PART III

Presentation and Layout

5
CHAPTER

Images

A great Web site isn't just about correct markup. Site organization, navigation, interactivity, content, delivery, and a multitude of other issues affect a user's perception of a site. However, images probably are the most obvious part of a great Web site. Carefully used imagery can add to both the appeal and usability of a Web site. Creation of Web-ready images certainly is beyond the scope of this book, but HTML authors should at minimum be aware of the basics of Web image formats such as GIF and JPEG and know when they are being used appropriately. Although the basic HTML/XHTML syntax of adding images to a page using the **img** element is relatively straightforward, creation of an aesthetically pleasing page is truly more art than it is science. Tools can make Web image creation easier, but readers should be realistic and consider both their own artistic limitations as well as the download constraints of the Web before going overboard with images.

Image Preliminaries

Before discussing image use in HTML/XHTML, it is important to discuss what image formats are supported on the Web today. In general, Web-based images come in two basic flavors: GIF (Graphics Interchange Format), as designated by the .gif extension, and JPEG (Joint Photographic Experts Group), as indicated by the .jpg or .jpeg file extension. A third format, PNG (Portable Network Graphics), as indicated by the .png file extension, is slowly gaining ground as a Web format and is supported fairly well in modern browsers. Table 5-1 details the supported image types found in most browsers. While browsers may support other image types, page authors should use only GIF or JPEG images to ensure that all users can see them.

NOTE *Internet Explorer and many other Windows browsers support the bitmap (BMP) file type popular with Windows users. This format has not been widely adopted on the Web.*

Choosing the correct image for the job is an important part of Web design. In general, GIF images tend to be good for illustrations such as logos or cartoons whereas JPEG images usually are the choice for complex imagery such as photographs. The main concerns for site designers when considering an image format are the size of the file itself and the quality of the reproduction. Table 5-2 provides a concise summary of the qualities of each format.

File Type	File Extension
GIF (Graphics Interchange Format)	.gif
JPEG (Joint Photographic Experts Group)	.jpg or .jpeg
XBM (X Bitmaps)	.xbm
XPM (X Pixelmaps)	.xpm
PNG (Portable Network Graphics)	.png

TABLE 5-1 Selected Web Image File Types

Subsequent sections will explain each of these basic features of the two main image formats in slightly more detail.

GIF Images

GIF is the most widely supported image format on the Web. Originally introduced by CompuServe (and occasionally described as CompuServe GIFs), there are actually two types of GIF: *GIF 87* and *GIF 89a*. Both forms of GIF support 8-bit color (256 colors), use the LZW (Lempel-Ziv-Welch) lossless compression scheme, and generally have the .gif file extension. GIF 89a also supports transparency and animation, both of which will be discussed later in this section. Today, when speaking of GIF images, we always assume the GIF89a format is in use and make no distinction between the formats, regardless of whether or not animation or transparency is actually used in the image.

GIF images use a basic form of compression called *run-length encoding*. This lossless compression works well with large areas of continuous color. Figure 5-1 shows the GIF compression scheme in practice. Notice how the test images with large horizontal continuous areas of color compress a great deal, while those with variation do not. As shown in the demo, simply taking a box filled with lines and rotating it 90 degrees shows how dramatic the compression effect can be. Given GIF's difficulty dealing with variability in images, it is

Format	Compression Scheme	Color Depth Supported	Progressive or Interlaced Rendering	Transparency	Animation
GIF	Lossless (preserves file size for minimal compression of continuous horizontal regions of color)	8-bit (256 colors)	Interlaced	Yes (1 degree)	Yes
JPEG	Lossy (trade image quality for file size)	24-bit (millions of colors)	Progressive	No	No

TABLE 5-2 Web Image Format Overview

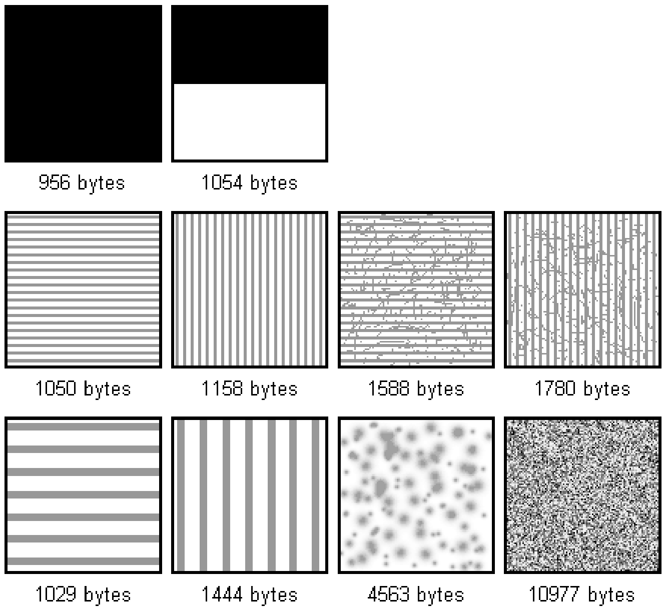
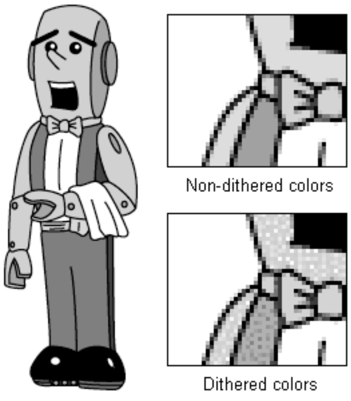


FIGURE 5-1 GIF compression scheme comparison

obvious why the format is good for illustrations and other images that contain large amounts of continuous color.

As mentioned earlier, GIF images only support 8-bit color for a maximum of 256 colors within the image. Consequently, some degree of loss is inevitable when representing true-color images, such as photographs. Typically, when an image is remapped from a large number of colors to a smaller color palette, dithering occurs. The process of dithering attempts to create the desired color that is outside of the palette, by taking two or more colors from the palette and placing them in some sort of checkered or speckled pattern in an attempt to visually create the illusion of the original color.

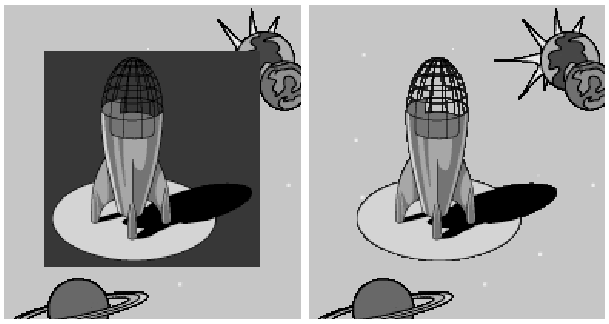


Note *There is a fairly esoteric use of GIF images that allows them to exceed the 256 color barrier by using more than one image block, each with its own color palette within the same GIF file. The so-called "true-color GIF" could provide for thousands of color support, but with a much larger file size. Those looking to exceed the 256 color limitation of GIF should look to JPEG or PNG files.*

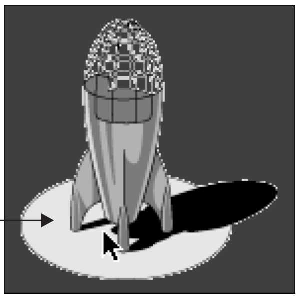
While having only an 8-bit color depth seems problematic, sometimes designers will further downward adjust the bit-depth of GIF files to reduce file size. In general, the higher the bit-depth in an image, the more colors and the greater amount of information required. It would make sense then, that if you can limit the number of colors as much as possible without reducing the quality of the image, you could create some extremely small files. The key to doing this is using just enough colors in the image to support what is there or what is reasonable to dither. Standard 8-bit GIFs will contain up to 256 colors, 7-bit up to 128 colors, 6-bit up to 64 colors, 5-bit up to 32 colors, and so on. Most graphics programs, such as Macromedia Fireworks or Adobe Photoshop with ImageReady, support color reduction directly on image save. Figure 5-2 shows an example of the file reduction possibilities using GIF color reduction.

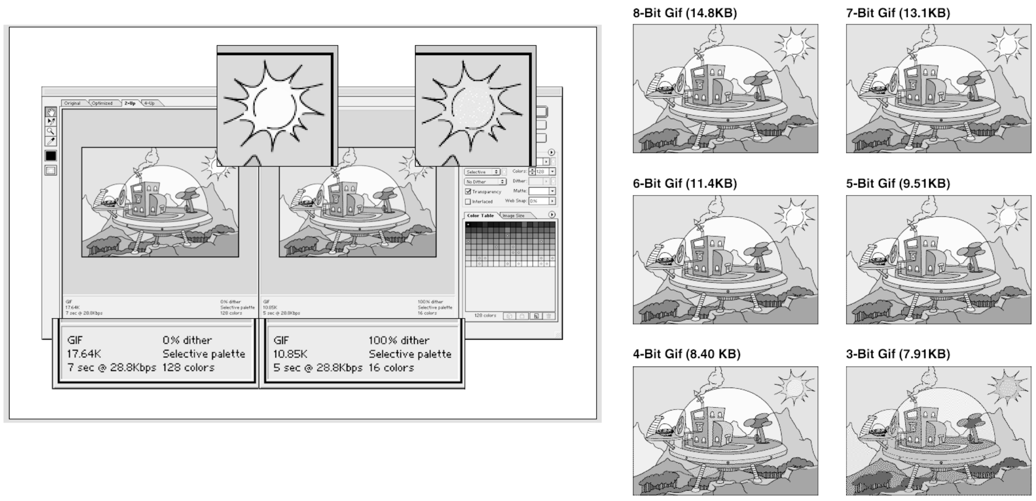
Transparency

GIF images also support *transparency*. One bit of transparency is allowed, which means that one color can be set to be transparent. Transparency allows the background that an image is placed upon to show through, making a variety of complex effects possible.



GIF transparency is far from ideal. Given that only a single color can be made transparent, it can be difficult to avoid a halo effect when placing transparent GIF images on backgrounds, as shown here:

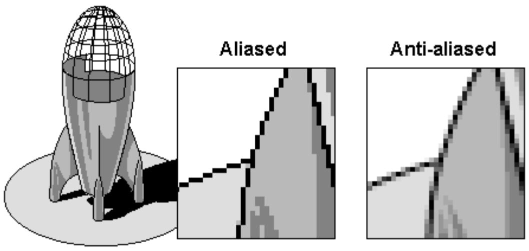



GIF 0% dither
17.64K Selective palette
7 sec @ 28.8Kbps 128 colors

GIF 100% dither
10.85K Selective palette
5 sec @ 28.8Kbps 16 colors

FIGURE 5-2 Color Reduction is useful to reduce GIF file size.

The main problem with 1-bit transparency is that *anti-aliasing* uses variable colors to blur the jagged edges of an image to smooth things out. Recall that everything that is displayed onscreen is made up of pixels and that pixels are square. It should therefore be obvious that creating an image that has rounded edges may pose some problems. Anti-aliasing allows us to create the illusion of rounded or smooth edges by partially filling the edge pixels in an attempt to blend the image into the background, as shown here:



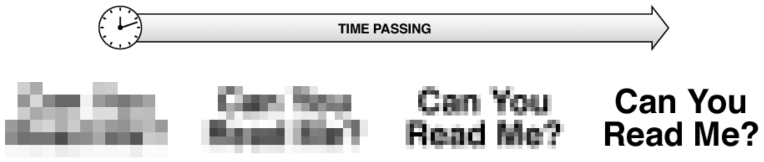
There are a variety of solutions to the anti-aliasing transparency interaction problem. First, you could simply not anti-alias the image, but this can produce unwanted "jagginess" in the image. A second possibility might be to avoid setting the transparency image on a complex background, and instead prefill the image with the appropriate background. This approach is seamless and completely avoids any trace of a halo, but it limits what we can put images on top of. For this reason, designers often avoid transparency in conjunction with complex backgrounds where this effect might be difficult to accomplish.

TIP *When using small text in a graphic, it is often a good idea to leave the text aliased. Anti-aliasing introduces an element of fuzziness, which may make smaller font sizes very difficult to read.*

Interlacing

GIF images also support a feature called *interlacing*. Interlacing allows an image to load in a venetian-blind fashion rather than from top to bottom a line at a time. The interlacing effect allows a user to get an idea of what an image looks like before the entire image has downloaded, thus avoiding user frustration as images download. See Figure 5-3 for an example of interlacing.

The previsualization benefit of interlacing is very useful on the Web, where download speed is often an issue. While interlacing a GIF image is generally a good idea, occasionally it comes with a downside; interlaced images may be larger than non-interlaced images. It is a bad idea to use interlacing for images that have text on them because it's impossible for the text to be read easily until the download is complete.



Animation

Finally, the GIF format also supports animation. This works by stacking GIF after GIF to create the animation, in a manner similar to a flip book. The animation extension also allows timing and looping information to be added to the image. Most popular graphics programs, such as Fireworks, support animated GIFs. An example of the interface to control GIF animation in Fireworks is shown in Figure 5-4.

Animated GIFs provide one of the most popular ways to add simple animation to a Web page because nearly every browser supports them. Browsers that do not support the animated GIF format generally display the first frame of the animation in its place. Even though plug-ins or other browser facilities are not required, authors should not rush out to use animation on their pages. Excessive animation can be distracting for the user, and is often inefficient to download.

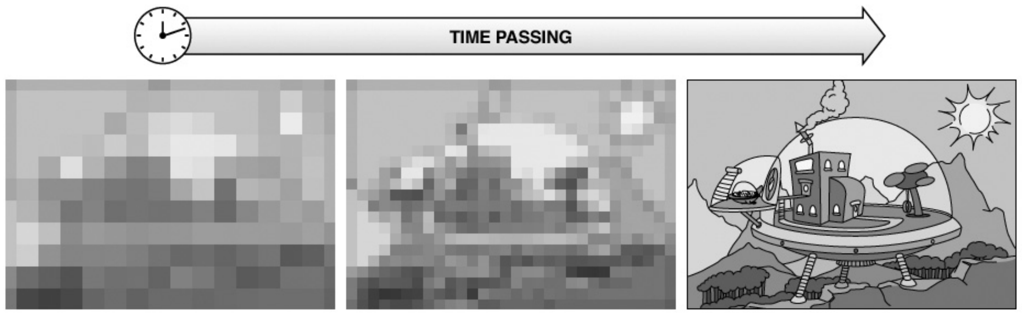


FIGURE 5-3 Interlaced GIF images show the gist of an image quickly

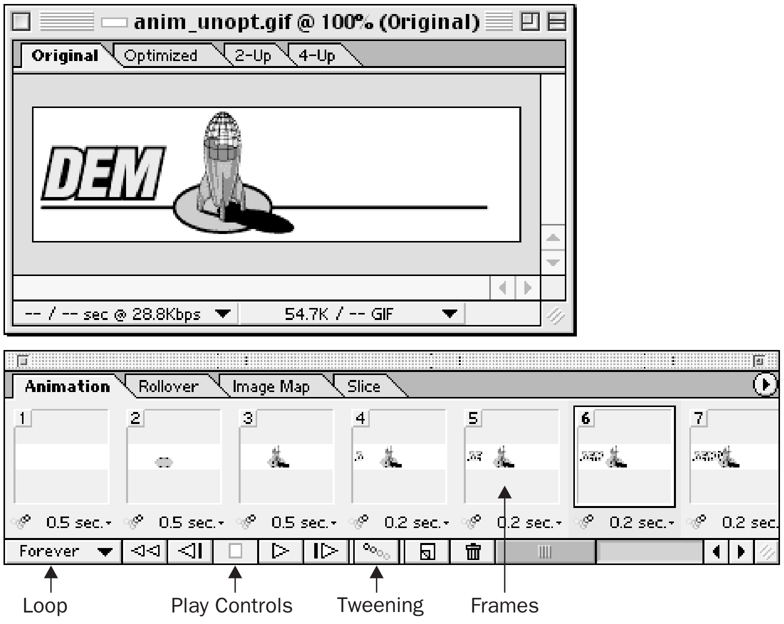


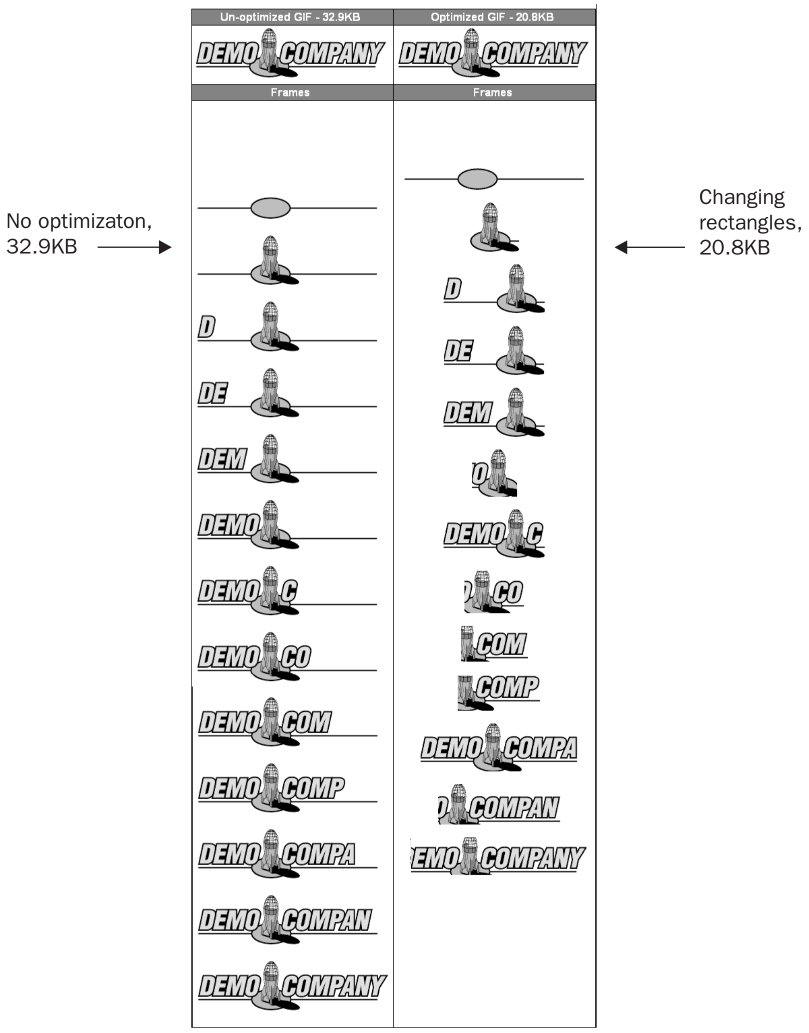
FIGURE 5-4 Animated GIFs provide only basic animation controls.

Because GIF animation is basically image after image, the file size is the total of all the images within the animation combined, which can result in a much larger image than the user is willing to wait for. Thus, it is very important to make sure that every frame of the animation is compressed as much as possible. One approach to combat file bloat is to optimize the image by replacing only the moving parts of an individual animation frame. This is often dubbed *changing rectangles* optimization. By replacing only the portion of the frame that is changing, you can use smaller images in some frames to help cut the file size down. Most of the GIF animating applications have a feature built in that will go through and optimize the images for you. This can result in a dramatic saving of file size, as shown in Figure 5-5.

JPEG

The other common Web image format is JPEG, which is indicated by a filename ending with .jpg or .jpeg. JPEG, which stands for the Joint Photographic Experts Group—the name of the committee that wrote the standard—is a lossy image format designed for compressing photographic images that may contain thousands, or even millions, of colors or shades of gray. Because JPEG is a lossy image format, there is some trade-off between image quality and file size. However, the JPEG format stores high-quality 24-bit color images in a significantly smaller amount of space than GIF, thus saving precious disk space or download time on the

FIGURE 5-5 Example of animated GIF frames and optimization



Web. See Figure 5-6 for an example of the quality versus file size tradeoff with JPEGs. Notice the significant file size savings obtained by sacrificing a little quality.

The trick with JPEG's lossy compression is that it focuses on slight smudging in areas of heavy detail that a viewer is unlikely to notice. However, in a situation where continuous color or text is used, JPEG's compression scheme may quickly become evident, as the artifacts introduced into the image will appear heavy in the flat color and text regions. It is possible to avoid this issue by selectively compressing portions of the image using an image manipulation program such as Fireworks or Photoshop.

While the JPEG format may compress photographic images well, it is not well suited to line drawings, flat color illustrations, or text. Notice the comparison between GIF and JPEG file sizes in Figure 5-7.

FIGURE 5-6
JPEG file size and quality comparison

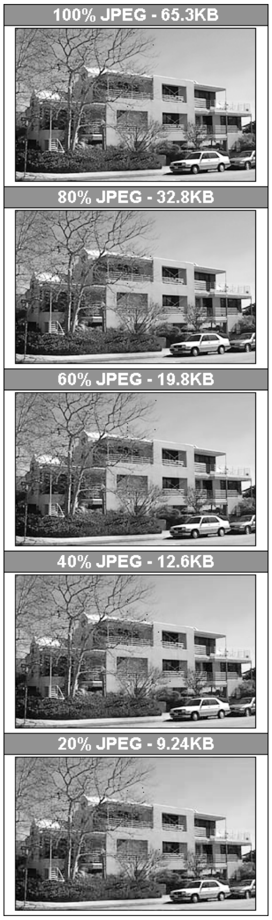
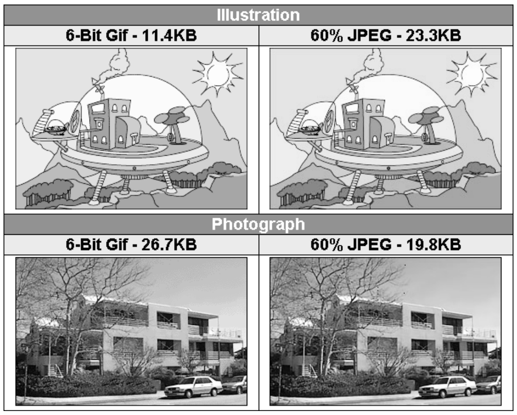


FIGURE 5-7
Comparison of GIF and JPEG files



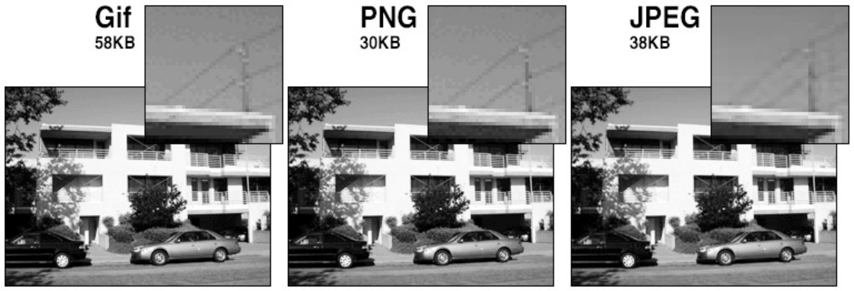
From Figure 5-7, it would seem that choosing between GIF and JPEG is usually very straightforward; photos suggest JPEG and illustrations GIF. However, in certain instances developers may be willing to distort a photo to put it in GIF in order to use the format's features because the JPEG format does not support animation, or any form of transparency. Fortunately, JPEG images do support a similar feature to GIF interlacing in a format called *progressive JPEG.* Progressive JPEGs fade in from a low resolution to a high resolution, going from fuzzy to clear. Like interlaced GIFs though, progressive JPEG images are slightly larger than their nonprogressive counterparts.

Finally, some designers are aware of the fact that because JPEG images are heavily compressed, decompression time can occasionally be a factor. With today's more powerful computers and higher speed lines, the decompression time of a JPEG will not be as noticeable much of the time. However, if you make an extremely large dimension JPEG and compress it highly, you will notice a delay. Of course, if you used a GIF, you'd have a worse looking image that might be just as large.

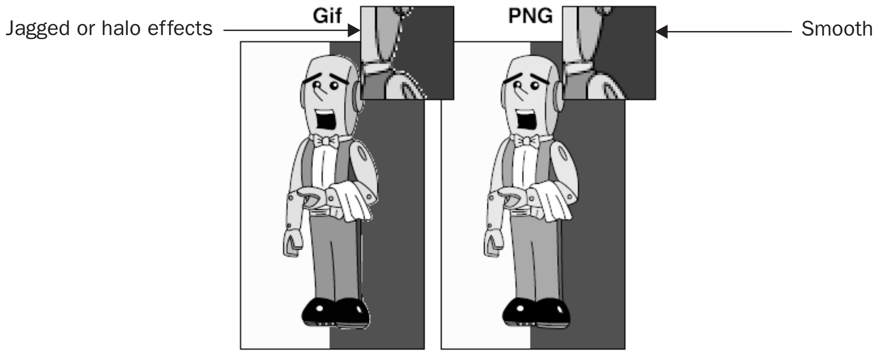
PNG

The Portable Network Graphics (PNG) format is an emerging format that has all of the features of GIF in addition to several other features. The compression algorithm for PNG is not proprietary compared to GIFs, which use LZW (owned by Unisys). Some designers have worried about the potential problems stemming from Unisys patent claims on LZW compression, but so far this has been a nonissue. PNG's compression algorithm is also slightly better than GIF's, as shown in Figure 5-8, but this alone is probably not much of a reason to give up GIF images given the browser compatibility problems that still plague the PNG format. PNG also supports slightly improved interlacing.

PNG images break the 8-bit color barrier normally found in GIF images, but with the degree of compression available in PNGs today it would not make sense to favor PNG files over JPEGs, as shown here.



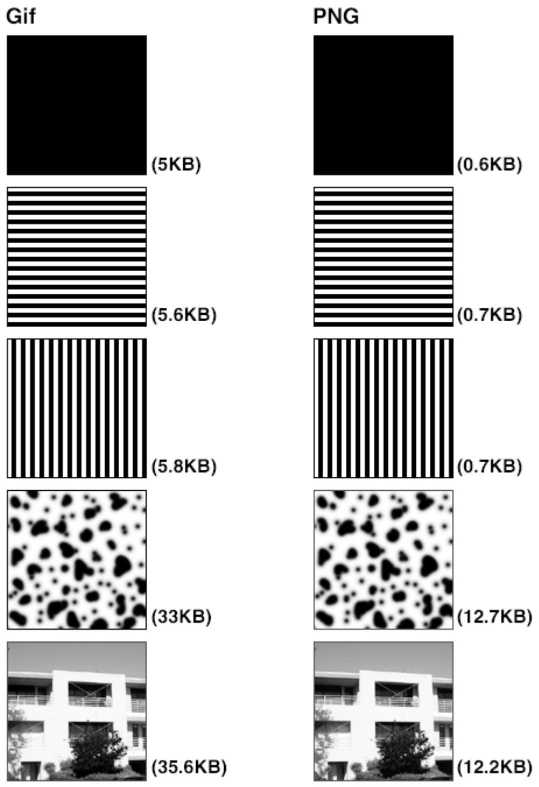
A significant plus for PNG images is the improved transparency possibilities. Rather than being limited to a single color for transparency masks, PNG files can use up to 256 colors in a mask, which lends itself to smooth transparent edges and shadow effects.

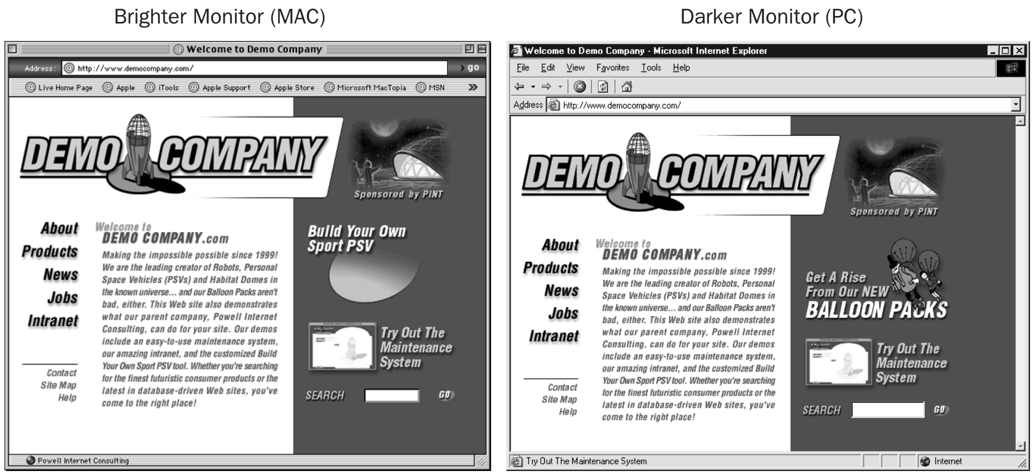


Another problem addressed by PNG is the apparent color shifting in images that are developed on a system with one Gamma or brightness value and shown on a system with different Gamma. Notice in Figure 5-9 how the images do not quite look the same at different Gamma values. PNG avoids this problem.

Finally, PNG supports animation through its related MNG (Multiple-image Network Graphics) format, similar to what is provided in GIF animations.

FIGURE 5-8 PNG Compression vs. GIF Compression




Brighter Monitor (MAC)
Darker Monitor (PC)
Welcome to Demo Company
Welcome to Demo Company - Microsoft Internet Explorer
http://www.democompany.com/
DEMO COMPANY
Sponsored by PINT
About
Products
News
Jobs
Intranet
Contact
Site Map
Help
Build Your Own Sport PSV
Get A Rise From Our NEW BALLOON PACKS
Try Out The Maintenance System
SEARCH
GO
Powell Internet Consulting
Try Out The Maintenance System
Internet

Welcome to DEMO COMPANY.com

Making the impossible possible since 1999! We are the leading creator of Robots, Personal Space Vehicles (PSVs) and Habitat Domes in the known universe... and our Balloon Packs aren't bad, either. This Web site also demonstrates what our parent company, Powell Internet Consulting, can do for your site. Our demos include an easy-to-use maintenance system, our amazing intranet, and the customized Build Your Own Sport PSV tool. Whether you're searching for the finest futuristic consumer products or the latest in database-driven Web sites, you've come to the right place!

FIGURE 5-9 Different Gamma values can cause images to look different

With all these great features, one wonders why PNG is not more common online. The main reason is that the browser vendors still don't consistently support PNG images. Even when the image format is supported, many features such as transparency are not fully implemented. Still, few browsers except those based upon Mozilla and Macintosh Internet Explorer support PNG well enough to rely on the format, so Web designers are warned to avoid using PNGs unless browser sensing is utilized.

Other Image Formats

There are many image formats in addition to GIF, JPEG, and PNG that can be used on the Web. Vector formats such as Flash (with the file extension .swf) or Scalable Vector Graphics (SVG) are available, and image files may even use exotic compression technology such as fractal or wavelet compression. Most of the less common image formats require a helper application or plug-in to allow the image to be displayed. Unless you have a specific need, you probably should avoid special image types requiring browser add-ons; users may become frustrated by the work involved in obtaining the extra software.

For now, let's assume that a page designer simply has a Web-compatible image that needs to be placed into a Web page and requires the appropriate HTML syntax to do so.

HTML Image Basics

To insert an image into a Web page, use an **<img>** tag and set its **src** attribute equal to the URL of the image. As discussed in Chapter 4, the form of the URL can be either an absolute URL or a relative URL. Most likely, the image element will use a relative URL to an image found locally. To insert a GIF image called logo.gif residing in the same directory as the current document, use

```
<img src="logo.gif">
```

Because **img** is an empty element under XHTML, you would use

```
<img src="logo.gif" />
```

We'll use the XHTML syntax from here on. Of course, in the previous example, an absolute URL also could be used to reference an image on another server.

```
<img src="http://www.democompany.com/images/logo.gif" />
```

Using an external URL is not advised because images can move or cause the page to load at an uneven pace.

NOTE *The* ***src*** *attribute must be included. Otherwise, browsers that support images might display a placeholder or broken image icon.*

To set up a simple example, first create a directory to hold your images. It usually is a good idea to store all your image media in a directory named "images." This helps you keep your site contents organized as you build the Web site. Now place a GIF format image named robot.gif in that directory. To retrieve an image from the Internet, you can simply right-click with your mouse on an image and save the file to your directory. Macintosh users must hold the mouse button down on an image to access the menu for saving the image. Once you have a GIF image, you should be able to use a short piece of HTML markup to experiment with the use of **img**, as shown in the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Image Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<h2 align="center">Image Example</h2>
  <img src="images/robot.gif" alt="robot" width="156"
  height="251" border="0" />
</body>
</html>
```

NOTE *The name of the image, its path, its width, and height are made up for this example. Your particular attribute values might be different.*

A rendering of the image example is shown in Figure 5-10.
The next few sections cover the basic attributes of **img**.

Alternative Text Using the alt Attribute

The **alt** attribute, which is required under HTML and XHTML specifications, provides alternative text for user agents that do not display images, or for graphical browsers where the user has turned off image rendering.

```
<img src="images/logo.gif" alt="Demo Company Logo" />
```

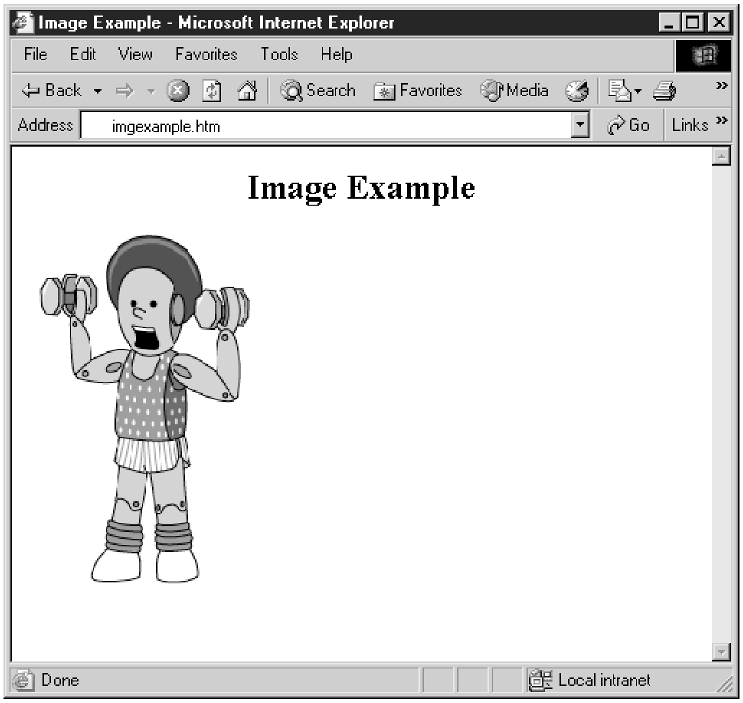
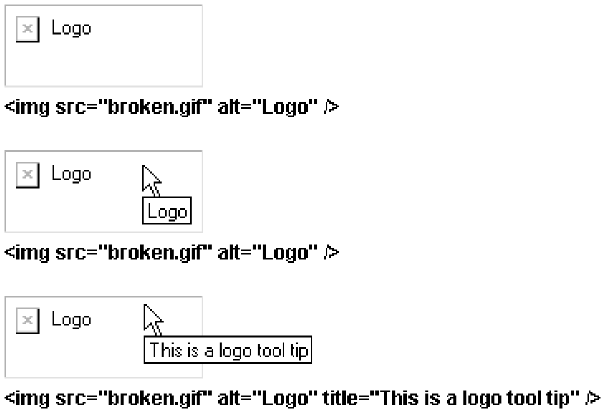


Image Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address imgexample.htm Go Links
Image Example
Done
Local intranet

FIGURE 5-10 Rendering of a simple <img> example

The **alt** attribute's value may display in place of the image or be used as a Tooltip or placeholder information in image-based browsers. Any HTML markup found in the **alt** element will be rendered as plain text. If the option to display images is turned off, the browser displays the alternative text, as shown here:



```
<img src="broken.gif" alt="Logo" />
<img src="broken.gif" alt="Logo" />
<img src="broken.gif" alt="Logo" title="This is a logo tool tip" />
```

A browser may also show the **alt** text as images load, giving the user something to read as the page renders.

Many modern graphical browsers will also display the **alt** text as the Tooltip for the image once the pointer is positioned over the image for a period of time. However, the core attribute **title** should override this and be displayed instead of the **alt** text in a conformant browser, as shown in the previous illustration.

While theoretically there is no limit to the alternative text that can be used, anything more than a few hundred characters may become unwieldy. Some browsers do not handle long Tooltips and **alt** text properly, and may not wrap the descriptive text. However, be warned that if you insert entities such as **&#13;**, which indicates a carriage return, to format the **alt** or **title** text, you may wreak havoc in voice browsers that read screen content, though the visual presentation might be improved.

The **alt** attribute's importance becomes clear when you reflect on how many people access the Web from a text-only environment. Unfortunately, much of the alternative text set does not always provide a substantial benefit. Do the previous examples really help by backing up the Demo Company logo graphic with the actual words "Demo Company logo"? Would simply "Demo Company logo" be sufficient, or insufficient? Try to make **alt** text reflect the meaning of an image; if an image is merely decorative, like a graphic bullet for a list item, setting to no value (**alt=""**) is perfectly acceptable.

Although a lot of people might argue that the Web wasn't popular until graphics were integrated or that the Web inherently is a visual medium, the value of textual content on the Web is indisputable. Consequently, it should be made as accessible as possible. There is no arguing that a picture might be worth a thousand words; but if that is the case, why not provide a few words in exchange?

Image Alignment

Probably the first thing a user wants to do after he or she is able to put an image in a Web page is to figure out how to position it on the page. Under the original HTML 2 standard, there was very little that allowed the user to format image layout on a page. Initially, the **align** attribute could be set to a value of **top**, **bottom**, or **middle**. When an image was included within a block element, the next line of text would be aligned either to the top, middle, or bottom of the image depending on the value of the **align** attribute. If the attribute wasn't set, it would default to the bottom. The example that follows illustrates basic image alignment as first defined in HTML 2. The rendering of the image alignment example is shown in Figure 5-11.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Basic Image Alignment</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<p><img src="images/aligntest.gif" align="top" alt="" border="1" />
This text should be aligned to the top of the image.</p>

<p><img src="images/aligntest.gif" align="middle" alt="" border="1" />
```

```
This text should be aligned to the middle of the image.</p>

<p><img src="images/aligntest.gif" align="bottom" alt="" border="1" />
This text should be aligned to the bottom of the image.</p>
</body>
</html>
```

One of the problems with image alignment in early HTML was that the text really didn't flow around the image. In fact, only one line of text was aligned next to the image, which meant the inline images had to be very small or the layout looked somewhat strange.

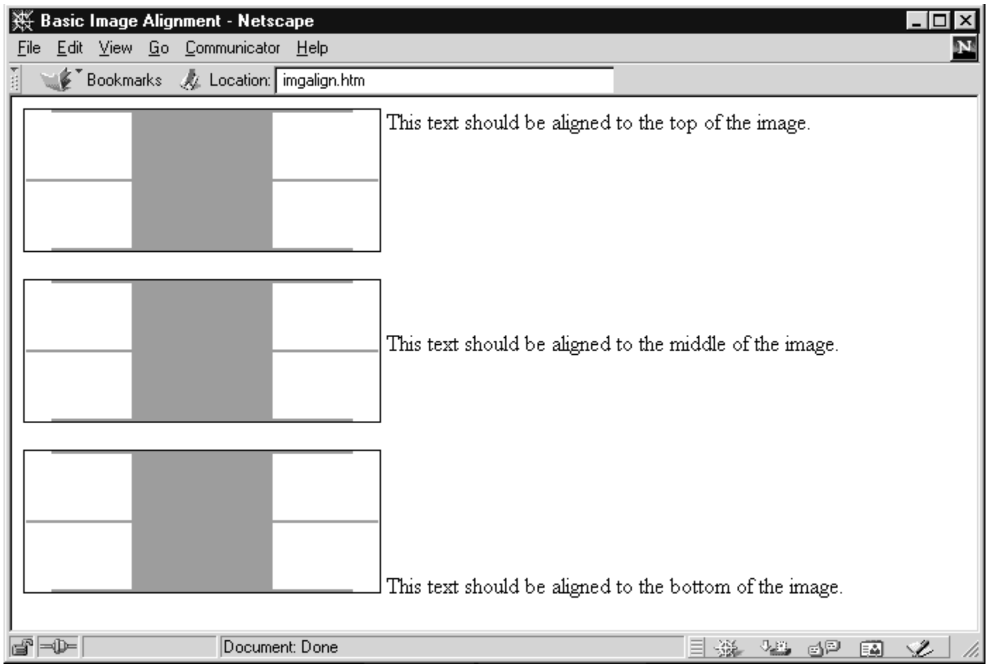
Netscape eventually introduced the **left** and **right** values for **align**, which allowed text to flow around the image. These values were later incorporated into the HTML specification, but eventually, like other image presentation values, were deprecated under strict HTML and XHTML. When setting an image element such as **<img src="logo.gif" align="left" />**, the image is aligned to the left and the text flows around to the right. Correspondingly, when you are using markup such as **<img src="logo.gif" align="right" />,** the image is aligned to the right and the text flows around to the left. It is even possible to flow the text between two objects if things are done carefully. The example presented here shows how the **align** attribute would work under transitional variants not using CSS. The rendering of this example is shown in Figure 5-12.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>Improved Text Flow</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<p>
<img src="images/redsquare.gif" alt="red square" align="left" />
The top image has its align attribute set to "left," so the text flows
around it to the right. The top image has its align attribute set to
"left," so the text flows around it to the right. The top image has its
align attribute set to "left," so the text flows around it to the right.

<br clear="left" /><br /><br />

<img src="images/redsquare.gif" alt="red square" align="right" />
The bottom image has its align attribute set to "right," so the text flows
around it to the left. The bottom image has its align attribute set to
"right," so the text flows around it to the left. The bottom image has its
align attribute set to "right," so the text flows around it to the left.
</p>
</body>
</html>
```

Notice in the previous example that there is a special attribute to the **br** element. This is necessary to force the text to flow properly and will be discussed shortly.

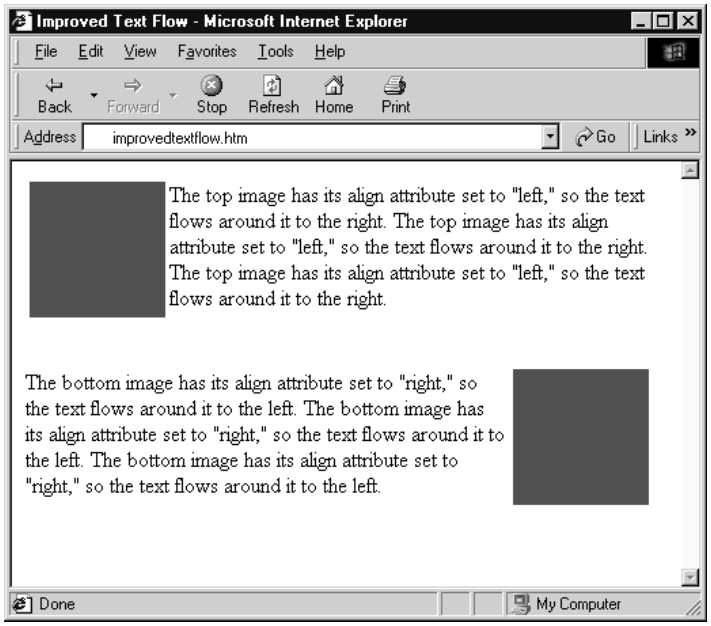

Basic Image Alignment - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: imgalign.htm
Document: Done

This text should be aligned to the top of the image.

This text should be aligned to the middle of the image.

This text should be aligned to the bottom of the image.

FIGURE 5-11 Image alignment rendering


Improved Text Flow - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Home
Print
Address improvedtextflow.htm
Go
Links
Done
My Computer

The top image has its align attribute set to "left," so the text flows around it to the right. The top image has its align attribute set to "left," so the text flows around it to the right. The top image has its align attribute set to "left," so the text flows around it to the right.

The bottom image has its align attribute set to "right," so the text flows around it to the left. The bottom image has its align attribute set to "right," so the text flows around it to the left. The bottom image has its align attribute set to "right," so the text flows around it to the left.

FIGURE 5-12 Image alignment rendering

Note *Netscape and Microsoft also support four other values for **align**: **texttop**, **baseline**, **absmiddle**, and **absbottom**. These attributes should be avoided in most cases because they are not standard, not supported consistently across browsers, and have been superseded by style sheets. In fact, formatting and positioning of images in general is handled more precisely by style sheets, which are discussed in Chapters 10 and 11.*

Buffer Space: hspace and vspace

Just floating an image and allowing text to wrap around it might not be adequate. You must also consider how to position the image more precisely with the text and make sure that text breaks where it ought to. Initially introduced by Netscape and made official in HTML 3.2, the **hspace** and **vspace** attributes can be used to introduce "runaround" or buffer space around an inline image. The **hspace** attribute is used to insert a buffer of horizontal space on the left and right of an image, whereas the **vspace** attribute is used to insert a buffer of vertical space between the top and bottom of the image and other objects. The value of both attributes should be a positive number of pixels. It is also possible to set the attribute values to percentage values, although this is inadvisable, as very high values can produce strange results. However, the most problematic aspect of the **hspace** and **vspace** attributes is the amount of buffer space that occurs on both sides of the image. Take a look at the XHTML transitional markup shown here to see how **hspace** and **vspace** work. Figure 5-13 displays a possible browser rendering of the example code.

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

<p>The image below has its <tt><b>&lt;hspace&gt;</b></tt> and
<tt><b>&lt;vspace&gt;</b></tt> attributes set to 50 pixels, so the
text will flow around it at a distance of 50 pixels. The rest of
this text is dummy text. If it said anything interesting you would
certainly be the first to know.

<img src="images/redsquare.gif" align="left" alt="red square"
hspace="50" vspace="50" />

This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
```

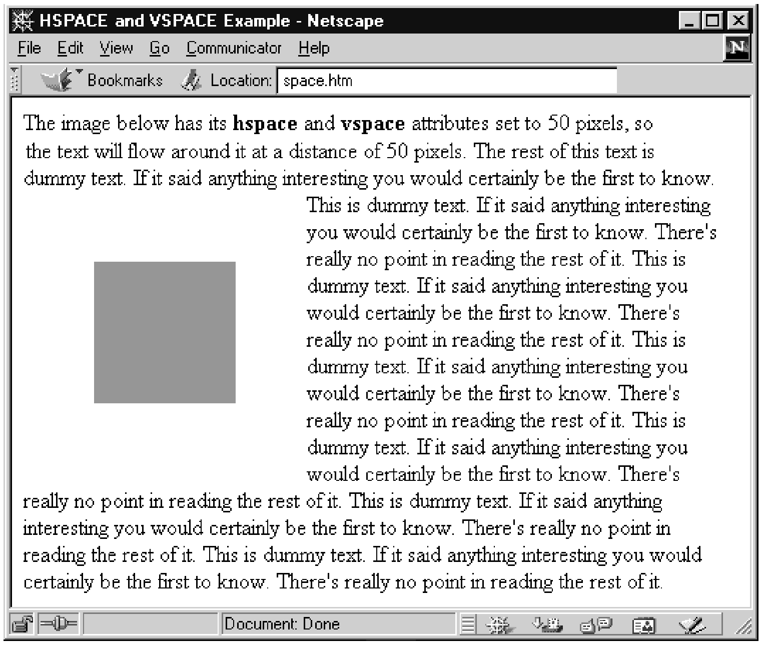
```
This is dummy text. If it said anything interesting you would certainly
be the first to know. There's really no point in reading the rest of it.
</p>
</body>
</html>
```

It turns out that in the future, by using style sheets (discussed in Chapter 10), it is possible to avoid these somewhat imprecise layout features altogether. The **hspace** and **vspace** attributes have been very useful, albeit occasionally abused by Web designers.

Extensions to <br>

In flowing text around an image, a designer may encounter a situation in which he or she wants to clear the text flow around the image. For example, it could be problematic to create an image with a caption like the one shown in Figure 5-14 because the text might reflow.

To deal with such problems, a new attribute called **clear** was added to the **br** element; this extension now is part of the HTML standard, though of course it is deprecated under strict HTML and XHTML by CSS, which provides a **float** property that does the same thing. Under older HTML versions and transitional XHTML, the **clear** attribute can be set to **left**, **right**, **all**, or **none** and will clear the gutter around an inline object such as an image. For example, imagine the fragment **<img src="photo. gif" align="left" />** with text wrapping around it. If **<br clear="left" />** is included in the text and the wrapped text is still wrapping around the image, the text will be cleared to pass the image. The **clear="right"** attribute to a <**br />** tag works for text flowing around right-aligned images. Using a value of **all** ensures


HSPACE and VSPACE Example - Netscape
File Edit View Go Communicator Help
Bookmarks Location: space.htm
Document: Done

The image below has its **hspace** and **vspace** attributes set to 50 pixels, so the text will flow around it at a distance of 50 pixels. The rest of this text is dummy text. If it said anything interesting you would certainly be the first to know. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it. This is dummy text. If it said anything interesting you would certainly be the first to know. There's really no point in reading the rest of it.

FIGURE 5-13 Rendering of hspace and vspace example

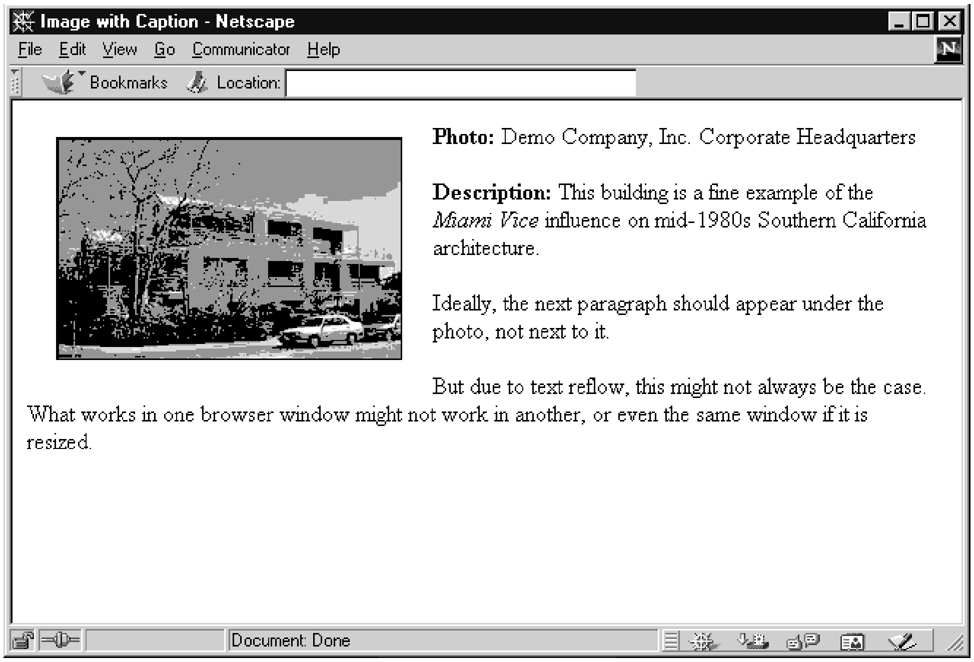


Image with Caption - Netscape
File Edit View Go Communicator Help
Bookmarks Location:
Document: Done

Photo: Demo Company, Inc. Corporate Headquarters

Description: This building is a fine example of the *Miami Vice* influence on mid-1980s Southern California architecture.

Ideally, the next paragraph should appear under the photo, not next to it.

But due to text reflow, this might not always be the case. What works in one browser window might not work in another, or even the same window if it is resized.

FIGURE 5-14 Image with misaligned caption

that the <**br** /> tag continues to break text until both left and right columns are clear. Setting the attribute to **none** makes the element act as it normally would and is implied when using the **<br />** by itself. An example of the use of this attribute is shown here; a rendering appears in Figure 5-15.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<title>BR Clear Example</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
<p>
<img src="images/building.jpg" width="234" height="150" border="2"
     alt="Outside of the DemoCompany corporate headquarters"
     align="left" hspace="20" vspace="10" />

<b>Photo:</b> Demo Company, Inc Corporate Headquarters<br /><br />

<b>Description:</b> This building is a fine example of the <i>Miami
Vice</i> influence on mid-80s southern California architecture.
<br /><br /></p>
```

```
<p>The next paragraph should appear under the photo, not next to it,
thanks to <b>&lt;br clear=&quot;left&quot: / &gt;</b>.
<br clear="left" />
<i>Photo copyright &copy; 2000 by Demo Company, Inc.</i>
</p>
</body>
</html>
```

height and width

The **height** and **width** attributes to the **img** element, introduced in HTML 3.2, are used to set the dimensions of an image. The value for these attributes is either a positive pixel value or a percentage value from 1–100 percent. Although an image can be stretched or shrunk with these attributes, the main purpose of **height** and **width** actually is to reserve space for images that are being downloaded. As pages are requested by a browser, each individual image is requested separately. However, the browser can't lay out parts of the page, including text, until the space that the image takes up is determined. This might mean waiting for the image to download completely. By telling the browser the height and width of the image, the browser can go ahead and reserve space with a bounding box into which the image will load. Setting the height and width thus allows a browser to download and lay out text quickly while the images are still loading. For an image called test.gif that has a height of 10 and a width of 150, use **<img src="test.gif" height="10" width="150" />**. The usability improvement of using **height** and **width** attributes for images is significant, and they should always be included.

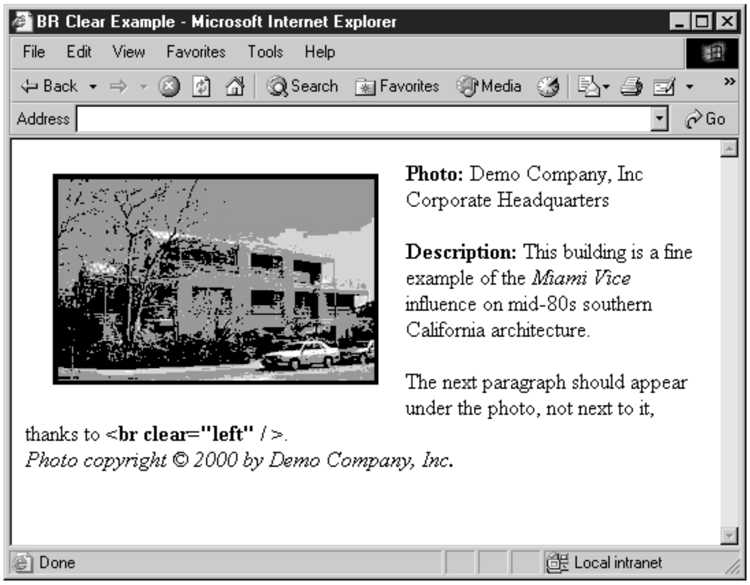

BR Clear Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go
Done
Local intranet

Photo: Demo Company, Inc Corporate Headquarters

Description: This building is a fine example of the *Miami Vice* influence on mid-80s southern California architecture.

The next paragraph should appear under the photo, not next to it, thanks to <**br clear="left"** / >.
Photo copyright © 2000 by Demo Company, Inc.

FIGURE 5-15 Rendering of <br clear> example

NOTE *Many people wonder what the measurements of a particular image are. Using Netscape, it is possible to view the dimensions quite easily. First, load the image into the browser by itself without any accompanying HTML. Now look at the title bar of the browser, which should display the dimensions. Also, using the option to view document information for the image within the browser should reveal the dimensions. Most Web editors also can automatically show the dimensions of an image.*

In addition to the prelayout advantages, the **height** and **width** attributes can also be used to size images. This is rarely a good idea, as the image might end up distorted. One way to avoid distortion is to shrink images in a proportional manner. However, if the image is to be made smaller, it is a better idea to size the image appropriately in a graphics program. Shrinking the image with the **height** and **width** attributes does not affect the file size, whereas resizing the image beforehand will shrink the file, hence reducing the download time. Another use of **height** and **width** sizing might be to increase the size of a simple image. For example, imagine an image of a single green pixel, and set the height and width alike: **<img src="greenpixel.gif" height="100" width="100" />**. The resulting image is a large green box with very little download penalty. A few sites even use the **height** and **width** attributes with percentage values such as 100 percent to create interesting effects such as full-screen images or vertical or horizontal color bars.

One other interesting use of the **height** and **width** attributes would be to help preload images. Preloading can be used to create the illusion of a quick download. Imagine that during the idle time on a page, the images on the next page are being downloaded so that they are precached when the user goes to the next page. A significant perceived performance improvement is achieved. One way to perform this prefetching is by putting an image that will appear later on the current page with **height** and **width** both set to **1**. In this case, the image won't really be visible but will be fully loaded into the browser's cache. Once the user visits the next page, the image can be fetched from the local disk and displayed quickly. The **link** element extension for prefetching content discussed in Chapter 4 should really be used over this **<img>** tag trick.

Low Source Images

Another potential speed improvement introduced by Netscape and supported by many browsers despite not being part of the HTML or XHTML standards is offered by the **lowsrc** attribute. The **lowsrc** attribute should be set to the URL of an image to load in first, before the so-called high source image indicated by the **src** attribute. In this sense, the attribute can be set to the address of a low-resolution or black-and-white file, which can be downloaded first and then followed by a high-resolution file. Consider the following:

```
<img src="hi-res-photo.gif" lowsrc="bw-photo.gif" height="100"
width="100" alt="Outside of building photograph" />
```

The **lowsrc** attribute can provide significant usability improvement when large full-screen images must be used.

One interesting aspect of the **lowsrc** attribute is that the browser tends to use the image dimensions of the **lowsrc** file to reserve space within the Web page if the **height** and **width** attributes are not set. Because of this, some strange distortion could happen if the high-resolution image is not the same size as the low-resolution image.

These are only the most common attributes for the **img** element. A more complete listing of **img** element attributes can be found in the element reference in Appendix A.

Images as Buttons

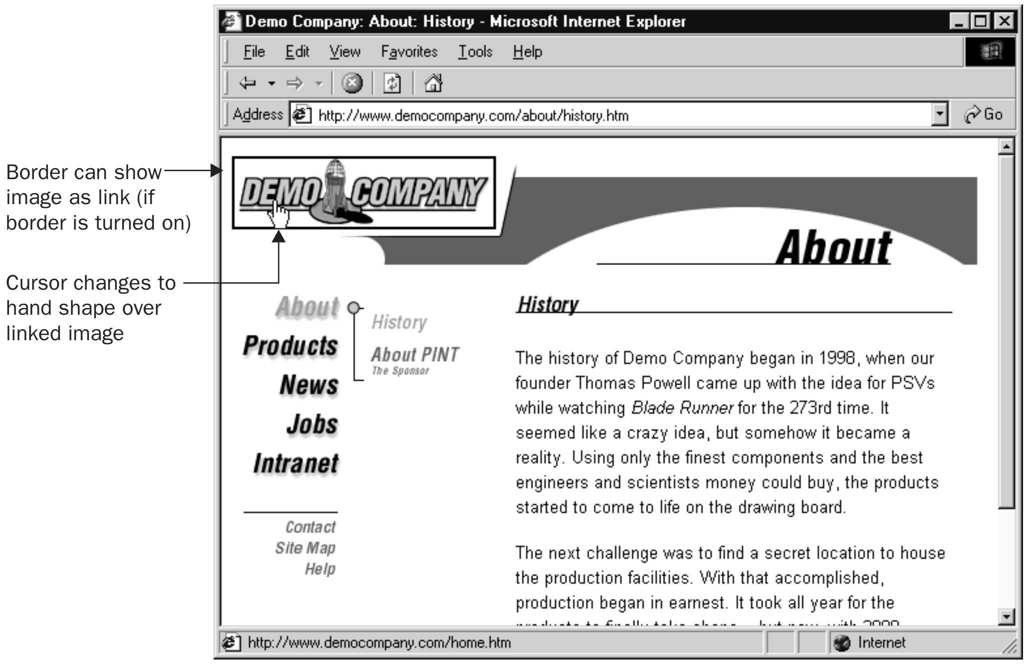
One of the most important aspects of images, as previously discussed in Chapter 4, is how they can be combined with the **a** element to create buttons. To make an image "pressable," simply enclose it within an anchor.

```
<a href="http://www.democompany.com"><img src="logo.gif"
alt="Demo Company" /></a>
```

When the page is rendered in the browser, clicking on the image will take the user to the anchor destination specified. Generally, to indicate that an image is pressable, the browser puts a border around the image, and provides some feedback to the user when the cursor or pointing device is over the hot area, such as turning the pointer to a finger or highlighting the text. For some basic feedback types, see Figure 5-16, which shows a border, finger pointer, and URL destination—all indicating that the image is pressable.

One issue that might be troublesome for page designers is the border that appears around the image when it is made pressable. It is possible to turn this border off by setting the **border** attribute of the image equal to **0**. Consider the following:

```
<a href="http://www.democompany.com"><img src="logo.gif"
alt="Demo Company"  border="0" /></a>
```


Border can show image as link (if border is turned on)
Cursor changes to hand shape over linked image
Demo Company: About: History - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.democompany.com/about/history.htm
Go
DEMO COMPANY
About
About
History
Products
About PINT
The Sponsor
News
Jobs
Intranet
Contact
Site Map
Help
http://www.democompany.com/home.htm
Internet

History

The history of Demo Company began in 1998, when our founder Thomas Powell came up with the idea for PSVs while watching *Blade Runner* for the 273rd time. It seemed like a crazy idea, but somehow it became a reality. Using only the finest components and the best engineers and scientists money could buy, the products started to come to life on the drawing board.

The next challenge was to find a secret location to house the production facilities. With that accomplished, production began in earnest. It took all year for the

FIGURE 5-16 Image as link feedback

Of course, without the border it may be difficult to determine which images on a page are links and which are not. This can cause users to play a little game of finding the active click region by running their mouse all over the screen. One way to avoid such usability problems is to provide visual cues in images that are made pressable. Although from a design perspective some of these effects, particularly drop shadows, are a little overused, there are tangible benefits to adding feedback information to button graphics. Another approach is to animate the buttons. Using a very simple piece of JavaScript, it is possible to animate a button so that when a mouse passes over an image it comes alive. A brief discussion about how Web pages can be made more dynamic using a scripting language such as JavaScript can be found in Chapter 14.

One nonbutton-oriented use of the **border** attribute is to put a simple stroke around an image. Many times people will use a graphics tool to create a frame on an image, but the **border** attribute is a bandwidth-cheap way to get much of the same effect. Try setting the **border** attribute equal to a positive value on a nonclickable image—for example, **< img src="portrait.gif" alt="" border="5" />**. Borders, particularly when added with CSS, which offers a much richer set of formatting possibilities, provide an easy way to frame an image.

Image Maps

Another form of clickable images, discussed previously in Chapter 4, is the image map. An image map is a large image that contains numerous hot spots that can be selected, sending the user to a different anchor destination. Recall from the previous chapter that there are two basic forms of image maps: *server-side* and *client-side*. In the server-side image map, the user clicks on an image but the server must decode where the user clicked before the destination page (if any) is loaded. With client-side image maps, all of the map information—which regions map to which URLs—can be specified in the same HTML file that contains the image. Including the map data with the image and letting the browser decode it has several advantages, including the following:

- There is no need to visit a server to determine the destination, so links are resolved faster.
- Destination URLs can be shown in the status box as the user's pointer moves over the image.
- Image maps can be created and tested locally, without requiring a server or system administration support.
- Client-side image maps can be created so that they present an alternate text menu to users of text-only browsers.

Although it's clear that client-side image maps are far superior to their server-side cousins, very old browsers may not support this feature. This does not have to be a problem, as it is possible to include support for both types of image maps at once.

Server-Side Image Maps

To specify a server-side image map, the **a** element is used to enclose a specially marked **img** element. An **<a>** tag's **href** attribute should be set to the URL of the program or map file to decode the image map. The enclosed **<img>** tag must contain the attribute **ismap** so the browser can decode the image appropriately. As with all linked images, it might be desirable to turn the image borders off by setting the **<img>** tag's **border** attribute equal to **0**. As mentioned in Chapter 4, server-side image maps do not provide adequate feedback to the user because they show coordinates, and may incur performance penalties. HTML authors are encouraged to use client-side image maps.

Client-Side Image Maps

The key to using a client-side image map is to add the **usemap** attribute to an <**img>** tag and have it reference a **map** element that defines the image map's active areas. An example of this syntax is **<img src="controlbar.gif" usemap="#controlmap" />**. Note that, like server-side image maps, the image will be indicated as a link regardless of the lack of an <**a>** tag enclosing the image. The **border** attribute should be set to **0** if necessary. The **map** element generally occurs within the same document, although it might be possible to link to a **map** element outside the document though this use is uncommon and support in browsers is inconsistent. While the **map** element can occur anywhere within the body of an HTML document, it usually is found at the end of HTML documents.

The **map** element has two important attributes, **name** and **id**, which are used to specify the identifier associated with the map. While **id** is standard XHTML, browser support still often favors **name**, so both are included for safety purposes. The map name then is referenced within an <**img>** tag using the **usemap** attribute and the associated fragment identifier. Within the <**map>** tag are "shapes" defined by **<area>** tags that are mapped onto an image and define the hot spots for the image map. A brief example is shown here with a detailed discussion in Chapter 4 and full syntax of related tags in Appendix A.

```
<img src="shapes.gif" usemap="#shapes" alt="shapes map"
border="0" width="400" height="200" />

<div>
<!-- start of client side image map -->
<map name="shapes" id="shapes">
<area shape="rect" coords="6,50,140,143" href="rectangle.html"
      alt="rectangle" />
<area shape="circle" coords="195,100,50" href="circle.html"
      alt="circle" />
<area shape="poly"
      coords="255,122,306,53,334,62,338,0,388,77,374,116,323,171,255,122"
      href="polygon.html" alt="polygon" />
<area shape="default" href="defaultreg.html" alt="" />
</map>
</div>
```

While the format of the mapping tags is discussed in Chapter 4, memorizing or creating client- or server-side image maps by hand is not advised. Page designers should find that most Web page editors like Macromedia Dreamweaver or HomeSite automate the creation of image hot spots.

Advanced Image Considerations

Although most of the basic uses of images have been discussed, there are some issues that should be mentioned for later discussion. First, because an image can be referenced by a style sheet or by a scripting environment, it might be very important to provide a name or identifier for it. The **class**, **id**, and **name** attributes can be used to provide names for images so they can be referenced and manipulated by scripting or style information that usually is found in the head of the document. Names should be unique and in the proper HTML form.

It is possible to include inline scripting or style information directly with an image. For example, setting the **style** attribute allows an inline style to bind to a particular **<img>** tag. Style sheets are discussed in Chapters 10 and 11. Furthermore, it is possible to have images bound to a particular event using an event attribute such as **onmouseover** and tying it to a script. A very simple use of tying an event with an image is to have the image change state depending on the user's action. The most basic use would be to create animated buttons or buttons that make a sound when clicked, but the possibilities are endless. A more detailed discussion and examples of how to bind JavaScript to create animated buttons are presented in Chapter 14.

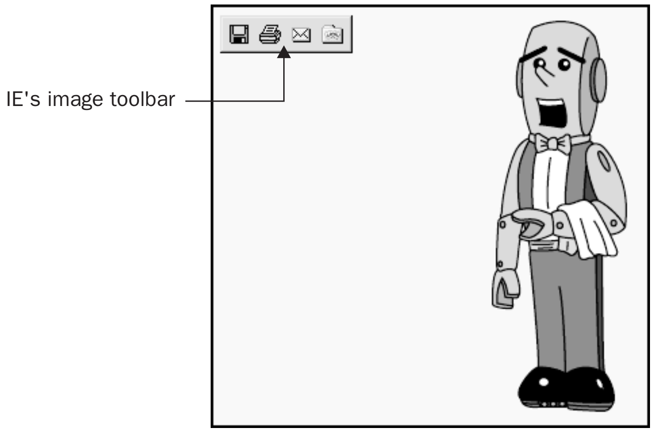
An important comment concerning the future of **img** is that starting with HTML 4, it is supposed to be possible to include images using an **<object>** tag. For example,

```
<object data="images/logo.gif">Picture of the Demo Company
building</object>
```

Similar to the **<img>** tag, the **data** attribute is set to the URL of the included image while the alternative rendering is placed within the **object** element. Although this new syntax might create some interesting possibilities, the reality is that browsers currently don't support this form of image inclusion. Whereas this generic **<object>** tag for image support makes sense given that an image is no different from any other included binary object, the fact is that until browser vendors embrace this, it should be avoided. A more complete discussion of this element can be found in Appendix A, which provides the full syntax of the **object** element.

Image Toolbar

A special browser-specific feature for images that necessitates some special comment is Internet Explorer 6's image toolbar. If you have ever held your mouse over a large image in a Web page using IE6, you may have noticed a strange pop-up toolbar.



The toolbar supports quick saving of images to a special "My Images" folder. The browser determines what images to show this toolbar for by looking at the dimension of the image. Typically, the image must be fairly large to receive an image tool bar, and using just this simple idea, the browser does a pretty good job of not showing this feature of navigation buttons and banner ads. But for everything else, it depends on if you pass its size threshold. To turn off the image toolbar on an individual image, just add the **galleryimg** attribute and set its value to **no**, like so:

```
<img src="democompanylogo.gif" alt="Demo Company" galleryimg="no"
     height="50" width="100" />
```

If you just want to be rid of the whole feature altogether in a page, either have your server issue an HTTP response header of **Imagetoolbar: no** or, more easily, use a **<meta>** tag in the **<head>** of each page.

```
<meta http-equiv="imagetoolbar" content="no" />
```

Tips on Image Use

Many readers find Web page creation frustrating because it always seems that other sites just look better or load faster. Although this book focuses on HTML and XHTML, it's important to consider a few issues concerning image use. A much deeper discussion of image considerations can be found in *Web Design: The Complete Reference, Second Edition* (www.webdesignref.com).

Image Use

The first thing to consider is that the quality of the image being used certainly will affect the outcome of the page layout. Even when armed with a scanner, digital camera, or appropriate software such as Adobe Photoshop, Adobe Illustrator or Macromedia Fireworks, you might be a long way from being able to produce aesthetically pleasing Web pages. Don't fret—you would never expect that just owning a copy of a word processor would destine you to produce a huge book; it takes skill, patience, and years of practice. Take it from me.

Although this certainly is not a book on Web design, a simple tip is to aim for a minimal design. Straight lines, basic colors, and modest use of imagery should produce a relatively clean and uncluttered design. Furthermore, the simple design probably will load very fast! When you decide to use imagery on your site, whether for pure decoration or information, don't skimp on quality. If you use clip art from some free Web site, your site will reflect this. Fortunately, there are many sites that sell professional quality clip-illustrations and photographs relatively cheaply. While this might save money, don't simply right-click your way to a nice new image free of charge. Web users are sophisticated enough to know when they're having a cheap site foisted on them.

Legal Issues with Images

Unfortunately, the expense of licensing images and the ease with which images can be copied have convinced many people that they can simply appropriate whatever images they need. This is stealing the work of others. Although there are stiff penalties for copyright infringement, it can be difficult to enforce these laws. Also, some page designers tend to bend the rules thanks to the legal concept called *fair use*, which allows the use of someone else's copyrighted work under certain circumstances.

There are four basic questions used to define the fair-use principle:

- **Is the work in question being appropriated for a nonprofit or profit use?** The fair use defense is less likely to stand up if the "borrowed" work has been used to make money for someone other than its copyright holder.
- **Is the work creative or factual?** A creative work could be a speculative essay on the impact of a recent congressional debate; a factual work would be a straightforward description of the debate without commentary. "Fair use" would cover use of the factual work more than use of the creative one.
- **How much of the copyrighted work has been used?** It is possible to use someone else's images if it is changed substantially from the original. The problem is determining what constitutes enough change in the image to make it a new work. Simply using a photo-editing tool to flip an image or change its colors is not enough. There is a fine line between using portions of another person's work and outright stealing. Even if you don't plan on using uncleared images, be careful of using images from free Internet clip art libraries. These so-called "free" images may have been submitted with the belief that they are free, but some of them may have been appropriated from a commercial clip art library somewhere down the line. Be particularly careful with high-quality images of famous individuals and commercial products. Although such groups often might appreciate people using their images, the usage generally is limited to noncommercial purposes.
- **What impact does the image have on the economic value of the work?** Although unauthorized use of a single *Star Trek* related image might not substantially affect the money earned by Paramount Pictures in a given fiscal year, Paramount's lawyers take a dim view of such use. In fact, some entertainment organizations have taken steps to make it very difficult for Web page designers to use such images.

Ultimately we could, perhaps, add a fifth question to the list: Who owns the original work, and how vigorously will the owner defend it? With such a dangerous question it is obvious to see this discussion begs many legal questions that are far beyond the scope of this book. Suffice it to say that in the long run, it's always safer to create original work, license images, or use material in the public domain. Just because many Web designers skirt the law doesn't mean you should.

Images and Download Speed

Even if it is filled with wonderful imagery, few people want to wait literally minutes for your beautifully designed page to load. Page designers should always consider download time when adding images to their pages. Never assume that everyone has the latest high-speed cable connection or that high bandwidth is right around the corner. This section presents a few tips for improving download time of pages:

- **Make sure to use the correct format for the job.** Recall that GIF images are good for illustrations whereas JPEG images are good for photographs. If you break this rule of thumb, you may find that your images are unnecessarily big byte-wise and will take longer to download.
- **Reduce colors if possible.** When using GIF images, reducing the number of colors in the image (the bit-depth) can substantially reduce the file size. If your company logo only has 30 colors in it, why use an 8-bit GIF image when you can use a 5-bit image that supports 32 colors? Tools such as Macromedia Fireworks or Adobe Photoshop make color reduction easy to do.
- **Reduce the number of images in the page.** The number of individual images in a page can substantially affect the load speed regardless of the total number of bytes transferred. Consider that each individual request does have some overhead and that the network might not be quite as effectively utilized compared to a few larger image downloads. Remember, from the user's point of view, time counts—not bytes delivered—so wherever possible try to reduce the number of individual image pieces used.
- **Use the browser's cache.** Once an image has been downloaded once, it should stay in the browser's cache. If the same file is used later on, the browser should be able to reuse the one from the cache. If you can use scripting it might even be possible to download images ahead of time to the browser cache, using precaching or preloading. However, reliance on the cache only works if the complete filenames are the same. This means a single image directory probably is better than copying the files to individual image directories all over your site.
- **Give a preview.** If it is going to take a while to download, give the user something to look at. Interlacing a GIF image or making a JPEG progressive results in images that load incrementally. The user might get the gist of an image long before it completely downloads. Thumbnails of images also are a useful way to let a user take a look at the general idea of an image before committing to a long download. If a long download is required, it is a good idea to warn the user as well.
- **Use markup correctly.** Using **alt**, **height**, and **width** attributes can do a lot to improve page rendering. The alternative text will give the user something to read as an image loads. Setting the **height** and **width** values properly will allow the browser to specify the page layout, quickly allowing the text to flow in right away.

If you have to resort to large file sizes on your Web site, then the ends should justify the means. A big wait for a huge logo or heavily designed page with little content will result in frustrated users who never want to come back again. Could this be why the largest sites such as Amazon and Yahoo! use relatively simple visuals that download quickly? Almost certainly this is the case. In short, always remember when using images to make sure they add something to the overall experience of the user, whether it be to make the site more pleasing visually or provide information.

Summary

Inline images are truly what helped popularize the Web. However, just because images can be used to improve the look and feel of a Web page doesn't mean that they should be used without concern. Although presentation is important, the Web is still fundamentally about the communication of information, some of which does well in image form and some of which does not. Adding images to a Web page is accomplished using an **<img>** tag, which has numerous attributes. Many of the attributes of the **img** element—including **alt**, **height**, **width**, and **lowsrc**—are useful in improving the accessibility and usability of Web pages. As always, the eternal struggle between nice-looking pages and download time continues, and knowledge of markup features is helpful to combat excessive wait time. Many of the other attributes for the **img** element were developed with layout in mind, particularly **align**. However, layout and image formatting are truly better performed using style sheets.