4

Colors, Images, and Objects

In this chapter, you begin learning some aspects of Web design that will really breathe life into your Web pages. You'll start by learning how XHTML deals with color. You might remember from Chapter 1 that there were some odd six-character codes that represented colors, and in this chapter you learn what they mean so that you can use them wherever you have to deal with color. While we are looking at color, I will also introduce you to some key concepts behind creating colors and get you working with the color wheel to help you choose colors for your site. The authors of too many sites on the Web simply have not taken time to plan a good color scheme; and the result is that these sites look like they were built by amateurs (no matter how great the content). Learning a bit about color will really help make your pages look a lot more professional.

Next you will learn how to add images into your documents using the `<img>` element. You will see the difference between some of the main formats used for images on the Web and you will learn how to prepare your images for use on the Web. You will also learn how to make an image a link, and even how to divide an image up into sections so that different parts of the image link to different pages—this is known as an *image map*.

Finally you'll meet the `<object>` element that you can use to insert all manner of objects into pages, from MP3s and Flash movies to Active X controls and even images. In fact, the W3C sees the `<object>` element eventually replacing the `<img>` element, which is used to add images to your pages.

So, it's time to begin making your pages look a little more interesting, starting by adding color to them.

Adding Color to Your Site

When building almost any Web site, you will want to add color to your pages, and there are a lot of ways in which you can add color. As you will see later in the chapter, you can add color to your site through the use of images, but you are also likely to want to change the color of headings, paragraphs of text, page backgrounds, borders, and so on.

In Chapters 1 and 2 you've already come across some attributes that allow you to change the color of an element's contents, such as `color` and `bgcolor`. While attributes that specify color have been dropped from Strict XHTML 1.0 in favor of using CSS, they are still present in Transitional XHTML (although they are deprecated), and are used widely so you will more than likely come across them.

It is important to address color early on because color not only goes hand in hand with images but also has a very powerful effect on visitors, and creating a great color scheme is an important topic that deserves early attention.

The main issues you are going to look at in this section are therefore:

- How to specify which color you want
- Some key concepts behind creating color schemes
- How to choose a set of colors for your site
- Things that affect the appearance of colors
- Which colors and combinations of colors to avoid

Color is not the easiest topic to talk about in a book that is printed in black and white, but if you download the code examples for this book from `www.wrox.com` you get a lot more from this section—it contains color examples and color tools to help you along the way.

Specifying the Color You Want

The first thing you need to learn about color is how to specify exactly the color you want; after all there are a lot of different reds, greens, and blues, and it is important you choose the right ones.

In XHTML there are two key ways of specifying a color:

- **Hex codes:** A six-digit code representing the amount of red, green, and blue that make up the color, preceded by a pound or hash sign # (for example, `#333333`).
- **Color names:** A set of names that represent over 200 colors, such as `red`, `lightslategray`, and `fuchsia`.

Appendix D contains a list of over 100 color names and their corresponding hex codes.

Using Hex Codes to Specify Colors

When you start using *hexadecimal codes* (or hex codes for short), they can appear a little daunting. The idea that colors are represented by a mix of numbers and letters might seem a little strange, but what follows the # sign is actually the amount of red, green, and blue that make up the color. The format for hex codes is:

```
# rrggbb
```

The table that follows shows some examples.

Color	Hexadecimal Code
Black	#000000
White	#FFFFFF
Red	#FF0000
Green	#008000
Blue	#0000FF
Purple	#800080

As you might already know, computer monitors work in a color space known as an *RGB color space.* When a computer monitor is not switched on, the screen is black because it is not emitting any color. To create the image you see onscreen, each of the pixels that makes up the screen emits different amounts of the colors red, green, and blue just like a television screen.

It is hardly surprising, therefore, that you specify colors in amounts of red, green, and blue that are required to make a given color. The values of red, green, and blue required to make a color are between 0 and 255, so when red, green, and blue all have a value of 0 you get black, whereas if each has a value of 255 you get white.

You may have seen that some software represents colors using three sets of numbers between 0 and 255. Figure 4-1 shows the color window in Adobe Photoshop.

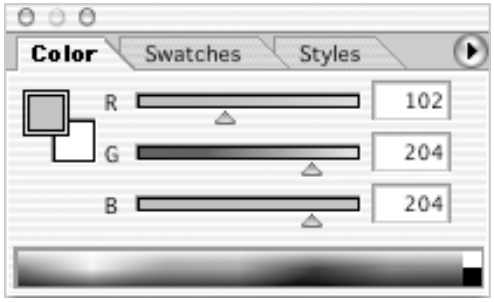

Figure 4-1

The hexadecimal codes used on the Web for color are a direct translation of these values between 0 and 255, except they use two characters not three to represent the numbers between 0 and 255. For example, FF represents 255 and 00 represents 0.

The best thing for really understanding how hex codes work is to take a quick look at how computers store information.

Understanding Hex Codes

You may have heard people say that computers store all their information in 0s and 1s, and while it may sound hard to believe, it's true! The smallest unit of information a computer stores in is known as a *bit*,

and a bit can only have one of two values:

- 0, which means off (or false)
- 1, which means on (or true)

These two values on their own will not store much information, yet if you combine four bits together you can get 16 different values. For example, using combinations of four 0s and 1s, you can represent the digits 0–9 (and still have values to spare):

```
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
  0    1    2    3    4    5    6    7    8    9    -    -    -    -    -    -
```

Four bits can be replaced by a single hexadecimal digit. There are 16 digits in hexadecimal numbers to represent the 16 possible values of four 0s and 1s:

```
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
0     1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
```

0 is the smallest; F is the largest.

Still, computers need to work with more than 16 possible values, so they tend to store information in yet larger segments. A group of 8 bits is known as a *byte*. A byte can therefore be represented using just two hexadecimal digits. For example:

```
Binary         0100     1111
Hexadecimal      4        F
```

This gives 256 possible combinations of 0s and 1s, plenty for the characters of the English language, and yes, that is why colors are represented in numbers between 0 and 255 come in.

So, while hexadecimal codes for Web colors may appear a little complicated, I think you would agree that `#4F4F4F` is a lot easier to read than `010011110100111101001111`. The following table shows some more hexadecimal codes and their corresponding decimal numbers.

Hexadecimal	Decimal
00	0
33	51
66	102
99	153
AA	170
BB	187
CC	204

Hexadecimal	Decimal
DD	221
EE	238
FF	255

Remember Appendix D holds a helpful table of over 200 colors for you to consult, and the download code features some handy color tools, too!

Using Color Names to Specify Colors

Rather than using hex values to specify colors, you can also use color names such as red, green, and white to specify the color you want. There are over 200 different color names supported by Netscape and IE, and there is a full list of these in Appendix D.

While names might sound a lot easier to understand than hex codes, some of the colors are easier to remember than others, and remembering which color each of the 200 names looks like is a tall order. Here is a sample of some of the color names:

```
aqua, beige, coral, darkcyan, firebrick, green, honeydew, indianred,
lavenderblush, maroon, navy, oldlace, palegreen, red, saddlebrown,
tan, white, yellow
```

Furthermore, if you do jobs for larger companies, such companies often want to specify very exact colors that represent their brand, and their color might not have an HTML name. Indeed, when clients specify the color they want, they usually specify a hex code.

> *Style guides are documents that larger companies often prepare when they employ other companies to perform marketing roles (such as advertising agencies and Web designers). Style guides can often be quite in-depth and contain brand-related information such as colors and fonts the company uses, photographic guidelines, and all manner of rules that govern how the company's brand is represented. The use of a strict style guide means that the client retains their unique look and feel even when several companies are producing marketing material for them.*

Hex Codes versus Color Names

It may seem as though color names are more straightforward to use than hex codes; if you use colors such as red, orange, green, blue, black, and white, then they are simple to remember and use. However, remembering each color name and the color it gives you is very difficult.

In practice, you often end up referring to a color chart to find the color you want, whether you're working with hex codes or color names. Given that hex codes give you a lot more choices of shades, tints, and hues of colors than color names, and bearing in mind that a lot of companies ask for specific colors to represent their company, hex codes tend to be the choice of Web professionals.

If you are using either a graphics program or a Web page authoring tool, that program will usually generate the color code you need for you, and many graphics packages also have a color-picking tool to help you select the exact color you want. Figure 4-2 shows the color picker from Photoshop.

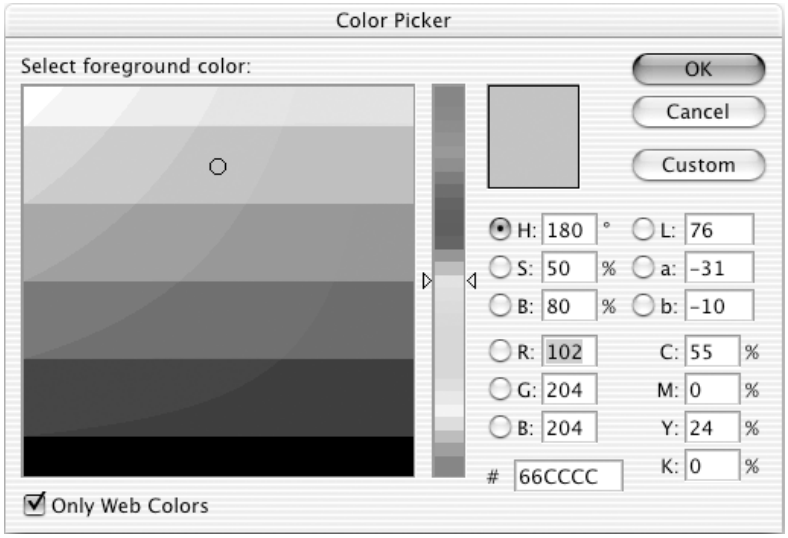

Figure 4-2

Note that the checkbox on the bottom left of this window indicates an option to use only Web safe colors. This is for a restricted color palette (containing a subset of all colors available) known as the Web Safe Color Palette, which you meet later in the chapter.

Choosing Your Colors for the Web

Now that you know how to specify a color, you need to think about choosing the colors for your site. If you look at the Web sites of any large international brands, you will see the colors you usually associate with that company. For example, you wouldn't expect the Coca Cola, Ferrari, or Virgin Web sites to be green. Big companies know the importance of using color to build a brand.

Color has a very strong influence on people and can affect how people view a company. So, when designing any Web site, color should be an important factor. A well-chosen color scheme is key to making any site attractive, and indeed well-chosen colors can mean the difference between an average looking site and a really attractive site.

In order to choose the colors you are going to use on the site you must have an understanding of the following:

- How colors are made up, and some key terms that describe colors
- What a color wheel is, and how it can help you choose your color schemes
- Different types of color scheme
- What the Web safe color palette is and why you can largely ignore it

- Factors that make colors look different on different computers
- Some issues regarding usability and color

While some of the following discussion of colors may be hard to follow in a black and white book, you will find color versions of many of the images used in this section in with the code examples for this chapter, which can be downloaded from `www.wrox.com`. Having these examples at hand while you read this section should make it easier to understand.

The Basics of Color

At school you probably learned that:

- There are three *primary colors*: red, yellow, and blue.
- Combining primary colors gives you *secondary colors* of green, orange, and purple.
- *Tertiary colors* are a combination of primary and secondary colors: red-orange, red-purple, blue-purple, blue-green, yellow-orange, and yellow green.

You now have a total of 12 colors. You can see these 12 colors in the color wheel illustrated in Figure 4-3. You may not be able to see all 12 colors clearly in this figure, but you can see a version on your own computer in `ch04_eg01.html` available with the code download for this chapter.

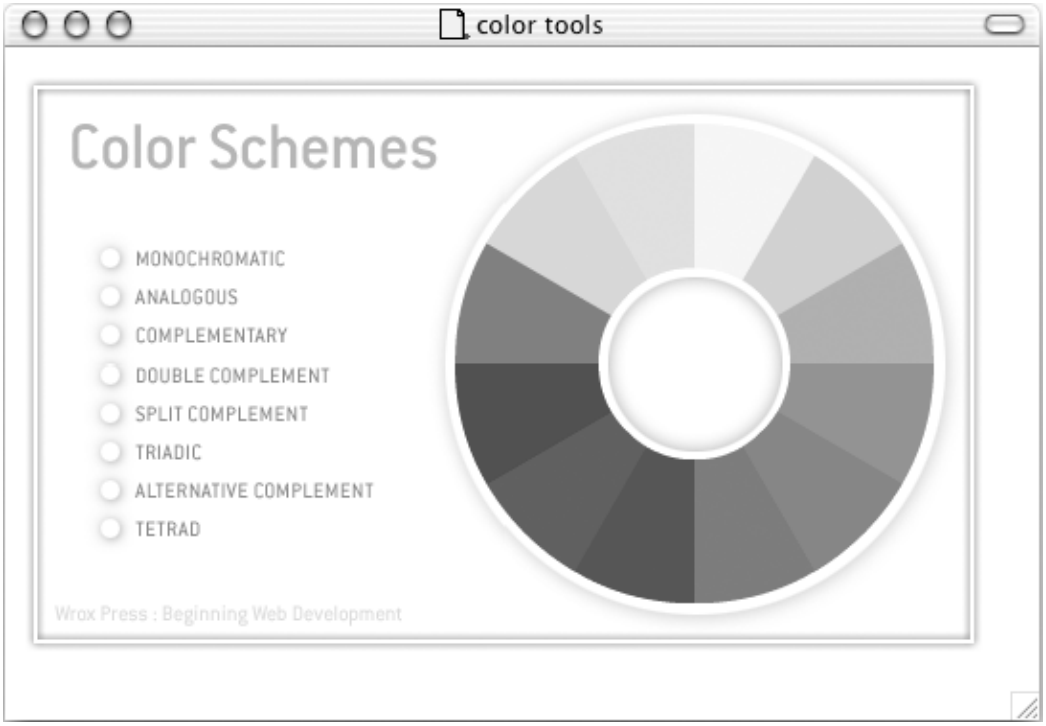

Figure 4-3

If you combine all of the colors in this basic color wheel you will end up with a more complex color wheel that shows every possible pure color (also known as a hue). Every color you use on the Web is either one of these pure colors, or what designers call a shade, tint, or tone of one of these colors (and seeing as designers like to use these terms it helps to know what each means):

- A *hue* is a pure color; it contains no black or white. It is the key part of a color that allows it to be identified as red, green, or blue.
- A *shade* is a hue with black added.
- A *tint* is a hue with white added.
- A *tone* is a hue with gray added.
- *Intensity* describes the intensity of a color, sometimes the terms *saturation* or *chromaticity* are used instead. In a hex code, for example, a blue with `#0000FF` would be an intense blue. Gray, meanwhile, has no identifiable hue, and is called *achromatic*.

You can see images that illustrate each of these color terms in the download code for this chapter (`ch04_eg02.html`); they will really help you understand these terms.

In order to specify these colors for the Web, you have seen that you can use hex codes. These indicate the amount of red, green, and blue required to make up that color. Before you turn it on, a computer screen is black. When you start to use your computer, however, color is created on the screen by emitting different amounts of red, green, and blue light.

You should be aware, however, that there are other ways to specify colors. For example, print designers usually specify colors using combinations of cyan, magenta, yellow, and black—often shortened to *CMYK*. (The *K* is used instead of black. If *B* were used for black people might think the **B** stood for blue.) Print designers specify colors this way because a plain sheet of paper is white (and it is white because it reflects all colors); then when you add cyan, magenta, or yellow ink to the page it subtracts from the light reflected, giving the paper color. Furthermore, black is present because cyan, magenta, and yellow together do not create black.

These two different ways of representing colors are known as *color spaces*. But why is this important? Consider the following:

- Most graphics programs will give you the option of preparing images in one of these two color spaces; you should prepare your graphics in RGB mode when creating work that will be displayed on screens and CMYK mode for print work.
- If you have worked in print, or moved from the Web to do a print job, you should be aware that CMYK has a smaller color space than RGB. Some of the colors in the RGB color space cannot be printed using CMYK because of the limitation of inks (your graphics program should indicate to you if a color cannot be printed).

Creating a Color Scheme

Now that you understand some of the key terms regarding color and how a color is specified, it is time to look at creating a color scheme—a selection of colors you will use in the design of your site. It is a good idea to start on this early on in the stages of planning your site because randomly selecting colors or just

picking a few of your favorite colors will probably result in your site looking a lot less professional. Once you have decided upon your color scheme, it is then a lot easier to design the pages using those colors.

Your color scheme may be as simple as using black and white with one color to accent it. Or you may choose a complicated set of up to three to four colors to use with black and white. You should, however, avoid too many colors.

Some of the most popular color schemes are created by first choosing a key color and then selecting other colors based on their position in relation to the key color on the color wheel. For example, you might choose a color that is directly opposite your key color on the color wheel. The following descriptions of color schemes can give you a good idea of how to choose the colors for your site. To really see how the colors work together you need to look at the example code for this chapter (available as a download from `www.wrox.com`) as it will show you the actual colors (these are example `ch04_eg03.html`).

- A *monochromatic* color scheme uses one base color and then adds shades and tints of that color.
- An *analogous* color scheme selects colors adjacent to each other on the color wheel.
- A *complementary* color scheme uses colors on the opposite side of the wheel from one another. You should, however, avoid using one of these colors on top of the other.
- A *double complement* takes two sets of complementary colors.
- A *split complement* takes a color and the two colors adjacent to its complement.
- A *triadic* color scheme selects three colors from points of an equilateral triangle within a color wheel.
- An *alternate complement* adds another color from between two points of the triangle

If you are not given a set of colors to work with when creating a site, a good place to start is by picking a color scheme based upon the colors used in the logo. Another good backup is taking the main hues from any featured photographs or images used on the site. Failing that, open up the color wheels in the example code and spin them around until you find some colors you like.

> *If you want to find out the latest fashionable colors, fashion magazines and paint manufacturers' recent brochures tend to push the most popular colors of the day. However, choosing the latest "in" colors is not always a good solution for choosing brand colors because what is in fashion one day is out of fashion the next.*

Contrast

Whichever colors you choose, you must ensure that there is sufficient contrast in the colors that readers can make out the text and images on your site. High contrast makes text easier to read. If there is not a lot of contrast between the text and background, users will struggle reading the text.

Black text on a white page is probably the most popular color scheme for documents we read each day; the contrast between them is high. If you look in the download folder for the code, you will see some examples of contrasting colors (`ch04_eg04.html`).

You should also be aware that your eyes can often play tricks on you when it comes to color; here are a couple points you need to be aware of:

- Complementary colors are sometimes known as *discordant colors* because they can play funny tricks on the eye, such as making one of the colors seem like it is moving. You should avoid using

a discordant color on top of another one unless you are deliberately going after that effect. You can see an example of discordant colors in `ch04_eg05.html` with the download code for the chapter.

- A dark block of color with a slightly lighter block on top of it will make the lighter color seem lighter than it really is. You can see an example of this in `ch04_eg06.html` with the download code for the chapter.

Other Things That Affect Color

You should be warned that, once you have chosen your color scheme and come to look at the colors on a different screen or in a different program, you might find that the colors look very different. So, when you have decided upon your color scheme, it pays to test it out on a few different computers and in a few different programs to make sure you are still happy with it.

There are a lot of reasons why colors for a site will look different on one computer than the next:

- PCs and Macs have different gamma correction, which is a modification to the color saturation and brightness on your monitor. Hence, colors on Macs tend to appear lighter than they do on PCs.
- The color in LCD screens can vary when looked at from different angles, whereas CRT screens tend to have more even colors from different angles. Also, near-white colors can merge into each other more on CRT screens.
- Different software can render colors slightly differently.

So, don't be alarmed if your colors look slightly different in different situations, but if possible check your colors on a different setup to make sure they still work as you intended. (In particular, you may find grays, beiges, and greens look very different on a PC than on a Mac.) This does not mean you have to own different computers; you might be able to check them on a friend's computer or at your local library or Internet cafe.

Also, when creating your site and choosing colors in graphics programs, use the hex numbers you defined in your color scheme—do not just assume a color "looks right."

The Web Safe Color Palette

I should say a few words about something you may have heard of: the *Web safe color palette*. This is a selection of 216 colors that is sometimes said to be "safe" to use on the Web.

The Web safe color palette was actually devised back in 1994 for Netscape as a way to reduce problems of colors looking different on different systems. Back then a lot of computers used 8-bit technology, which (as you might remember from the introduction of hex codes) supports only 256 colors. Forty of those colors were reserved for use by the operating system, which meant that you had 216 colors left.

If a computer could not display a block of color it would try to recreate the color by having two alternating colors on pixels next to each other. This is a process known as *dithering*, and the resulting colors are not accurately replicated on different systems. Therefore, the 216 colors not used by the operating system became known as the Web safe color palette.

For a number of years it was suggested that Web designers pick their color schemes for Web sites from the 216 colors in this Web safe color palette. However, nowadays, statistics suggest that fewer than 5 percent of Web users still use 8-bit graphics cards, so you are quite safe to ignore this restriction.

> *When 16-bit graphics cards came along, they were the newest, greatest thing! However, these cards did not use a subset of the 24-bit true color palette. There are therefore slight differences in 194 of the supposed 216 "Web safe" colors. This resulted in some people claiming that there were only 22 truly Web safe colors that looked the same on each system. But such a restriction is extreme and is again safe to ignore.*

In reality, the shift in colors will not affect your work too much, especially if you use blocks of colors. The only two pitfalls you really need to be aware of are:

- If the background color of your page will meet a flat area of color in a GIF image (for example the image features a flat orange—not textured—and your background is orange), some users might see a slight color shift between the two. It is better to make the GIF stretch the area you want to cover, or make a border.
- When designing a site, make sure that there is enough contrast between colors to make the text readable even if there is a slight shift in colors. For example, black against white is a good, safe contrast; yellow on orange is not, as already discussed in the *Contrast* section.

Final Words on Color

Color is a very powerful tool in the right hands, but can really turn visitors away when not given due care and attention.

What you have seen in this chapter simply scratches the surface of color theory, which in itself is a huge topic. You don't need to be an expert in color theory to make your site look good; you just need to think about color before you start designing the site.

If you have not been given a set of colors to work with, or you have been given only one color you need to work with (and told to use what other colors you like), take a look at those color wheels, which should prove a great help. And remember, you can always use different shades, tints, and tones of the colors you choose as accents. But try not to use every color in the rainbow and don't be afraid to ask your friend with the impeccable taste what he or she thinks of the colors before building your whole site in them.

Another interesting area of color that we do not have space to go into here is color psychology. Certain colors elicit different emotions, expectations, and reactions in your visitors. If you have found this section interesting, you will probably be fascinated to learn more about how we interpret colors.

Adding Images to Your Site

Images and graphics can really bring your site to life. In this section you will not only learn how to insert images and graphics into your pages, but also which image formats you can use and when you should choose which format.

You will see how careful you have to be when using images on the Web, as they can really slow down the speed of a site—and slow sites frustrate users. When you are writing a Web site and all of the images are

on your computer, pages will appear to load very quickly. But as soon as you put your site on the Web, it will often slow down dramatically. So, choosing the right format for your images and saving them correctly will help make your site faster, and result in happier visitors. After all, even though many people access the Internet on fast connections there is still a large proportion of users who use dial-up modems.

> *For practice purposes, you can download images from other peoples sites by right-clicking the image (or Ctrl-clicking) and selecting either the* `download image to disk` *or* `save image as` *options. Remember, however, that images are subject to copyright, and you could land yourself in legal trouble if you use other peoples' images on your site.*

Now that you've learned how to insert the right kind of images into your pages, you will then see how to turn them into links and even how to write code that divides them up so that when users click different parts of the image they get taken to different Web pages.

Types of Image Formats

To start off, you should look at how computers store and render pictures. There are two main ways in which graphics are created for computers:

- *Bitmapped graphics* divide a picture into a grid of pixels and specify the color of each pixel, rather like a computer tells a screen the color of each pixel. Broadly speaking, bitmaps are ideal for photographs and complicated graduations of shade and color. There are several different formats of bitmap; common ones include (the rather confusingly named) bitmap or BMP, JPEG, GIF, TIFF, and PNG. You will be learning more about JPEGs, GIFs, and PNGs later in the chapter.
- *Vector graphics* break the image into lines and shapes (like a wireframe drawing), and store the lines as coordinates. They then fill the spaces between the lines with color. Vector graphics are commonly used for line art, illustration, and animation. They often feature large areas of flat color (as opposed to textures, shades of colors, and photographic styles).

For a long time, bitmaps were the main image format for the Web, although more recently there are some formats on the Web that are making use of vector graphics, such as Flash and SVG.

Bitmap Images

Most static images on the Web will be bitmapped images. Browsers tend to support three common bitmap graphics formats, and most graphics programs will save images in these formats:

- **GIF:** Graphics Interchange format (pronounced either "gif" or "jif")
- **JPEG:** Joint Photographic Experts Group format (pronounced "jay peg")
- **PNG:** Portable Network Graphics (pronounced "ping" or "pee en gee")

We will take a quick look at each of these.

GIF Images

The GIF (or Graphics Interchange Format) used to be the standard for all Web graphics. GIF is an *indexed color format* that allows an image to select up to 256 colors from a range of over 16 million colors to recreate the picture; each pixel is one of these 256 colors. The file includes a lookup table and the pixels

reference the detailed color information in the lookup table rather than each pixel having to specify this information. Figure 4-4 shows a GIF file and its color palette in Adobe Photoshop:

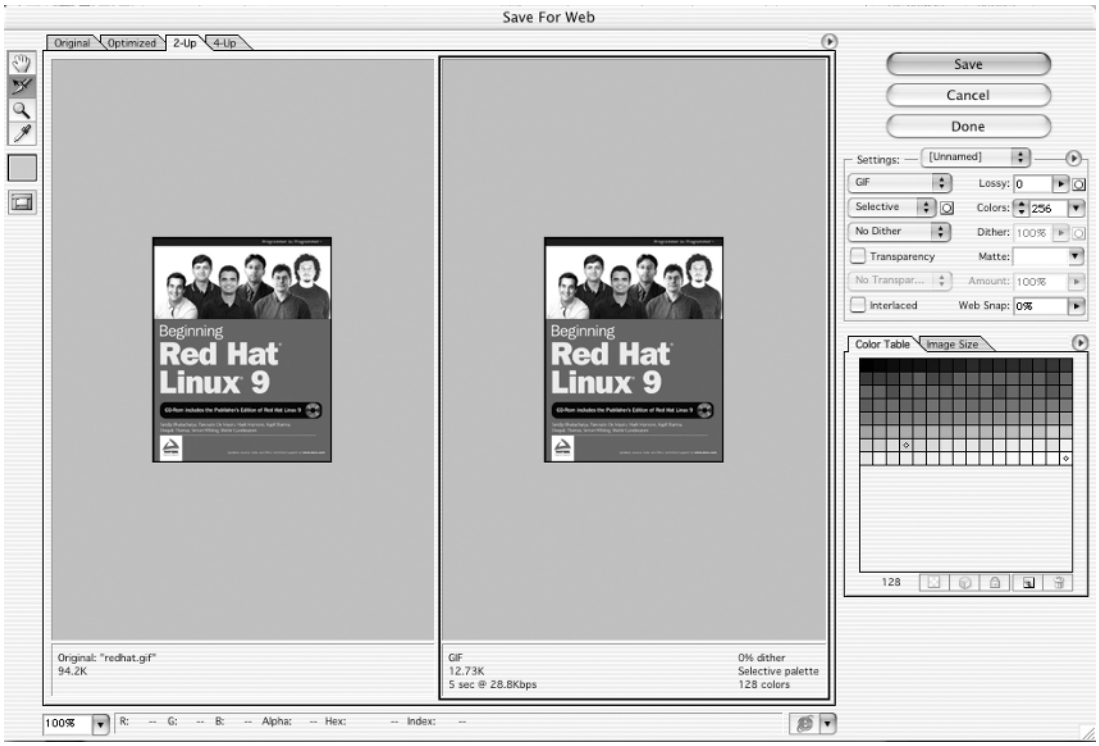

Figure 4-4

GIF images are particularly suited to images where there are large flat areas of color. A flat area of color is a section that is just one shade, for example a rectangle that is one green is a flat color whereas a picture of grass contains lots of different greens. The fewer colors the image uses, the smaller the GIF file is. A GIF containing less than 16 colors, also known as a 4-bit GIF, will take less than half the space of an 8-bit GIF using 256 colors. Therefore, if you are creating an image that uses less than 16 colors it is worth checking whether your program will automatically save your image as a 4-bit GIF because it would result in a smaller file that is quicker to download.

> *Note that if your text or lines are two colors (say black and white) and you have used anti-aliased edges to make them look smoother, your image will contain more than two colors because the edges use a variety of other colors to make them look smooth.*

Most graphics programs, when saving GIFs, will use a technique called *dithering* if they need to represent more than 256 colors. This means they use two or more colors in adjacent pixels to create an effect of a third color. Dithering has two drawbacks:

- It can result in some *banding* in colors that might appear flat to the eye in the original, but are actually very slightly different shades; this means that the colors that appeared the same color no longer have a smooth transition and changes in the shading become visible.

- If you place a flat color next to a dithered color you will be able to see where the change occurs (because the dithered color is really made up of more than one color).

Because GIFs support only 256 colors and have to use dithering to achieve any further colors, they are not really suitable for detailed photographs, which tend to contain more than 256 colors. If you have a photograph, gradient, or any image with similar shades of the same color next to each other, you are often better off using a JPEG, which can support unlimited colors, or sometimes a PNG—both of which you will meet shortly.

GIFs do have another handy feature: you can specify one color in a GIF to represent a *transparent background*—in parts of the image that are the specified color, the background will be allowed to show through. You should be aware, however, that each pixel is either on or off, opaque or transparent—there are not degrees of transparency, as there are in alpha-color transparency formats. This means that if you try to use it with curved corners, the corners may appear pixelated. To help overcome this problem you should try to make the transparency color as close to the background color as possible.

Figure 4-5 shows how a pixelated effect is created when a GIF is not created on a suitable background (notice the corners in particular).

Figure 4-5

To make the GIF files smaller, they are compressed using a technique called *LZW compression* that scans rows of the image looking for consecutive pixels that share the same color, and when it comes across them it indicates that x number of pixels should be written from this point onwards using the same color.

LZW compression is known as a *lossless compression* technique because no data is lost and therefore there is no loss of quality (this is contrasted with *lossy compression* techniques where some of the data is discarded during compression and cannot therefore be recovered from the compressed file). However, when there are not many consecutive pixels of the same color, there is little saving in file size. This means that the format does not compress photographic images well because while the adjacent pixels may look the same in photographs, they tend to be very slightly different. Furthermore, if the picture uses complex dithering to achieve subtle coloring effects there is less chance of finding pixels of the same consecutive color, and therefore file size cannot be compressed.

Some programs will give you an option of saving the file as an *interlaced image*. *Interlacing* means that lines of the image are stored in a different order than they would appear in the image. This allows a browser to display every eighth row in turn and then fill in the lines between. The idea behind interlaced images was that, if you had a large file on a slow connection, the user would see something appearing early on, and the image would get progressively clearer. This can help users when the image is easily recognizable or for logos, but it is not ideal for images or text with a lot of detail. Interlaced GIFs are becoming less widely used on the Web as bandwidth rises.

Animated GIFs

GIF images can store more than one frame (or copy of the image) within a file, allowing the GIF to rotate between the versions/frames and create a simple animation. If you think of how a flip-book animation works, each page has some movement, and when a user flips the pages it looks like the images are moving. Compression for this technique is quite effective because only the changed pixels need storing with each frame, along with their positions.

You should be very careful about the use of animated GIFs. A lot of sites offer animated GIFs, from cartoon characters doing something amusing to bouncing or flaming bullet points. Sure, they might be impressive or fun the first time you see a page, but they soon become tiresome, slow down the site, and distract users from the real content. While animated GIFs can be fun on a personal home page, you will not find such animations on the sites of large companies. If you are trying to create a professional looking site, you should use animated GIFs only if the animation gives additional information to the user.

JPEG Images

The JPEG image format was developed as a standard for storing and compressing images such as photographs with wide ranges of colors. When you save a JPEG, you can usually specify by how much, if at all, you want to compress an image—which depends upon the image quality you want. The process of compressing a JPEG involves discarding color data that people would not normally perceive, such as small color changes. However, because the image format discards this data when the image is compressed, some of the data is lost and the original cannot be recreated from a compressed version—hence it is known as *lossy compression*.

The amount of compression you apply will change from image to image, and you can judge how much to compress a JPEG by only by looking at it. Hence the size of the file varies depending upon how much you compress the image. When you are saving the image, you will often be asked for a percentage of quality to be used; 100 percent does not compress the picture at all, and around 60 percent is the most you can use for a photo (some programs use words such as excellent, very good, good, and so on instead).

A good image-editing program will allow you to compare the original image side by side with the compressed version as you choose how much compression to add. Figure 4-6 shows you how Adobe Photoshop lets you compare two versions of the image next to each other as you prepare to save the JPEG for the Web.

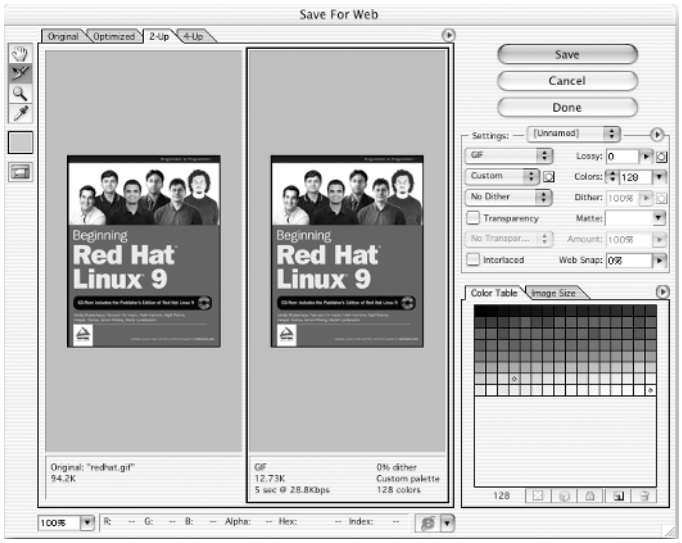

Figure 4-6

Because the JPEG format was designed to work with photo-realistic images, they do not work so well with images that have large amounts of flat color, or high-contrast hard edges (such as lettering and line drawings). As you increase compression in a JPEG you may also see banding start to show in colors that are very similar.

JPEG does support interlacing using the *Progressive JPEG*, allowing an initially blocky view of the image to download first, with greater detail being filled in as the rest of the image loads. The most helpful aspect of this is that it gives the user an idea of the size of the image that is being downloaded, and a rough idea of how complete it is. Because JPEGs tend to have a lot of detail, however, you often need a lot of the image to come through before you really get to see the intended picture.

PNG Images

The Portable Network Graphics format is the most recent format on the block. It was developed in the late 1990s because the company that owns the patent for GIFs (Unisys) decided to charge companies that developed software for creating and viewing GIFs a license fee to use the technology. While Web designers and Web surfers are not affected by this charge, the companies that make the software they use are.

The PNG format was designed for the same uses as GIF images, but while it was being created the designers decided to solve what they thought were some of the disadvantages with the GIF format. The result is two types of PNG. The 8-bit PNG has the same limitations as an 8-bit GIF—only 256 colors, and when transparency is used each pixel is either on or off. Then there is the enhanced PNG-24, a 24-bit version, which has the following advantages:

- The number of colors available for use in an image is not restricted, and so any color can be included without losing any data.
- A map (like the color map that indicates the color of each pixel in GIFs) is used to provide different levels of transparency for every pixel, which allows for softer, anti-aliased edges.
- The approach of sampling one in eight lines was replaced with a two-dimensional sample of pixels displaying a rough image using just $\frac{1}{64}$ of the file (whereas the GIF required $\frac{1}{8}$).
- PNG 24-bit files can contain gamma correction information to allow for slight differences in color between different monitors and platforms.

Furthermore, all PNGs tend to compress better than a GIF equivalent. The real drawback with the PNG format, however, is that support is limited. While the browser manufacturers introduced it into version 3 and 4 browsers, some of the more advanced features (such as transparency) still had problems in IE 5. As a result, most designers still use the GIF format.

Keeping File Sizes Small

You will often save the images for your site in the format that best compresses the image and therefore results in a smaller file size. This will not only make your pages quicker to load, but can also save you on the charges made for hosting your site.

Usually one or another format will be the obvious choice for you. The rule of thumb is:

- Use JPEGs for photo-realistic pictures with a lot of detail, or subtle shade differences you want to preserve.
- Use GIFs for images with flat color (rather than textured colors), and hard edges, such as diagrams, text, or logos.

You can also consider using PNGs if you do not need the advanced features such as transparency, or if you know the majority of your visitors will be using version 6 browsers.

If you look at the following images—one a photo of a forest, and the second, the logo of a fictional company called Color Wheels that uses only three colors—you can see the file size of each saved as a GIF and as a JPEG.

As you can see, the Color Wheels logo has areas of flat, plain color, whereas the photo of the forest uses lots of different shades. Therefore the logo is better suited to the GIF or PNG formats, while the photo of the forest with all its shadows is suited better to the JPEG format.

You should also make sure that your image resolution is no greater than 72 dots per inch (dpi) because this is the maximum resolution of computer monitors. While print designers usually work with images that are 300dpi, anything above 72 is not shown on computer screens.

Figure 4-7

Good image editing software is very helpful if you use a lot of images on your site. Adobe Photoshop is the most popular software used by professionals, although unfortunately it is very expensive. There is, however, a limited functionality version called Photoshop Elements that includes many of the common features—including the Save for Web options. Another popular image editing tool is Paint Shop Pro.

If you do have to include large complex photographic images on your site, it's good practice to offer users smaller versions first and then add a link to the larger version if it interests them. These smaller images are often referred to as *thumbnails*, and you will usually see them in image galleries or pages that contain summaries of information (such as the home pages of news sites and pages that list several products).

When creating the smaller version, scale the image down in an image-editing program; do not just alter the `width` *and* `height` *attributes of the* `<img/>` *or* `<object>` *elements you are about to meet or users will still have to download the full-sized image even though they are getting to see only a smaller version of it (which will take much longer to download). By creating a special thumbnail of any smaller images you use, your pages will load a lot quicker.*

Vector Images

Most illustration and animation software uses vector formats; for example, Macromedia Freehand and Adobe Illustrator both save images as vectors. As far as the Web goes, the champion of vector graphics is Macromedia Flash (which you will see on a lot of sites).

Because vector formats store information in terms of coordinates for lines and then colors inside those lines, it is very easy for vector formats to scale to different sizes. This has great potential for the Web, as it allows images to work with different resolution devices. Therefore you can expect to see an increasing adoption of vector image formats on the Web. Also, because vector technology is widely used for computer games, increasing amounts of work are being done on vector formats.

Browsers and XHTML do not, by default, support any vector graphics formats, although the main browsers now ship with the Macromedia Flash Player that is required to view Flash files, and as a result Flash is currently the most popular way of deploying vector graphics and animations on the Web. While the Macromedia Flash Player is free for download, and the browsers feature it, you should be aware that Macromedia charges for the software to create Flash files and doing so does require that you learn a whole new skill (which is outside the scope of this book).

As an alternative vector graphics format the W3C developed Scalable Vector Graphics (SVG), which (like XHTML) is written in XML, and means the two will be able to be integrated easily (furthermore it is an open standard, not the creation of an individual company like Macromedia Flash is). There are a number of tools that have recently emerged, and that are in development to support this format, and some of the latest browsers are supporting it, so you can expect it to grow significantly in the coming years.

Both Flash and SVG files tend to be included in pages using the more recent `<object>` element. (Indeed, the W3C would like to see all images included using this element in the long run, but for the moment images are more often added using the `<img>` element.)

Adding Images Using the <img> Element

Images are usually added to a site using the `<img>` element. It must also carry the `src` attribute indicating the source of the image and an `alt` attribute whose value is an alternate description for the image in case it does not load or the user has a visual impairment.

For example, the following line would add a logo to the page. In this case the image is in the same folder as the file, and is called `logo.gif`:

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

In addition to carrying all of the universal attributes and the UI event attributes, the `<img>` element can carry the following attributes

```
src alt align border height width hspace vspace ismap usemap longdesc name
```

The src Attribute

The `src` attribute is required to specify the URL of the image to load.

```
src="url"
```

The URL can be an absolute URL or a relative, just like the URLs when linking between pages that you met in Chapter 3, and use the same shorthand notations to indicate which folder an image is in.

It's a good idea to create a separate directory (or folder) in your Web site for images. If you have a very large site, with separate directories for each section, you can create an image folder for each section of the site.

> *Generally speaking, images for your site should always reside on your server. It is not good practice to link to images on other sites because if the owner of the other site decides to move that image your users will no longer be able to see the image on your site.*

The alt Attribute

The `alt` attribute is required to specify a text alternative for the image in case the browser cannot display the image (for any of a number of reasons). For example:

```
alt="Wrox logo"
```

Often referred to as *alt text*, the value of this attribute should really describe the image for users who cannot see it—either because the browser did not download the file correctly because the file cannot be found, or because the user has visual impairment that prevents him or her from seeing the image. The alt text should not just be the same as the filename. If you are using images for buttons then the button's alt text should usually be the same as what the button reads and describe what will happen if the user presses the button.

If your image is just used for layout and is not strictly visible (for example a block of color used to help position another element rather than something the user is supposed to see), then the `alt` attribute should still be used but given no value, as follows:

```
alt=""
```

The align Attribute (deprecated)

The `align` attribute is used to align the image within the page or the element that contains the image (such as a table cell).

```
align="right"
```

It can take one of the values in the table that follows.

Value	Purpose
`top`	The top of the image is aligned with top of the current line of text.
`middle`	The middle of the image is aligned with the current text baseline.
`bottom`	The bottom of the image is aligned with the baseline of the current line of text (the default), which usually results in images rising above the text.
`left`	The image is aligned to the left side of the containing window or element and any text flows around it.
`right`	The image is aligned to the right side of the containing window or element and any text flows around it.

You may come across the `absbottom`, `texttop`, `absmiddle`, and `baseline` values, but these are nonstandard extensions that can produce inconsistent results.

The border Attribute (deprecated)

The `border` attribute specifies the width of the border around the image in pixels:

```
border="2"
```

The default is value is `0`, and if the attribute is not used there will not be a border unless the image is used as a link in which case you should specify `border="0"` (see the "Using Images as Links" section later in this chapter). This attribute has been replaced by the CSS `border` property.

The height and width Attributes

The `height` and `width` attributes specify the height and width of the image:

```
height="120" width="180"
```

The values can either be pixels as shown in the preceding code or a percentage of the page or containing element (in which case it will be followed with the percent sign).

Specifying the size of the image can help browsers lay out pages faster because they can allocate the correct amount of space to the image and continue to render the rest of the page before the image has finished loading.

If you want to scale an image, you can just provide the value for one of the attributes and the browser will maintain the correct ratio for the image (its width compared to the height). You can even distort images by providing a different width in relation to its height.

Figure 4-8 shows an image at its actual size (top: 100 pixels by 100 pixels), the image magnified (middle: the `width` attribute is given a value of 150 pixels), and the image distorted (bottom: the width attribute is given a value of 75 pixels and the `height` attribute a value of 125 pixels).

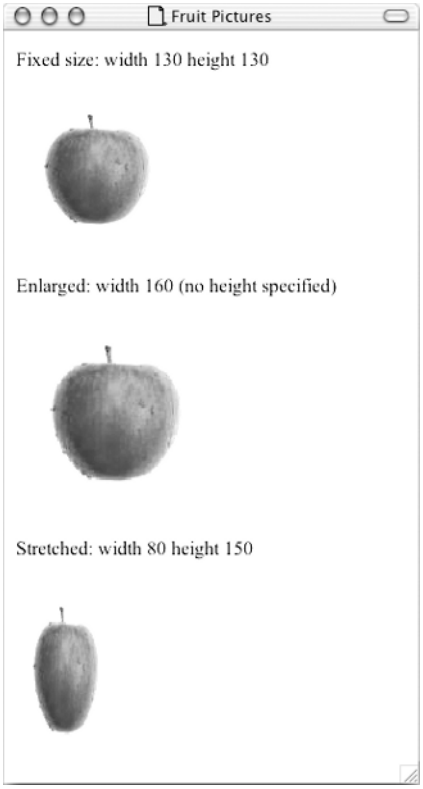

Figure 4-8

Here is the code for this example (`ch04_eg07.html`):

```
<p>Fixed size: width 130 height 130</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="130"
height="130" />
<p>Enlarged: width 160 (no height specified)</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="160" />
<p>Stretched: width 80 height 150</p>
<img src="images/apple.jpg" alt="Photo of red apple" width="80" height="150"/>
```

If you want to display the image a lot smaller than the original version rather than just specifying the smaller dimensions for the same image, you should resize the image in an image manipulation program to create the smaller version for use on the Web site. If you just reduce the size of the image using the `height` and `width` attributes the user will still have to download the full sized image, which will take longer than a special small version and use up more bandwidth.

The hspace and vspace Attributes (deprecated)

The `hspace` and `vspace` attributes are used to control the amount of whitespace around an image.

```
hspace="10"
vspace="14"
```

The value is the amount in pixels of whitespace that should be left around the image, and is like a white border. The `hspace` and `vspace` attributes are particularly helpful because text can flow around an image, and unless there is a gap between the text and the image, the text becomes hard to read and doesn't look as professional. Figure 4-9 illustrates this idea.

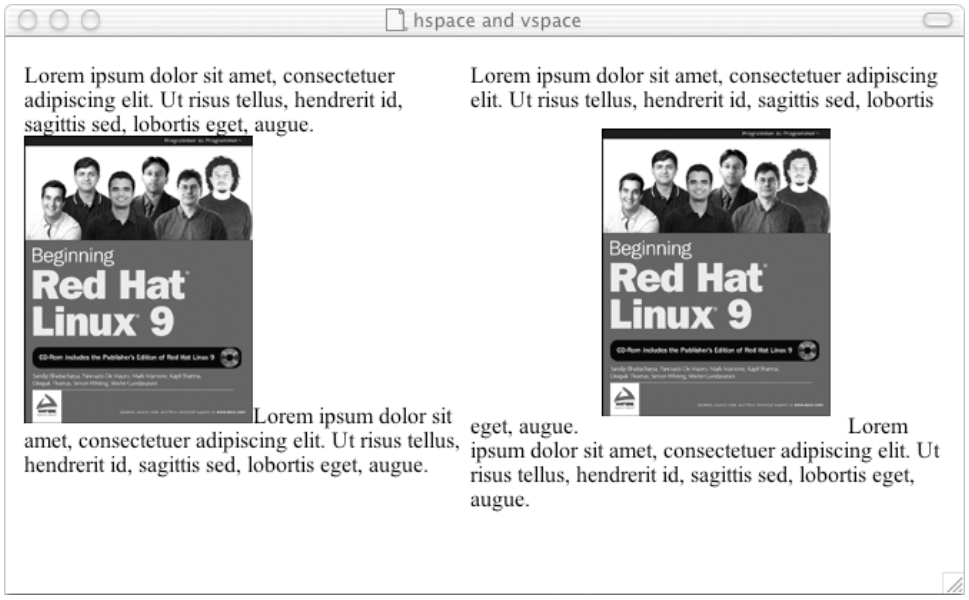

Figure 4-9

These attributes have been deprecated, and you can achieve the same result by using the `border` or `margin` properties in CSS.

The ismap and usemap Attributes

The `ismap` and `usemap` attributes are used with image maps. Image maps are covered in the "Image Maps" section later in the chapter.

The longdesc Attribute

The `longdesc` attribute is used to indicate the URL of a document containing a description for the image in more detail.

```
longdesc="../accessibility/profit_graphs.txt"
```

It is designed for users who cannot see the image, and for providing extra information that cannot be seen in the image. It would also be particularly helpful for providing explanations for things such as graphs and charts. Netscape 7 and IE 6 still do not support this attribute.

The name Attribute (deprecated)

The `name` attribute allows you to specify a name for the image so that it can then be referenced from script code. It is the predecessor to, and has been replaced by, the `id` attribute.

```
name="image_name"
```

Try It Out Adding Images to a Document

In this example you're going to add some images to a document; they will be some brightly colored images of food accompanied by a description of each. So, open up your text editor or Web page authoring tool and follow these steps:

1. Start with the `XML` and `DOCTYPE` declarations and add the skeleton of the XHTML document, like so:

```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
  <title>Fruit Pictures</title>
</head>

<body>
</body>
</html>
```

2. Add the following to the body of the page. Pay particular attention to the `<img>` elements:

```
<h1>The Fruit Pictures Page</h1>
<p>The first image is an image of an apple.</p>
```

```
<img src="images/apple.jpg" alt="Photo of red apple" width="130" height=
"130" />
<p>The second image is an image of an orange cut in half.</p>
<img src="images/orange.jpg" alt="Photo of orange" width="130" height="130" />
<p>The third image shows a group of bananas.</p>
<img src="images/banana.jpg" alt="Photo of bananas" width="130" height="130"/>
```

3. Save the file as `fruit.html` and open it in your browser. You should end up with something that looks like Figure 4-10.

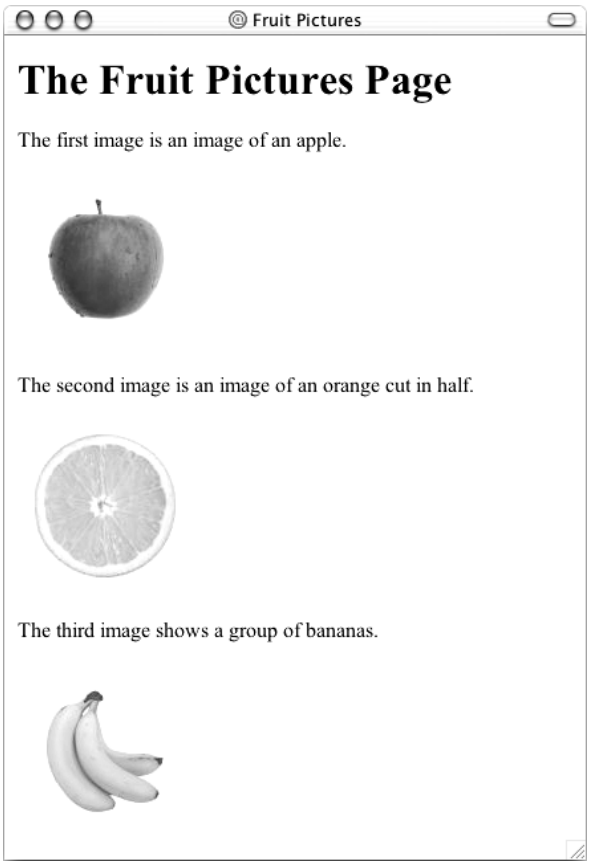

Figure 4-10

How It Works

You have met most of this code enough times already. But the parts to concentrate on are the `<img />` elements. Each `<img />` element adds a new image. There are three in the example.

```
<img src="images/apple.jpg" alt="Photo of red apple" width="130" height="130"
/>
```

The `src` attribute indicates the URL for the image. The URLs in this example are all relative URLs pointing to an `images` directory that is contained in the same directory as the example page. You might remember from Chapter 3 that I said that organizing your file structure was very important—you can here why this is the case.

The `alt` attribute should be used on every `<img />` element you write. It will be shown if the browser cannot load the image and tells those who have vision impairments what the image is of.

The `width` and `height` attributes tell the browser how big the image should be displayed. By including these attributes the browser can lay out the page quicker because it can continue to display other items on the page without waiting for the image to download. While you can use these attributes to stretch or scale up an image, if you want to make the image smaller, you should save a new version of it rather than just using these attributes to save your viewers' time and bandwidth.

Adding Other Objects with the <object> Element

The W3C introduced the `<object>` element into HTML 4 with the intention that it be used to embed all media types into documents, not just graphics, but also MP3 files, Flash movies, QuickTime movies, JavaScript objects, Java applets, and so on. Note that, with formats that require a browser plug-in, you have to specify the program you expect to run files (for example the Macromedia Flash Player, Windows Media Player, the QuickTime Active X control, and so on).

The support for the `<object>` element in IE has been quite good for a while, except when it comes to including images, but Netscape really got up to speed only with the later releases of Netscape 6.1 (from late 2001 onwards).

> *The `<object>` element was initially introduced by Microsoft for support of their Active X technology (sometimes referred to as* ActiveX plug-ins*) so you may see older pages use it for this purpose.*

Before the `<object>` element was introduced there was a range of elements browsers used to insert multimedia objects into pages, such as the `<applet>`, `<embed>` and `<bgsound>` elements, but these elements have been deprecated (they are covered in Chapter 13).

Because `<object>` elements often require an application such as a plug-in to be installed on the user's computer, you cannot be guaranteed every visitor to your site will be able to play the file. Furthermore, when it comes to large audio or movie files, you should give the visitor the option to download these large files (that can take a long time to download) if they want to. If you do not enable visitors to choose whether they download these files they will often leave a site.

The most common way of embedding moving graphics into Web pages without asking the user first is by using Macromedia Flash, which is not only installed on most computers but also uses vector graphics to create animations so it compresses them well (resulting in smaller file sizes). However, when Flash is used to add music to a page, it is widely considered good practice to offer a button to turn the music off.

> *If you are using Flash, you will most likely use the publishing tool (the Publish item on the File menu) to generate the code to insert a Flash movie into your page. Note, however, that Flash does not always generate XHTML code; it can use older HTML syntax, so you might need to add trailing slashes to the empty `<param / >` elements.*

To embed an object into a page, you need to specify:

- The location of the code used to display or play the object (sometimes referred to as the *implementation* of the object)
- The actual data to be rendered
- Any additional values the object needs at run time

The first two are added using the `<object>` element while additional values are provided in the `<param>` element, which can be a child of the `<object>` element.

While the `<object>` element can contain a child `<param>` element, any other content of the `<object>` element should be displayed only if the browser cannot render the object:

```
<object>

Your browser does not appear to support the format used in this film clip,
for more details please look <a href="../help/video.htm">here</a>

</object>
```

You can nest `<object>` elements in order of preference in which they are viewed, so you can put an alternative format of object inside your preferred one. If neither is supported, the browser then displays the text content. To support older or different versions of browsers you might add older code, such as deprecated `<embed>` and `<applet>` elements inside the `<object>` element.

The <object> Element's Attributes

The `<object>` element can carry all of the universal attributes, the UI event attributes, and the following attributes:

```
archive border classid codebase codetype data declare height width hspace
vspace name standby tabindex usemap
```

The archive Attribute

The `archive` attribute is particularly of use with Java-based applications. It allows you to preload classes or collections of objects in an archive, for example when one class relies on others, and tends to be used to improve speed. The value should be one or more URLs to the resources in a space-separated list.

The border Attribute (deprecated)

The `border` attributes specifies the width of the border to appear around the object; the value is specified in pixels. You should use the `border` property in CSS instead.

The classid Attribute

The `classid` attribute is designed to specify the objects' implementation. When you are trying to include Flash or QuickTime files and a plug-in needs to be loaded, this value would indicate the application required to play or run the file. When you are working with Java, the value of this attribute is likely to be the Java class you want to include.

The value is supposed to be a URL according to the W3C, although IE for Windows tends to use it to store the registry key associated with that program, as shown here with a key for QuickTime movies:

```
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
```

A registry key is added to the registry whenever a new program is installed, and is a unique identifier for that program.

> **Note that you should not go into the registry without backing it up first, and do not make any changes to values in there, or you might prevent your computer from working properly.**

The codebase Attribute

The `codebase` attribute is supposed to give an alternative base URL for any relative URLs in the `<object>` element; otherwise the folder the page is in will be used. For example, if you were working with Java it might look like this:

```
codebase="http://www.example.org/javaclasses/"
```

However, when it comes to files such as QuickTime movies and Flash, IE uses it to specify where the program required to play or run the file can be found. For example, the QuickTime ActiveX control (required to play QuickTime movies) can be downloaded from here:

```
codebase="http://www.apple.com/qtactivex/qtplugin.cab"
```

It can also identify the version of the file that should be downloaded. If the object isn't installed on the machine loading the page, the browser should go to the URL specified to get it (although it will probably show users an alert first before starting to download).

The codetype Attribute

The `codetype` attribute specifies the MIME type expected by the browser. It is relevant only if a `classid` attribute has already been specified. For example, if you are working with Java, it might be:

```
codetype="application/java"
```

If you wanted to embed a QuickTime movie, you would use a value like this:

```
codetype="video/quicktime"
```

Browsers can use the `codetype` attribute to skip over unsupported media types without having to download unnecessary objects. Appendix H covers MIME types.

The declare Attribute

The `declare` attribute is used to declare an object without instantiating it. It could be used for forward references to objects, so they get loaded only if used, to create cross-references to other objects, or when you are using the object as a parameter within another object.

It is a Boolean attribute, and while it does not need a value in HTML, all attributes in XHTML require a value, so you would use:

```
declare="declare"
```

The data Attribute

If the object has a file to process or play, then the `data` attribute specifies the URL for that file. For example, here is a URL to an MP3:

```
data="http://www.example.com/mp3s/newsong.mp3"
```

The value can be a relative URL, which would be relative to the value provided in the `codebase` attribute if specified, otherwise relative to the page itself.

The height and width Attributes

The `height` and `width` attributes specify the height and width of an object. The values should be in pixels or a percentage of the containing element. It is treated just the like `height` and `width` attributes of the `<img>` element. The use of these attributes should make the page load faster because the browser can lay out the rest of the page without completely loading the object.

The hspace and vspace attributes (deprecated)

The `hspace` and `vspace` attributes specify the amount of whitespace that should appear around an object, just like when they are used with the `<img>` element. They have been replaced by the `margin` and `border` properties of CSS.

The name Attribute (deprecated)

The `name` attribute provides a name that can be used to refer to the object, in particular for use in scripts. It has been replaced by the `id` attribute in Strict XHTML.

The standby Attribute

The `standby` attribute specifies a text string that will be used when the object is loading.

```
standby="Trailer for Harry Potter 27 is loading"
```

The value should be a meaningful description of the object that is loading.

The tabindex Attribute

The `tabindex` attribute indicates the tab index of the object within a page. Tabbing order is discussed in Chapter 6.

The usemap Attribute

The `usemap` attribute indicates that the object is an image map containing defined areas that are hyperlinks. Its value is the map file used with the object. It can be a complete URL to an external file or a reference to the value of an inline `<mapElement>`'s `mapName` attribute. See the "Image Maps" section later in this chapter.

The <param> Element

The <param> element is the first thing inside an <object> element. It is used to specify the parameters that each object can take, and values for those parameters that are required at runtime.

For example, a QuickTime movie would accept a parameter with the name autoplay; this parameter would be used to indicate whether the movie should automatically start to play when the page loads, in which case the value could be true. Otherwise, if the user is expected to press a play button then the value specified would have to be false.

As well as the universal attributes and basic events, the <param> element can carry the following attributes:

```
name type value valuetype
```

The name and value Attributes

The name and value attributes act as a name/value pair (rather like attributes themselves). The name attribute provides a name for the parameter you are passing to the application, while the value gives the value of the parameter.

Here are a couple of examples, taken from a QuickTime movie. The first parameter indicates the source of the file being loaded to play, while the second indicates that the movie should start playing automatically as it is loading (without the user having to start it):

```
<param name="src" value="movieTrailer.movie" />
<param name="autoplay" value="true" />
```

If you were working with a Java applet, you could use the name and value attribute to pass values into a method.

The valuetype Attribute

If your object accepts parameters, then the valuetype attribute indicates whether the parameter will be a file, URL, or indeed another object. The table that follows shows the possible values.

Value	Purpose
data	The parameter value is a simple string—this is the default value.
ref	The parameter value is a URL
object	The parameter value is another object

The value Attribute

You do not need to specify a value attribute if you are just passing a string to an object as a parameter. However, if you are passing a URL or object, then you should use the value attribute. Its purpose is to tell the object the MIME type of the parameter it is being passed.

For example, you might want to specify that you were passing a Java object as a parameter, in which case you would use the value attribute like so:

```
value="application/java"
```

Using Images as Links

Images are often used to create graphical buttons or links to other pages. Turning an image into a link is as simple as placing the `<img />` element inside an `<a>` element, like so:

```
<a href="../index.html" title="Click here to return to the home page">
<img src="images/banana.jpg" width="130" height="130" alt="Banana" border="0"
/></a>
```

Note the use of the deprecated `border` attribute. When you use an image inside an `<a>` element, the image will gain a border in IE for Windows, as shown in Figure 4-11.

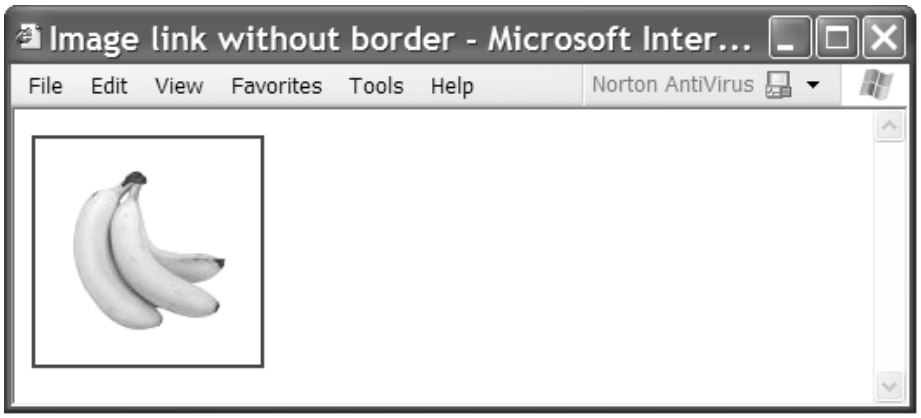

Figure 4-11

This border can be quite unsightly, so you either specify that the `border` should be `0` pixels wide, or set the `border` property of CSS for `<img />` elements to be `0` (which you will learn how to do in Chapter 8).

This will also work when you want to include images using the `<object>` element, but not necessarily when you use the `<object>` element to include other objects, such as a Flash movie.

> *Putting a Flash movie inside `<a>` elements will rarely turn it into links. If you want a Flash movie to act as a link, you must put the link in the Flash file instead. A popular way of doing this to add a transparent layer as a button over the whole movie and use this to link to your chosen destination.*

Image Maps

Image maps allow you to specify different areas of an image in your code, so that when users click different parts of the image, they get taken to different pages.

There are two types of image map:

- Server-side image maps
- Client side image maps

The difference between the two is how it is decided which link you should be taken to. With client-side image maps, the browser indicates which page you should be taken to based upon where the user clicks, whereas with server-side image maps the browser sends the server coordinates of where the user clicked and these are processed by a script file on the server that determines which page the user should be sent to.

Figure 4-12 shows a GIF that you will see turned into an image map. When users click the circle, they see what is on in the gallery; when they click the garden, they see the pages about the sculpture garden, and when they click the studios they see a page about the studios. Each of these sections is known as a clickable *hotspot*.

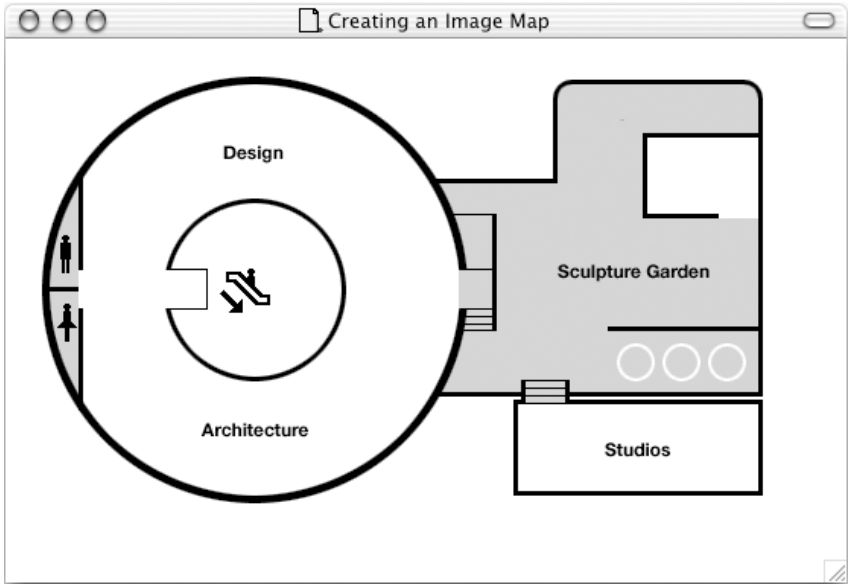

Figure 4-12

Image maps are particularly helpful when the image needs to be divided up in irregular shapes, such as maps. However, if the image can be divided up in a grid you might be better off chopping up an image manually and putting it together in a table, as you will see in Chapter 5.

> *These hotspots should not be too small; otherwise users might have difficulty in selecting the correct area they want. Users will soon get frustrated and leave your site if they click the wrong links on an image map.*

While image maps are often the only way to turn irregular shaped images into links, you are far better off putting images in a table if your image can be divided into a grid. Image maps can also be difficult for people with motor control difficulties to navigate, so if for any reason you use image maps as the main

method of navigation for your site you should offer text links at the bottom of the page (and indicate this in the `alt` text).

Server-Side Image Maps

With server-side images, the `<img>` element (inside an `<a>` element) carries a special `ismap` attribute, which tells the browser to send the server x, y coordinates representing where the user's mouse was when he or she clicked the image map. Then a script on the server is used to determine which page the user should be sent to based on the coordinates fed to it.

For example, look at the following link, where the `<img>` element carries the `ismap` attribute with a value of `ismap` (this is another attribute that did not require a value in HTML, and therefore uses its own name as a value in XHTML to make the attribute valid):

```
<a href="../location/map.asp"><img src="../images/states.gif" alt="map
of US States" border="0" ismap="ismap" /></a>
```

Now, if the user clicks the image 50 pixels to the right of the top-left corner of the image and 75 pixels down from the that same corner, the browser will send this information with the URL like so:

```
http://www.example.org/location/map.asp?50,75
```

You can see the coordinates appended at the end of the URL that is specified in the `<a>` element.

The thing about a server-side image map is that there needs to be a script, map file, or application on the server that can process the coordinates and know which page the user should then be sent to. The implementation of image maps will vary depending on what kind of server you are running on (whether it is a Windows server, or one of the many flavors of UNIX).

Because server-side image maps are processed on the server, the implementation of them is not covered by HTML or XHTML recommendations, and unfortunately there is not space to cover different possible implementations for each different platform here. If you want to learn about server-side image maps you should pick up a book that covers server-side scripting, such as a book on ASP, PHP, CGI, or JSP (which are discussed in Chapter 16). See the book list at Wrox.com for a list of books on topics such as these.

Client-Side Image Maps

Because server-side image maps rely on server technology, an alternative for the browser was introduced and client-side image maps were born. Client-side image maps use code within the HTML or XHTML page to indicate which parts of the image should link to which pages. Because the code that divides up the sections of the image is on the browser, it is possible for the browser to offer extra information to users either by showing them a URL in the status bar or as tooltip when the mouse is hovered over the image.

There are two methods of creating a client-side image map, using the `<map>` and `<area>` elements inside an `<img>` element, and more recently, using the `<map>` element inside the `<object>` element.

Client-Side Image Maps Using <map> and <area>

This earlier method of creating image maps has been supported for longer in browsers, going back to Netscape 4 and IE 4.

The image that is going to form the map is inserted into the page using the `<img />` element as normal, except it carries an extra attribute called `usemap`. The value of the `usemap` attribute is the value of the `name` attribute on the `<map>` element, which you are about to meet, preceded by a pound or hash sign.

The `<map>` element actually creates the map for the image and usually follows directly after the `<img />` element. It acts as a container for the `<area>` elements that actually define the clickable hotspots. The `<map>` element carries only one attribute, the `name` attribute, which is the name that identifies the map. This is how the `<img />` element knows which `<map>` element to use.

The `<area>` element specifies the shape and the coordinates that define the boundaries of each clickable hotspot. Here's an example from the image map that was used for the image in Figure 4-12.

```
<img src="gallery_map.gif" alt="Gallery Map" width="500" height="300"
border="0" usemap="#gallery" />

<map name="gallery">
  <area shape="circle" coords="154,150,59" href="foyer.html" target="_self"
     alt="Foyer" >
  <area shape="poly" coords="272,79,351,79,351,15,486,15,486,218,272,218,292,
     166,292,136,270,76" href="sculpture_garden.html" target="_self"
     alt="Sculpture garden" />
  <area shape="rect" coords="325,224,488,286" href="workshop.html"
     target="_self" alt="Artists workshops" />
</map>
```

As you can see, the value of the `usemap` attribute on the `<img />` element is `#gallery`, and this is used on the `<map>` element. Then the `<area>` elements actually define the sections of the image that are clickable.

If you have two areas that overlap each other, the first one in the code will take precedence.

The attributes that the `<area>` element can carry may look familiar from the `<a>` element. The ones that are relevant to image maps are covered here; otherwise see the "Adding Images Using the `<img />` Element" section earlier in this chapter.

```
accesskey alt shape coords href nohref target tabindex taborder notab
```

The shape Attribute

The `value` of the shape attribute actually affects how the browser will use the coordinates specified in the `coords` attribute, and is therefore required. If you do not specify a `shape` attribute, IE usually assumes the area is a rectangle.

The table that follows shows the possible values of the `shape` attribute.

Value	Shape Created
`default`	The whole of the image not defined in an area (should be specified last)
`rectangle` or `rect`	Rectangle
`polygon` or `poly`	Polygon
`circle` or `circ`	Circle

You are better off using the abbreviated versions of the values as they are better supported in older browsers. The value `default` should be used last if you want to indicate any sections of the image not otherwise indicated by an `<area>` element—it's like a catch-all for the rest of the image.

The coords Attribute

The `coords` attribute specifies the area that is the clickable hotspot. The number of coordinates you specify depends on the shape you are creating (and have specified in the `shape` attribute).

- A rectangle contains four coordinates. The first two coordinates represent the top left of the rectangle, and the second two the bottom right.
- A circle contains three coordinates; the first two are the center of the circle, while the third is the radius in pixels.
- A polygon contains two coordinates for each point of the polygon. So a triangle would contain six coordinates, a pentagon would contain ten, and so on. You do not need to specify the first coordinate at the end again because the shape is automatically closed.

Some Web authoring and image editing programs will help work out the coordinates of an image map for you; they provide a tool that allows you to select the areas you want to turn into a map and use those shapes to create the coordinates for you. Figure 4-13 shows you Dreamweaver's Image Map tool—because each program is different, you should look in the help files for that program to see how yours creates an image map.

The href and nohref Attributes

The `href` attribute works just like the `href` attribute for an `<a>` element. Its value is the URL of the page you want to load when the user clicks that part of the image.

If you do not have an `href` attribute, you must use a `nohref` attribute indicating that the area will not take you anywhere.

The alt Attribute

The `alt` attribute specifies a text alternative for that section of the image and works just like the `alt` attribute on the `<img />` element. It will actually override the `alt` text specified for the image when the user rolls over the area.

The target Attribute

The `target` attribute specifies which frame or window the page should be loaded into. Possible values are the same as for the target attribute of the `<a>` element.

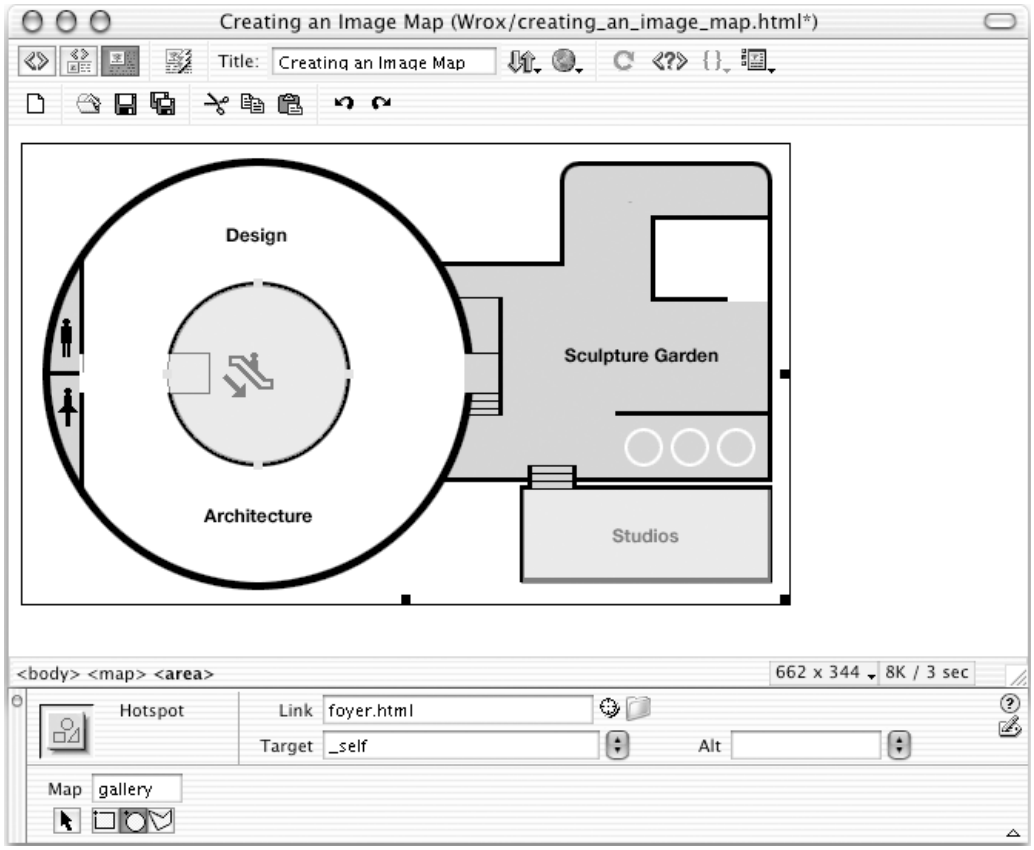

Figure 4-13

The tabindex Attributes

The `tabindex` attribute allows you to specify the order in which users can tab through items on a page. The value is a number between 1 and 32767. It is discussed in full in Chapter 7.

Client-Side Image Maps Using the <object> Element

HTML 4 started to promote the use of the `<object>` element rather than the `<map>` element for adding image maps to your documents (although you can still use the `<map>` element in Strict XHTML 1.0). The `<object>` element takes a different approach to creating image maps.

It is the `<object>` element that carries the `usemap` attribute (whose value is the value of the name attribute on the `<map>` element preceded by the pound or hash sign). Inside the `<object>` element you use the familiar `<map>` element with the `name` attribute. But inside the `<map>` element are standard `<a>` elements.

The presence of the `<a>` element in this context helps explain why it can carry attributes such as `shape` and `coords`.

```
<object data="gallery_map.gif" type="image/gif" alt="Gallery Map" width="500"
     height="300" border="0" usemap="#gallery" />

<map name="gallery">
  <a shape="circle" coords="154,150,59" href="foyer.html" target="_self">
Foyer</a>
  <a shape="poly" coords="272,79,351,79,351,15,486,15,486,218,272,218,292,166,
     292,136,270,76" href="sculpture_garden.html" target="_self">Sculpture
garden
     </a>
  <a shape="rect" coords="325,224,488,286" href="workshop.html"
target="_self">
     Artists workshops</a>
</map>
```

Rather than using `alt` attributes, you should put alt text (or a description of the link) inside the `<a>` element.

Unfortunately the support for this way of creating image maps is rather poor, so you are better off sticking to the old method for the moment.

Summary

In this chapter you have learned how to make your pages look a lot more exciting by adding color, images, and other multimedia objects.

First you saw how colors are specified in Web pages, using either color names or hexadecimal codes. While the hexadecimal codes can take a little getting used to, you can always use a reference to help you find the colors you want.

You then learned all about the different types of images used on the Web. While images add life to a page, you have to be careful with their sizes. If you have too many images or your images are too large they will slow down your site significantly. You therefore have to choose the format that will compress your image the best while retaining its quality. The GIF format is the format of choice for images with flat colors, while JPEGs are better for photographic images and graphics with gradients of the same color. Investing in good image-editing software that allows you to save images in these formats is a good idea if you use a lot of images on your pages.

While the `<img />` element is the most common way of including an image in your document today, you also saw the `<object>` element which is going to be used more in the future. The `<object>` element is already widely used for embedding other types of files and code into your pages, from Flash or QuickTime movies to Java applets and JavaScript objects.

Finally, you saw how to divide up an image into clickable hotspots that turn different parts of the image into separate links. Another way of creating separate links in the one image is by chopping it up and putting the separate sections into different cells of a table; you'll learn about tables in Chapter 5.

Exercises

The answers to all of the exercises are given in Appendix A.

1. Add the images that describe a shade, a tint, and a tone to the following example. All of the images are provided in the `images` folder in the download code for Chapter 4.

```
<h1>Color Definitions and Examples</h1>
<p>A <b>hue</b> is a pure color; it contains no black or white. It is the key
part of a color that allows it to be identified as red, green or blue.</p>
<img src="images/hue.gif" alt="Color sample for hues" />
<p>A <b>shade</b> is a hue with black added</p>
  Add shade1.gif here
<br />
<p>A <b>tint</b> is a hue with white added</p>
  Add tint1.gif here
<p>A <b>tone</b> is a hue with gray added</p>
  Add tone1.gif here
```

Your finished page should look like Figure 4-14.

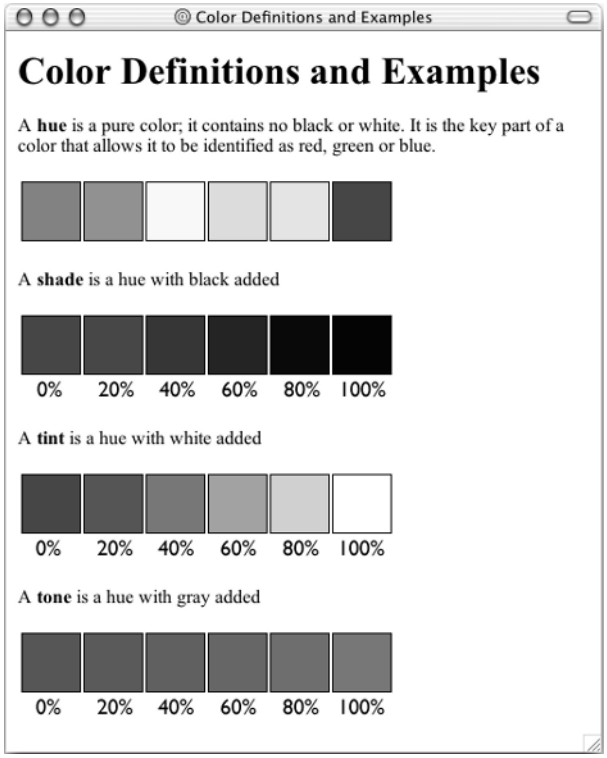

Figure 4-14

2. Look at the four images shown in Figure 4-15 and decide whether you are more likely to get smaller file sizes if you save them as JPEGs or GIFs.

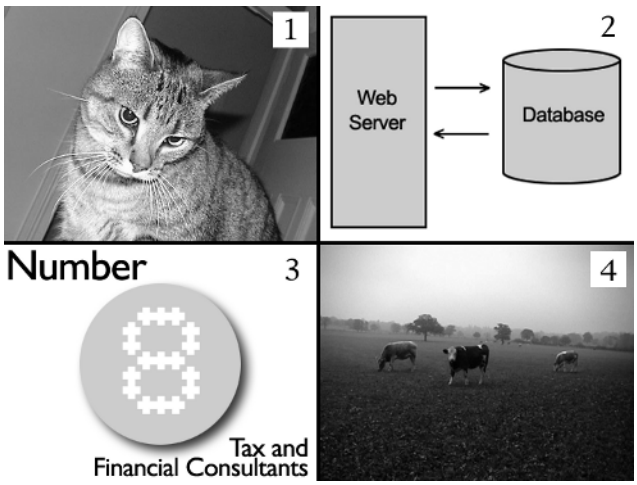

Figure 4-15