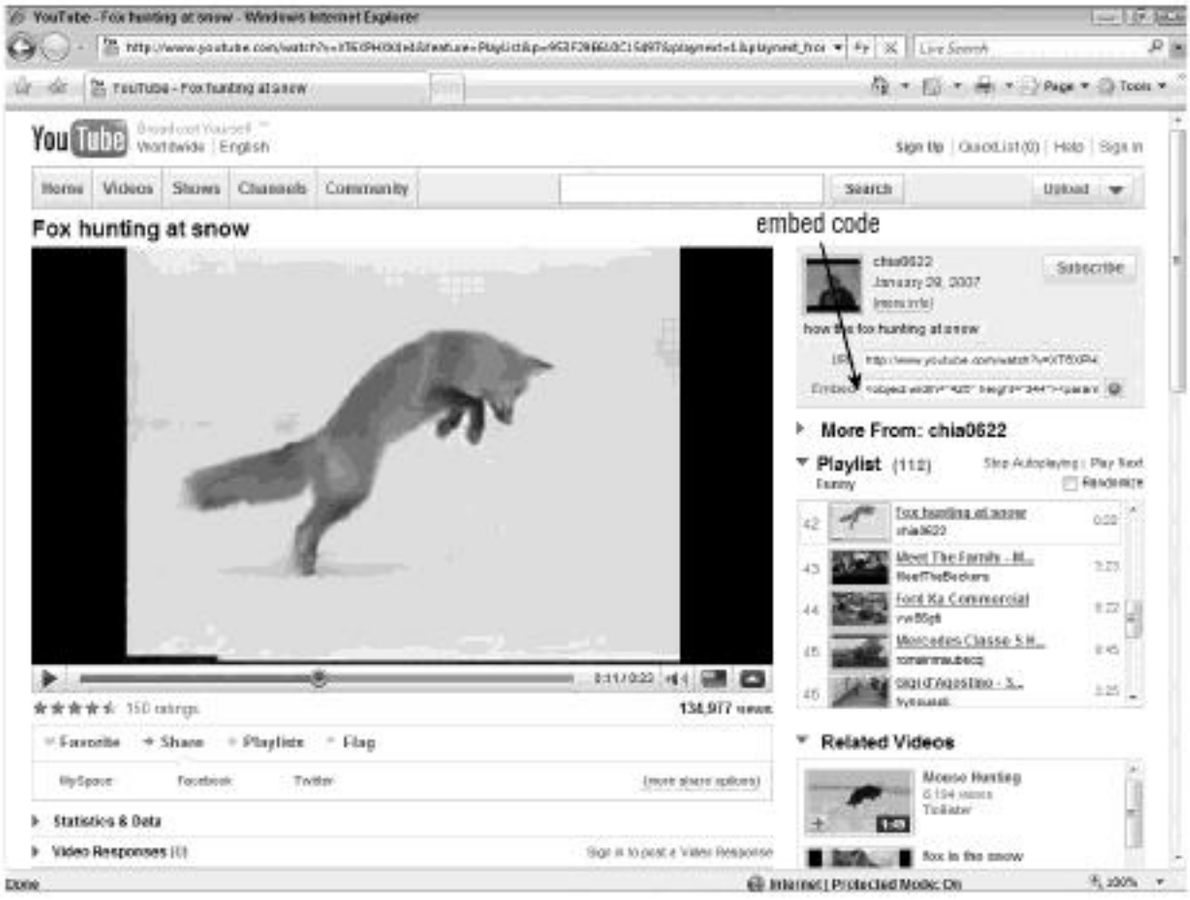


Embedding YouTube Videos

One popular and easy way to add movies to your documents is to use YouTube as the medium. Adding a YouTube video to your documents is very easy thanks to YouTube's embed display, which accompanies all videos on their site. A sample video with the embed code is shown in Figure 13-5.

FIGURE 13-5

Every video on YouTube has accompanying code to embed the video in a document.



Note

Keep in mind that YouTube videos require the Flash plug-in, so your audience needs to have the plug-in installed to see the video. ■

The embed code resembles the following:

```
<object width="425" height="344">
<param name="movie"
    value="http://www.youtube.com/v/XT6XPHXK4e4&hl=en&fs=1">
</param>
<param name="allowFullScreen" value="true">
</param>
<param name="allowscriptaccess" value="always"></param>
<embed src="http://www.youtube.com/v/XT6XPHXK4e4&hl=en&fs=1"
type="application/x-shockwave-flash" allowscriptaccess="always"
allowfullscreen="true" width="425" height="344"></embed>
</object>
```

Note

If you use the YouTube embed code in your documents, you should be aware that the documents will no longer validate against HTML 4. That's because the code needs to work on most browsers and as such employs loose HTML 4 code. ■

To embed a YouTube video, follow these steps:

1. Locate the video on YouTube and navigate to its page.
2. Select the code in the embed field and copy it to the clipboard.
3. Paste the code into your document where you want the video to appear.

You can customize the look of the video by clicking the gear next to the embed field and choosing from a handful of options, like border color and size of the video, before copying the embed code to your document.

You can also add parameters to the URL contained in the `<param>` and `<embed>` tags to cause the video to autoplay when the page is loaded or to start at a particular place in the video.

To cause the video to autoplay, add `&autoplay=1` to the end of the URL. For example,

```
www.youtube.com/v/XT6XPHXK4e4&hl=en&fs=1
```

becomes

```
www.youtube.com/v/XT6XPHXK4e4&hl=en&fs=1&autoplay=1
```

Tip

The number used with the autoplay parameter also controls how many times the video will play. For example, using 2 (instead of 1) will cause the video to play two times without user intervention. ■

To cause the video to start at a designated place, use the `&start` parameter, with the number of seconds into the video as an argument. For example, to start the video 10 seconds into it, add `&start=10` to the end of the URL.

Adding Sound to Web Pages

There are two deprecated ways to add sound files to Web pages. Generally speaking, you shouldn't use these methods, but if you need to add a MIDI or WAV file to your page and can't embed either into a player supported by `<object>`, you might find it necessary to use one.

The first method is using the background sound tag (`<bgsound>`). This tag has the following format:

```
<bgsound src="url_to_sound_file" loop="times_to_play" />
```

This tag causes the specified sound file to be delivered to the user agent and played the number of times specified by the `loop` attribute.

The other method is to use the `<embed>` tag as previously described in this chapter.

Note

Remember that the media file delivered to the user agent must be supported on that platform; that is, the target platform must have a player or plug-in to play the file. Also, using the `<bgsource>` tag doesn't give you the chance to specify a player that the user agent can retrieve, if necessary. ■

Creating Multimedia Files

Creating multimedia files takes dedicated applications geared to create the appropriate file for the format you want to use. For example, creating Flash files requires the Adobe Flash application. QuickTime file production requires a suitable video encoder for MPEG or other QuickTime formats. RealOne is one of the few formats that requires a bit more work and dedicated servers to deliver the media files.

Unfortunately, most of the creation programs are fairly expensive. There are less capable video creation tools on the market, most of which are sold to translate and edit video from personal camcorders and the like. While these programs can create video that can be used on the Web, the file sizes are usually too large to be practical. If you want to develop directly for the Web, look for a professional program such as Adobe Premier to do your video editing and creation.

Note

Many files output their content in Flash format and/or can be used to create original Flash content. If you already have a 3D or animation program, check its export options for Flash format. In addition, search the utility sites for Flash-capable creation tools; more are cropping up every day. ■

A Final Word About Multimedia

Keep in mind that most multimedia is not natively supported by user agents — specific plug-ins or operating system players are required to play a multimedia file delivered over the Web. When a user's system has the correct player installed, the embedded or otherwise delivered multimedia can be played rather seamlessly. However, if users don't have the right player software installed, you are asking them to go through one more step to view your content — a step they might not want to take.

Although the `<object>` tag can be used to deliver almost any content *and* a link to an appropriate player, consider sticking to the most popular players — Flash, QuickTime, RealOne, and Windows Media Player — and porting your content to formats supported by those players.

The best route today, in this author's opinion, is to encapsulate your multimedia content in Flash files whenever possible. Most user agents will already have the Flash Player installed and can instantly play your content.

Also, avoid using anchor tags to link to the media file(s). Use the `<object>` and `<embed>` tags to embed the media in a properly validated HTML document.

Last, remember that the best kind of multimedia to use on your site might be no multimedia at all, which saves the user the aggravation of encountering unwanted multimedia, downloading unwanted plug-ins, and so on.

Summary

This chapter showed how easy it is to embed multimedia into your Web documents, and what concerns you need to be aware of in doing so. You saw how using the `<object>` tag is easy and flexible and that with it you can deliver just about any content. You also learned about other tags that have been deprecated but can still be used in a pinch.

The next two chapters wrap up the HTML coverage with special characters and internationalization. The final two chapters in this section (Chapter 16 and 17) cover scripting.

CHAPTER 14

Special Characters

As its roots are firmly grounded in plain text, HTML needs to be able to display a wide range of characters — many that cannot be typed on a regular keyboard. Language is rich with extended and accented characters, and there are many reserved characters in HTML.

The HTML specification defines many entities — specific codes — to insert special characters. This chapter introduces you to the concept of entities and lists the various entities available for use.

Note
The W3C website is a good source of information about entities. The HTML entities are listed at `www.w3.org/TR/html4/sgml/entities.html`. ■

IN THIS CHAPTER

Understanding Character Encodings

Character encoding at its simplest is the method that maps binary data to its proper character equivalents. For example, in a standard American English document character, 65 is matched to a capital A.

Most English fonts follow the American Standard Code for Information Interchange (ASCII) coding. So when Web designers insert a capital A, they can be assured that users will see the appropriate "A" in their user agent.

There are, of course, plenty of caveats to that statement. The document must be encoded as English, the specified font must also be encoded as English, the font must be an alphanumeric font capable of producing the letter, and the user agent must not interfere with either encoding.

Note

Document encoding is typically passed to the user agent in the `Content-Type` HTTP header, such as the following:

```
Content-Type: text/html; charset=EN-US
```

However, some user agents don't correctly handle encoding in the HTTP header. If you need to explicitly declare a document's encoding, you should use an appropriate meta tag in your document, similar to the following:

```
<meta http-equiv="Content-Type" content="text/html; charset=EN-US"> ■
```

What happens when any of the necessary pieces are different or changed from what they were intended to be? For example, what if your document is viewed in Japan, where the requisite user agent font is in Japanese instead of English? In those cases, using the proper document encoding helps ensure that the right characters are represented.

Most fonts have international characters encoded in them as well as their native language character set. When a non-native encoding is specified, the user agent tries to use the appropriate characters in the appropriate font. If appropriate characters cannot be found in the current font, then alternative fonts can be used.

However, none of this can be accomplished if the document does not declare its encoding. Without knowing the document encoding, the user agent simply uses the character that corresponds to the character position arriving in the data stream. For example, a capital A is translated to whatever character is in the 65th place in the font table the user agent is using.

Cross-Ref

More information on character encoding and internationalizing documents can be found in Chapter 15. ■

Special Characters

Several characters have special meaning in HTML. For example, the less than symbol (<) signals the beginning of a tag. As such, you cannot use that character in normal text. Instead, you must encode the character using a means that user agents understand. Such codes are referred to as *entities*. When the user agent renders a document and encounters an entity, the entity is rendered as the correct character.

Entities in HTML begin with an ampersand (&), end with a semicolon (;), and contain a numeric code or mnemonic phrase in between.

Numerically coded entities can use decimal or hexadecimal numbers. Either must be preceded by a pound sign (#). Hexadecimal numbers also need to be preceded by an x. For example, a nonbreaking space is character number 160. The following entity in decimal references this character:

```
 
```

The following entity in hexadecimal also references character 160:

```
 
```

Mnemonic entities use a few characters to specify the entity — the characters usually are an abbreviation or mnemonic representation of the character they represent. For example, the following entity represents a nonbreaking space:

```
 
```

Other essential entities are listed in Table 14-1.

TABLE 14-1

Essential Entities

Decimal Entity	Mnemonic Entity	Character
&#34;	&quot;	Double quote mark
&#38;	&amp;	Ampersand
&#60;	&lt;	Less than symbol
&#62;	&gt;	Greater than symbol
		Nonbreaking space

Additional special-use characters are covered in the following sections.

Note

Specifying special characters via their mnemonic codes is the only way to ensure that the correct character is used by the user agent. Using the decimal or hexadecimal equivalent only instructs the user agent to use the corresponding character at that place in the current font map — it doesn't guarantee that the character is the one you intended. However, the user agent will use the character mapped to the mnemonic, despite its position in the font map.

All that said, remember that entity use is not a substitution for correctly internationalizing your documents. Refer to Chapter 15 for more information. ■

En and Em Spaces and Dashes

The en space and dash characters got their name from their relative size; they are as wide as a capital N. Likewise, em characters are as wide as a capital M.

These characters have specific uses in the English language:

- *En spaces* are used when you need a larger space than a normal space provides. For example, en spaces can be used between street numbers and street names for clarity (e.g., 123 Main Street).

- *Em spaces* are used to separate elements such as dates and headlines, figure numbers and captions, and so on (e.g., Figure 02 A simple prompt).
- *En dashes* are used instead of hyphens in constructs such as phone numbers, element numbering, and so on.
- *Em dashes* are used grammatically when you need to divide thoughts in a sentence (e.g., "The excuse was nonsense — at least that's how it seemed to me.").

Table 14-2 lists the entities for en and em elements.

TABLE 14-2

En and Em Entities

Decimal Entity	Mnemonic Entity	Character
		En space
		Em space
&#8211;	&ndash;	En dash
&#8212;	&mdash;	Em dash

Copyright and Trademark Symbols

Copyright and trademark symbols are special symbols that indicate a legal relationship between individuals (or companies) and text.

The copyright symbol (©) is used to indicate that someone has asserted certain rights on written material — text included with the symbol usually indicates which rights. For example, many written works include the following phrase as a copyright: "Copyright © 2009. All rights reserved."

The trademark and registered marks (™ and ®) are used to indicate that a particular word or phrase is trademarked — that is, marked (trademarked) or registered for unique use by an individual or company. For example, "Windows" is a registered trademark of Microsoft, and "For Dummies" is a registered trademark of Wiley Publishing.

Note

Trademark and registered trademark symbols are typically superscripted after the word or phrase to which they apply. As such, you should generally use each within superscript (<sup>) tags. ■

Table 14-3 lists the entities for copyright, trademark, and registered trademark symbols.

Note that some fonts include the trademark symbol. However, because the symbol is actually two characters, it is included as an exception, not a rule. As such, you shouldn't rely on an entity to display the symbol, but use specific small and superscript font coding such as the following:

```
<small><sup>TM</sup></small>
```

TABLE 14-3

Copyright, Trademark, and Registered Entities

Decimal Entity	Mnemonic Entity	Character
&#169;	&copy;	Copyright symbol
&#8482;	&trade;	Trademark symbol
&#174;	&reg;	Registered trademark symbol

Note
Use of styles is preferred over the use of the <small> tag. ■

Currency Symbols

There are many currency symbols, including the U.S. dollar ($), the English pound (£), the European euro (€), and the Japanese yen (¥). There is also the general currency symbol (¤). Table 14-4 lists many of the most common currency symbols.

TABLE 14-4

Currency Entities

Decimal Entity	Mnemonic Entity	Character
&#162;	&cent;	The cent symbol (¢)
&#163;	&pound;	English pound
&#164;	&curren;	General currency
&#165;	&yen;	Japanese yen
&#8364;	&euro;	European euro

Note that the dollar symbol ($) is typically ASCII character 24 (in U.S. fonts) and can be accessed directly from the keyboard.

"Real" Quotation Marks

Real quotation marks, used in publishing, do not exist on a standard keyboard. The quote marks available on the keyboard (" and ') are straight quotes; that is, they are small, superscripted, vertical lines.

Quote marks used in publishing typically resemble the numbers 6 and 9 — that is, dots with a serif leading off of them. For example, the following sentence is set off with real quote marks:

"This sentence is a real quote."

Table 14-5 lists the entities for real quotes.

TABLE 14-5

Quote Mark and Apostrophe Entities

Decimal Entity	Mnemonic Entity	Character
&#8216;	&lsquo;	Left/Opening single quote
&#8217;	&rsquo;	Right/Closing single quote and apostrophe
&#8220;	&ldquo;	Left/Opening double quote
&#8221;	&rdquo;	Right/Closing double quote

Arrows

A variety of arrow symbols are available as entities. Table 14-6 lists these entities.

TABLE 14-6

Arrow Entities

Decimal Entity	Mnemonic Entity	Character
&#8592;	&larr;	Left arrow
&#8593;	&uarr;	Up arrow
&#8594;	&rarr;	Right arrow
&#8595;	&darr;	Down arrow
&#8596;	&harr;	Left right arrow
&#8629;	&crarr;	Down arrow with corner leftwards
&#8656;	&lArr;	Leftwards double arrow
&#8657;	&uArr;	Upwards double arrow
&#8658;	&rArr;	Rightwards double arrow
&#8659;	&dArr;	Downwards double arrow
&#8660;	&hArr;	Left right double arrow

Accented Characters

Many accented character entities are available in the HTML standard. These characters can be used in words such as résumé. Table 14-7 lists the accented character entities.

TABLE 14-7

Accented Character Entities

Decimal Entity	Mnemonic Entity	Character
&#192;	&Agrave;	Latin capital letter A with grave
&#193;	&Aacute;	Latin capital letter A with acute
&#194;	&Acirc;	Latin capital letter A with circumflex
&#195;	&Atilde;	Latin capital letter A with tilde
&#196;	&Auml;	Latin capital letter A with diaeresis
&#197;	&Aring;	Latin capital letter A with ring above
&#198;	&AElig;	Latin capital letter AE
&#199;	&Ccedil;	Latin capital letter C with cedilla
&#200;	&Egrave;	Latin capital letter E with grave
&#201;	&Eacute;	Latin capital letter E with acute
&#202;	&Ecirc;	Latin capital letter E with circumflex
&#203;	&Euml;	Latin capital letter E with diaeresis
&#204;	&Igrave;	Latin capital letter I with grave
&#205;	&Iacute;	Latin capital letter I with acute
&#206;	&Icirc;	Latin capital letter I with circumflex
&#207;	&Iuml;	Latin capital letter I with diaeresis
&#208;	&ETH;	Latin capital letter ETH
&#209;	&Ntilde;	Latin capital letter N with tilde
&#210;	&Ograve;	Latin capital letter O with grave
&#211;	&Oacute;	Latin capital letter O with acute
&#212;	&Ocirc;	Latin capital letter O with circumflex
&#213;	&Otilde;	Latin capital letter O with tilde
&#214;	&Ouml;	Latin capital letter O with diaeresis
&#216;	&Oslash;	Latin capital letter O with stroke

continued

TABLE 14-7 *(continued)*

Decimal Entity	Mnemonic Entity	Character
&#217;	&Ugrave;	Latin capital letter U with grave
&#218;	&Uacute;	Latin capital letter U with acute
&#219;	&Ucirc;	Latin capital letter U with circumflex
&#220;	&Uuml;	Latin capital letter U with diaeresis
&#221;	&Yacute;	Latin capital letter Y with acute
&#222;	&THORN;	Latin capital letter THORN
&#223;	&szlig;	Latin small letter sharp s = ess-zed
&#224;	&agrave;	Latin small letter a with grave
&#225;	&aacute;	Latin small letter a with acute
&#226;	&acirc;	Latin small letter a with circumflex
&#227;	&atilde;	Latin small letter a with tilde
&#228;	&auml;	Latin small letter a with diaeresis
&#229;	&aring;	Latin small letter a with ring above
&#230;	&aelig;	Latin small letter ae
&#231;	&ccedil;	Latin small letter c with cedilla
&#232;	&egrave;	Latin small letter e with grave
&#233;	&eacute;	Latin small letter e with acute
&#234;	&ecirc;	Latin small letter e with circumflex
&#235;	&euml;	Latin small letter e with diaeresis
&#236;	&igrave;	Latin small letter i with grave
&#237;	&iacute;	Latin small letter i with acute
&#238;	&icirc;	Latin small letter i with circumflex
&#239;	&iuml;	Latin small letter i with diaeresis
&#240;	&eth;	Latin small letter eth
&#241;	&ntilde;	Latin small letter n with tilde
&#242;	&ograve;	Latin small letter o with grave
&#243;	&oacute;	Latin small letter o with acute
&#244;	&ocirc;	Latin small letter o with circumflex
&#245;	&otilde;	Latin small letter o with tilde
&#246;	&ouml;	Latin small letter o with diaeresis

Decimal Entity	Mnemonic Entity	Character
&#248;	&oslash;	Latin small letter o with stroke
&#249;	&ugrave;	Latin small letter u with grave
&#250;	&uacute;	Latin small letter u with acute
&#251;	&ucirc;	Latin small letter u with circumflex
&#252;	&uuml;	Latin small letter u with diaeresis
&#253;	&yacute;	Latin small letter y with acute
&#254;	&thorn;	Latin small letter thorn
&#255;	&yuml;	Latin small letter y with diaeresis

Greek and Mathematical Characters

Table 14-8 lists various Greek symbol entities available in HTML.

TABLE 14-8

Greek Symbol Entities

Decimal Entity	Mnemonic Entity	Character
&#913;	&Alpha;	Greek capital letter alpha
&#914;	&Beta;	Greek capital letter beta
&#915;	&Gamma;	Greek capital letter gamma
&#916;	&Delta;	Greek capital letter delta
&#917;	&Epsilon;	Greek capital letter epsilon
&#918;	&Zeta;	Greek capital letter zeta
&#919;	&Eta;	Greek capital letter eta
&#920;	&Theta;	Greek capital letter theta
&#921;	&Iota;	Greek capital letter iota
&#922;	&Kappa;	Greek capital letter kappa
&#923;	&Lambda;	Greek capital letter lambda
&#924;	&Mu;	Greek capital letter mu

continued

TABLE 14-8 *(continued)*

Decimal Entity	Mnemonic Entity	Character
&#925;	&Nu;	Greek capital letter nu
&#926;	&Xi;	Greek capital letter xi
&#927;	&Omicron;	Greek capital letter omicron
&#928;	&Pi;	Greek capital letter pi
&#929;	&Rho;	Greek capital letter rho
&#931;	&Sigma;	Greek capital letter sigma
&#932;	&Tau;	Greek capital letter tau
&#933;	&Upsilon;	Greek capital letter upsilon
&#934;	&Phi;	Greek capital letter phi
&#935;	&Chi;	Greek capital letter chi
&#936;	&Psi;	Greek capital letter psi
&#937;	&Omega;	Greek capital letter omega
&#945;	&alpha;	Greek small letter alpha
&#946;	&beta;	Greek small letter beta
&#947;	&gamma;	Greek small letter gamma
&#948;	&delta;	Greek small letter delta
&#949;	&epsilon;	Greek small letter epsilon
&#950;	&zeta;	Greek small letter zeta
&#951;	&eta;	Greek small letter eta
&#952;	&theta;	Greek small letter theta
&#953;	&iota;	Greek small letter iota
&#954;	&kappa;	Greek small letter kappa
&#955;	&lambda;	Greek small letter lambda
&#956;	&mu;	Greek small letter mu
&#957;	&nu;	Greek small letter nu
&#958;	&xi;	Greek small letter xi
&#959;	&omicron;	Greek small letter omicron
&#960;	&pi;	Greek small letter pi
&#961;	&rho;	Greek small letter rho

Decimal Entity	Mnemonic Entity	Character
&#962;	&sigmaf;	Greek small letter final sigma
&#963;	&sigma;	Greek small letter sigma
&#964;	&tau;	Greek small letter tau
&#965;	&upsilon;	Greek small letter upsilon
&#966;	&phi;	Greek small letter phi
&#967;	&chi;	Greek small letter chi
&#968;	&psi;	Greek small letter psi
&#969;	&omega;	Greek small letter omega
&#977;	&thetasym;	Greek small letter theta symbol
&#978;	&upsih;	Greek upsilon with hook symbol
&#982;	&piv;	Greek pi symbol

Table 14-9 lists a variety of mathematical symbols.

TABLE 14-9

Mathematical Symbol Entities

Decimal Entity	Mnemonic Entity	Character/Symbol
&#215;	&times;	Multiplication sign
&#247;	&division;	Division sign
&#8704;	&forall;	For all
&#8706;	&part;	Partial differential
&#8707;	&exist;	There exists
&#8709;	&empty;	Empty set = null set = diameter
&#8711;	&nabla;	Nabla = backward difference
&#8712;	&isin;	Element of
&#8713;	&notin;	Not an element of
&#8715;	&ni;	Contains as member
&#8719;	&prod;	n-ary product = product sign
&#8721;	&sum;	n-ary summation

continued

TABLE 14-9 *(continued)*

Decimal Entity	Mnemonic Entity	Character/Symbol
&#8722;	&minus;	Minus sign
&#8727;	&lowast;	Asterisk operator
&#8730;	&radic;	Square root = radical sign
&#8733;	&prop;	Proportional to
&#8734;	&infin;	Infinity
&#8736;	&ang;	Angle
&#8743;	&and;	Logical and = wedge
&#8744;	&or;	Logical or = vee
&#8745;	&cap;	Intersection = cap
&#8746;	&cup;	Union = cup
&#8747;	&int;	Integral
&#8756;	&there4;	Therefore
&#8764;	&sim;	Tilde operator = varies with = similar to
&#8773;	&cong;	Approximately equal to
&#8776;	&asymp;	Almost equal to = asymptotic to
&#8800;	&ne;	Not equal to
&#8801;	&equiv;	Identical to
&#8804;	&le;	Less than or equal to
&#8805;	&ge;	Greater than or equal to
&#8834;	&sub;	Subset of
&#8835;	&sup;	Superset of
&#8836;	&nsub;	Not a subset of
&#8838;	&sube;	Subset of or equal to
&#8839;	&supe;	Superset of or equal to
&#8853;	&oplus;	Circled plus = direct sum
&#8855;	&otimes;	Circled times = vector product
&#8869;	&perp;	Up tack = orthogonal to = perpendicular
&#8901;	&sdot;	Dot operator
&#8968;	&lceil;	Left ceiling
&#8969;	&rceil;	Right ceiling

Decimal Entity	Mnemonic Entity	Character/Symbol
&#8970;	&lfloor;	Left floor
&#8971;	&rfloor;	Right floor
&#9001;	&lang;	Left-pointing angle bracket
&#9002;	&rang;	Right-pointing angle bracket

Other Useful Entities

Table 14-10 lists other miscellaneous entities.

TABLE 14-10

Miscellaneous Entities

Decimal Entity	Mnemonic Entity	Character/Symbol
&#161;	&iexcl;	Inverted exclamation mark
&#166;	&brvbar;	Broken bar = broken vertical bar
&#167;	&sect;	Section sign
&#168;	&uml;	Diaeresis = spacing diaeresis
&#170;	&ordf;	Feminine ordinal indicator
&#171;	&laquo;	Left-pointing double angle quotation mark = left pointing guillemet
&#172;	&not;	Not sign
&#173;	&shy;	Soft hyphen = discretionary hyphen
&#175;	&macr;	Macron = spacing macron = overline = APL overbar
&#176;	&deg;	Degree sign
&#177;	&plusmn;	Plus-minus sign = plus-or-minus sign
&#178;	&sup2;	Superscript two = superscript digit two = squared
&#179;	&sup3;	Superscript three = superscript digit three = cubed
&#180;	&acute;	Acute accent = spacing acute
&#181;	&micro;	Micro sign

continued

TABLE 14-10 *(continued)*

Decimal Entity	Mnemonic Entity	Character/Symbol
&#182;	&para;	Pilcrow sign = paragraph sign
&#183;	&middot;	Middle dot = Georgian comma = Greek middle dot
&#184;	&cedil;	Cedilla = spacing cedilla
&#185;	&supl;	Superscript one = superscript digit one
&#186;	&deg;	Masculine ordinal indicator
&#187;	&raquo;	Right-pointing double angle quotation mark = right-pointing guillemet
&#188;	&frac14;	Vulgar fraction one-quarter = fraction one-quarter
&#189;	&frac12;	Vulgar fraction one-half = fraction one-half
&#190;	&frac34;	Vulgar fraction three-quarters = fraction three-quarters
&#191;	&iquest;	Inverted question mark = turned question mark
&#338;	&OElig;	Latin capital ligature OE
&#339;	&oelig;	Latin small ligature oe
&#352;	&Scaron;	Latin capital letter S with caron
&#353;	&scaron;	Latin small letter s with caron
&#376;	&Yuml;	Latin capital letter Y with diaeresis
&#710;	&deg;	Modifier letter circumflex accent
&#732;	&tilde;	Small tilde
		Thin space
&#8204;	&zwnj;	Zero width non-joiner
&#8205;	&zwj;	Zero width joiner
&#8206;	&lrm;	Left-to-right mark
&#8207;	&rlm;	Right-to-left mark
&#8218;	&sbquo;	Single low-9 quotation mark
&#8222;	&bdquo;	Double low-9 quotation mark
&#8224;	&dagger;	Dagger

Decimal Entity	Mnemonic Entity	Character/Symbol
&#8225;	&Dagger;	Double dagger
&#8240;	&permil;	Per mille sign
&#8249;	&lsaquo;	Single left-pointing angle quotation mark
&#8250;	&rsaquo;	Single right-pointing angle quotation mark

Summary

Although most of your Web documents will contain standard characters, there are times when you need accented or special characters as well. Taking character and language encoding into account, you can also fall back on HTML entities to insert these special characters.

The next, and final, HTML chapter in this section covers internationalization (Chapter 15). The next two chapters (16 and 17) cover JavaScript and DHTML, bringing scripting to your now robust knowledge of HTML. Part II covers tools and utilities, and Part III provides in-depth coverage of CSS.

CHAPTER 15

Internationalization and Localization

Even though this book is written in English, chances are good it will be translated into other languages. From a website perspective, if your site is only in English, you may eliminate a huge portion of your potential world audience. This chapter takes a look at some options for improving your documents' access to the world's population.

IN THIS CHAPTER

Note

The terms Internationalization and Localization are often used interchangeably. To Internationalize an online document refers to making it more suitable to a global audience, while Localizing an online document refers to making the document suitable for one particular locale. In this chapter we deal with the latter. ■

Internationalization and Localization

Only a small percentage of the world's population uses English as a first language. If you anticipate a wide range of international visitors to your site, then you should consider localizing your site.

Localization is the process of creating different sites for each language you intend to support. By creating documents in Japanese, Chinese, German, French, and Spanish, for example, you vastly increase the size of your potential audience.

The actual implementation of localization can be very straightforward: create different URL "branches" for your site, one branch for each language. For example, if you need to support English, German, and Japanese, your home page URLs might resemble the following:

```
http://www.on-target-games.com/en/index.htm
http://www.on-target-games.com/de/index.htm
http://www.on-target-games.com/jp/index.htm
```

On your Web server, each URL branch would be linked to document directories that store the appropriate language version. Other sites that link to your site can simply use the proper URL — that is, include "en," "de," or "jp" in the URL pointing to a document on your site — depending on the audience's preferred language. In addition, each version of your home page should include a prominent navigational link at the top of the document to the other language versions of the document, as shown in Figure 15-1.

FIGURE 15-1

Localized sites should include navigational aids for users to reach the other versions of a document.

However, maintaining such an implementation can quickly become overwhelming. When a document changes on the site, the change must be translated and incorporated into each of the mirrored sites as well. This exponential increase in labor is why many sites avoid localization.

Translating Your Web Site

There's much more to localization than simply creating branches on your Web server. To be truly global you need to create documents in the native language of your audience — that means translating your text into those languages. Of course, simply changing the words from one language to another won't always accomplish your goals. Simple, colloquial phrases in English won't always carry the same meaning when translated to other languages.

Tip

You can download free localization software from The RWS Group at: `www.translate.com/technology/tools/index.html`. ■

For this reason, you should try to use native speakers of any language to which you are translating. This can be an expensive proposition, and may not be an option if you run a small site that is only marginally profitable. But if you have already made the decision to localize, it's important to do it right.

You can also use translation services that specialize in such endeavors. Use Google or another search engine to point you to "translation services" and see which options may be right for you. Another, often overlooked, resource is International sales partners — companies that want to sell your products or services in their area of the world. If your company or business has ties to other locales, see if you can take advantage of those resources for your localization needs.

If you have only a small amount of text to translate, you can try using an online translator like Yahoo's Babel Fish (`http://babelfish.yahoo.com/`). Babel Fish enables you to translate blocks of text or entire pages between several different languages. Babel Fish's main purpose is to aid in translating pages and text *from* other languages — a snippet of Japanese to English for an English-speaking person — but in a pinch it can be used to translate from English as well.

Understanding Unicode

Unicode is a standard developed by the Unicode Consortium for processing the world's alphabets in a consistent way. Unicode consists of a number of tables, each containing numerical references to alphanumeric characters. Every character in nearly every written language in the world is represented in Unicode, as shown in Table 15-1.

The Unicode tables are also known as *code pages*. Each code page serves a specific set of languages and has a table of numeric references for each character. Each row in Table 15-1 represents a code page, and each code page consists of several rows of numeric references for the characters in the corresponding language.

Basic Latin (U + 0000–U + 007F)

All nations in America, most European nations, most African nations, as well as Australia and New Zealand use the Latin encoding. In Unicode, the Latin encoding is broken down into different parts, with the most basic called Basic Latin. Only a few languages can be written

entirely with a Basic Latin encoding. You generally need to incorporate additional Latin encodings because Basic Latin consists of only characters between 0 and 7F (hexadecimal).

Tip

All of these Latin encodings are automatically included as part of the UTF-8 encoding. In fact, Unicode-based UTF-8 includes most of the world's written languages. ■

TABLE 15-1

Alphabets Represented in Unicode

Start Code	End Code	Block Name
\u0000	\u007F	Basic Latin
\u0080	\u00FF	Latin-1 Supplement
\u0100	\u017F	Latin Extended-A
\u0180	\u024F	Latin Extended-B
\u0250	\u02AF	IPA Extensions
\u02B0	\u02FF	Spacing Modifier Letters
\u0300	\u036F	Combining Diacritical Marks
\u0370	\u03FF	Greek and Coptic
\u0400	\u04FF	Cyrillic
\u0530	\u058F	Armenian
\u0590	\u05FF	Hebrew
\u0600	\u06FF	Arabic
\u0700	\u074F	Syriac
\u0750	\u077F	Arabic Supplement
\u0780	\u07BF	Thaana
\u07C0	\u07FF	N'Ko
\u0900	\u097F	Devanagari
\u0980	\u09FF	Bengali
\u0A00	\u0A7F	Gurmukhi
\u0A80	\u0AFF	Gujarati
\u0B00	\u0B7F	Oriya
\u0B80	\u0BFF	Tamil
\u0C00	\u0C7F	Telugu

Start Code	End Code	Block Name
\u0C80	\u0CFF	Kannada
\u0D00	\u0D7F	Malayalam
\u0D80	\u0DFF	Sinhala
\u0E00	\u0E7F	Thai
\u0E80	\u0EFF	Lao
\u0F00	\u0FFF	Tibetan
\u1000	\u109F	Myanmar
\u10A0	\u10FF	Georgian
\u1100	\u11FF	Hangul Jamo
\u1200	\u137F	Ethiopic
\u1380	\u139F	Ethiopic Supplement
\u13A0	\u13FF	Cherokee
\u1400	\u167F	Unified Canadian Aboriginal Syllabics
\u1680	\u169F	Ogham
\u16A0	\u16FF	Runic
\u1700	\u171F	Tagalog
\u1720	\u173F	Hanunoo
\u1740	\u175F	Buhid
\u1760	\u177F	Tagbanwa
\u1780	\u17FF	Khmer
\u1800	\u18AF	Mongolian
\u1900	\u194F	Limbu
\u1950	\u197F	Tai Le
\u1980	\u19DF	New Tai Lue
\u19E0	\u19FF	Khmer Symbols
\u1A00	\u1A1F	Buginese
\u1B00	\u1B7F	Balinese
\u1D00	\u1D7f	Phonetic Extensions
\u1D80	\u1DBF	Phonetic Extensions Supplement
\u1DC0	\U1DFF	Combining Diacritical Marks Supplement

continued

TABLE 15-1 *(continued)*

Start Code	End Code	Block Name
\u1E00	\u1EFF	Latin Extended Additional
\u1F00	\u1FFF	Greek Extended
\u2000	\u206F	General Punctuation
\u2070	\u209F	Superscripts and Subscripts
\u20A0	\u20CF	Currency Symbols
\u20D0	\u20FF	Combining Diacritical Marks for Symbols
\u2100	\u214F	Letterlike Symbols
\u2150	\u218F	Number Forms
\u2190	\u21FF	Arrows
\u2200	\u22FF	Mathematical Operators
\u2300	\u23FF	Miscellaneous Technical
\u2400	\u243F	Control Pictures
\u2440	\u245F	Optical Character Recognition
\u2460	\u24FF	Enclosed Alphanumerics
\u2500	\u257F	Box Drawing
\u2580	\u259F	Block Elements
\u25A0	\u25FF	Geometric Shapes
\u2600	\u26FF	Miscellaneous Symbols
\u2700	\u27BF	Dingbats
\u27C0	\u27EF	Miscellaneous Mathematical Symbols-A
\u27F0	\u27FF	Supplemental Arrows-A
\u2800	\u28FF	Braille Patterns
\u2900	\u297F	Supplemental Arrows-B
\u2980	\u29FF	Miscellaneous Mathematical Symbols-B
\u2A00	\u2AFF	Supplemental Mathematical Operators
\u2B00	\u2BFF	Miscellaneous Symbols and Arrows
\u2C00	\u2C5F	Glagolitic
\u2C60	\u2C7F	Latin Extended-C
\u2C80	\u2CFF	Coptic
\u2D00	\u2D2F	Georgian Supplement