Chapter 1

CSS Fulfills a Promise

In This Chapter

- Improving HTML with CSS
- Making CSS work with the tools you already use
- Creating practical style sheets
- Avoiding browser compatibility problems
- Getting dramatic with filters

Underneath all Web pages is good old HTML, the markup language that controls things such as font sizes and color of text, where an image goes, and info about other elements of the page. HTML is sometimes called *plain* HTML, to distinguish it from Web pages built with more sophisticated techniques such as style sheets. And *plain* is sure a good word for HTML.

Without help, HTML often produces truly boring pages. Just as unpleasant as the lackluster pages it produces is the jumble of HTML code that results from trying to describe an entire Web page using HTML alone. Style sheets to the rescue.

Improving HTML

CSS (Cascading Style Sheets) was a technology recommended by the World Wide Web Consortium (W3C) in 1996. An easy way to understand the purpose of CSS is to view it as an addition to HTML that helps simplify and improve Web page design. In fact, some CSS effects are not possible via HTML alone.

Another advantage of CSS is that it allows you to specify a style once, but the browser can apply that style many times in a document. For example, if you want some of the pictures displayed in your Web site to have a thin, blue frame around them, you can define this frame as a style in your CSS. Then, instead of having to repeat an HTML definition of the thin and blue frame — each and every time you want that particular frame — you can merely insert the CSS style as an attribute for each graphic element that you want framed.

Put another way, you use CSS to define general rules about how the elements in your Web pages behave and how they look — where they're located, their size, their opacity, and so on. Then you can merely refer to the rule's name whenever you want to enforce it within your HTML page.

Here's a CSS rule that defines a couple of qualities you decide to apply to your largest headlines, H1:

```
<style>
H1 { font-size:16pt color:blue;}
</style>
```

With this CSS rule in effect, any HTML code containing an H1 element is automatically rendered in 16-point type and colored blue:

```
<html>
   <body>
      <h1>this headline is blue and 16 pt.</h1>
   </body>
</html>
```

CSS rules can be defined in a separate .css file or embedded within the HTML file. Here's the CSS headline style rule embedded within the header of an HTML file:

```
<html>
   <head>
      <style>
h1 { font-size:16pt color:blue;}
      </style>
   </head>
   <body>
      <h1>this headline is blue and 16 pt.</h1>
   </body>
</html>
```

Notice the `<style>` element. You can define your CSS styles inside this element. (You can also have multiple `<style>` elements on a page if you wish.)

For efficiency, nearly all the CSS code for the examples in this book is put right in the HTML document, within a `<style>` element, as in the preceding code. This makes saving the entire example — CSS plus HTML — as an .htm file easier. Just double-click the file in Windows Explorer to automatically load the example into Internet Explorer to see it work. However, in your own work, you're likely to put CSS in its own separate file, and then use the `<link>` element in the HTML document to import the CSS. You can put CSS styles in three places: an external file (with .css as the file extension); in the HTML file within the header section inside a `<style>` element; or even inside an HTML element, using the `style=` attribute. More on these issues in Chapter 3.

Getting Efficient with CSS

Defining a style in one location as CSS does has several advantages. First, it eliminates redundancy: You don't have to keep specifying its font size and color each time you use the <h1> tag in your document, for example. That makes Web page code easier to read and to modify later. If you're familiar with computer programming, think of a simple CSS style rule as something like a programming language *constant*: You specify, for example, the local tax rate by making up a name such as LocalTax, and then assigning a value to it like this: Constant LocalTax = .07. Thereafter, throughout your program, you don't need to repeatedly specify the .07. You merely use the constant's name LocalTax.

Similarly, after you've defined a CSS headline style, you can thereafter merely use the class name for that style, no matter how lengthy and complex that style might be. In this example, you use no class name, so every H1 headline is rendered with this style:

```
      <style>
h1 { font-size:16pt color:blue;}
      </style>
```

A second advantage of gathering all style definitions into a single location is that you can more easily make global changes. What if you decided to change all the H1 headlines to red instead of blue? If you didn't use a style sheet, you would have to search for all H1 elements throughout the entire Web site's HTML files and modify each of those elements in turn.

But if you had the foresight to use a style sheet, you need only change the *single* definition of the style for H1 in the style sheet itself. The specs are automatically applied throughout the HTML. In other words, just make this change from blue to red in the style sheet:

```
H1 { font-size:16pt color:red;}
```

All the headlines between the <h1> and </h1> tags throughout the entire Web site are now displayed as red text.

Changing Web design for the better

HTML originally was designed to work something like an outline, specifying the *structure* of a document, without too much attention paid to the actual *visual style*, or design, of the document. An outline merely organizes ideas hierarchically: A, B, C, and so on are the major ideas. Within those categories,

you have subdivisions such as 1, 2, 3, 4 and even lower divisions such as a, b, c, d and so on. The equivalent outline structure in HTML is described with various headline levels such as H1, H2, H3, and so on.

HTML was supposed to simply define content: This is body text, this is a headline, this is a table, and so on. But Web designers naturally wanted to offer ever more compelling, visually attractive Web pages. After all, the Internet more often competes with lively television ads than with dry, highly structured, academic journals. HTML began to grow willy-nilly by adding many special formatting elements and attributes such as italics and color. This inflation of tags made creating, reading, and modifying HTML increasingly cumbersome. Separating the content (structure) from the page's design and layout became necessary. Enter CSS. When you use CSS, the HTML is left to primarily handle the structure and the CSS file contains the styles defining how the HTML elements look.

Also, CSS also offers the Web page designer features unavailable in plain HTML. And as you'll see throughout this book, CSS gives a designer much greater control over the appearance of a Web page.

Being ready for anything

Of course, you'll never have *absolute* control over Web pages if you create sites for the Internet. There will never be a truly stable, single, predictable display for Web pages. Why? Because, like some celebrities, a Web page never knows where it's going to end up from minute to minute. It has to be prepared to be on display in all kinds of situations.

A Web page might be shown on a Pocket PC PDA screen — with very few pixels and in black and white. Or it might be shown on the huge Diamond Vision display in Hong Kong, which is longer than a Boeing 747, or even the Jumbotron screen in Toronto's Skydome, which measures 110 feet wide by 33 feet tall.

Not only do you have to consider huge differences in size, but also in *aspect ratio* (shape). Many computer monitors are still the traditional square shape, but increasingly Internet users are switching to widescreen monitors — wider than they are high, like a movie screen — to better display HDTV and DVDs. For Internet users, widescreen just means you see more horizontal information per page. Web pages designed with absolute (unchanging) positioning leave several inches of empty white space along the right side of a widescreen monitor. What would Vincent do?

How would van Gogh have dealt with the problem of designing a picture of a vase of sunflowers that might be shown on a widescreen Jumbotron, but also on a little square monitor?

The basic solution to this problem is to specify size and position in *relative* rather than absolute terms. For example, instead of saying, "The sunflower is 2 inches high and is located 12 inches from the left side," (an *absolute* specification), you say, "The sunflower is 6 percent large and 35 percent from the left side" (*a relative specification*). Other ways of specifying sizes relatively include *pixels* (which are the smallest units of information that a given monitor can display, so they vary from monitor to monitor) or such general terms as *x-large* or *large*.

Alert readers might be asking at this point, "Six percent of what?" The percentage is calculated based on the *containing block*. It can be the browser window (`<body>`), but it can also be such blocks as a `<div>` within the `<body>`. In this example, the containing block is the total size of the browser, but you can also specify percentage for other, smaller, containers within the browser window. More on this issue in Chapter 4.

Relative specs translate well into various sizes of displays. A sunflower 6 percent large would be displayed with about 48 pixels on an 800x600 computer monitor, but displayed 18 feet wide on a Jumbotron that's 300 feet wide.

In other words — when you specify *relative* measurements or positions — your graphics or text are automatically *scaled* as necessary to fit whatever size display is being used at the time.

Of course, if you're building pages for an *intranet* site, you might well know that everyone in your office network is required to use the same size screen, the same browser, the same operating system, and allowed no family photos in their cubicle. If that's the case, why are you working for a fascist organization? Just kidding. In those situations where uniformity is enforced across the entire company, you *can* provide absolute specifications, but such situations are relatively rare.

To play devil's advocate here, I would advise that you not worry yourself *too* much about how your Web pages look on various devices. I realize that most books on CSS — and certainly the theorists and committees that wrestle with CSS standards — are very troubled by "browser independence." They want CSS styles to not only be scalable (stretch or shrink to fit various screen sizes), but to also display your page designs, colors, and other effects *the same way on different browsers and even all the old versions of all those browsers.*

One big problem with this theory is that when you try to put browser- and device-independence into practice, you're often forced to accept the lowest common denominator. In my view, you should design Web pages for Internet Explorer (IE) version 6 running on a typical 17" monitor. Why? Here are the reasons:

- More than 95 percent of the people visiting your Web site use IE 6.
- You can take advantage of lots of cool effects that work only in IE or IE 6.

- Your job is much easier if you're designing for a predictable, stable canvas.
- A design that works equally well on a PDA screen and a computer monitor is rare indeed, and many more users access your Web pages with a desktop computer than a PDA.

True, several years ago, Internet users were divided between Netscape and IE, so you had to take Netscape and its peculiarities into account. No more. At least for now, the browser wars are over, and Netscape is merely a small, marginal player these days.

Designers Want to Design

It's not surprising that designers, not to mention marketing people, want to build attractive Web pages. Color, transition effects, and even various kinds of animation and other special effects are all desirable attributes and, designers say, necessary goals in a competitive world.

Designers have worked for years with feature-rich image manipulation tools such as Photoshop and powerful page design tools such as PageMaker. In the early years of the World Wide Web, designers saw no reason why they shouldn't be able to manipulate Web pages with the same freedom. True, animation adds considerable complexity, and there's always the possibility of future multi-platform conflicts for Web design, requiring that you sometimes design for more than one platform.

But regardless of the daunting obstacles, the goal remains to make Web sites as compelling, entertaining, and beautiful as possible. CSS is clearly a step in the right direction. Designing for a predictable target platform such as Internet Explorer 6 makes design far easier, and the results far more attractive.

With CSS, a designer can accomplish many things that are either difficult or impossible using ordinary HTML. For example, just a few of the tasks you can accomplish via CSS are:

- Customizing text indention
- Creating fades, dissolves, and other transitions between pages
- Gaining additional control over formatting, such as adding frames around blocks of text
- Precisely positioning or tiling background graphics
- Being highly specific about point size and other measurement units such inches when describing the size and position of graphics or text

Understanding the digital effect

Those of us who work within the digital domain are just beginning to realize what a profound difference digitization makes. A digital camera memorizes a mathematical pattern of pixels. With film, you can manipulate the picture only grossly — with techniques such as over exposure, solarization, scratching it with a knife, cutting and pasting, or superimposing two negatives. These and other *analog* effects are extremely crude compared to digital effects. Digital manipulation can be as complete, as subtle, and as refined as reality itself. You've doubtless seen those short animations where one object transforms into another — a boy into a girl, an ostrich into a Buick, and so on. This illustrates the total manipulability of digital information. Given that you can easily control *every pixel* in a digital photo, you can transform anything into anything else. What's more, you have the ability to modify an image infinitely.

It's no longer a world of compromises, with less-than-special effects like Claymation, stop-frame, scale models, and so on. These have become quaint historical techniques.

It is no longer a matter of whether you do something on screen: It's just a matter of how much it costs and how long it takes. New cartoons like *The Polar Express* and *Finding Nemo* demonstrate that digital effects are increasingly easy to achieve.

Artists are now getting control over the auditory (music) and visual realms (movies and photos) that publishers got over the typographic realm when Guttenberg invented moveable type. No longer must things be done clumsily by hand, like monks lettering and designing pages of the Bible, one Bible at time. Instead, with digital effects, you can, for example, effectively add a shadow to a visual element by merely selecting the object and then clicking a button to add a semi-transparent shadow. What makes all this so easy is that *every tiny dot* in the photo is represented as a set of numbers. And numbers, unlike film negatives, can be endlessly and precisely manipulated in *any way*. Adding a shadow is a matter of figuring out the mathematical function that adjusts pixels to make them look shadowed. This, and countless other visual functions, has been worked out by the people who developed Photoshop and other graphics applications. (One approach is to have the computer analyze a real shadow to see its mathematical gradient and other qualities.) So, if you have a particular background effect in mind for your Web page, you can achieve it — if you have the experience and skill to go about digital manipulation.

- Managing margins effectively
- Manipulating with great precision character and word spacing, in addition to kerning (adjusting the spacing between lines of text), leading (space between lines), and justification
- Providing unique navigation tools for the user
- Specifying the z-axis (what is "on top") for layers of text and graphics

Where CSS Fits with the Tools You Already Use

You can write a style sheet using any plain text editor, such as Notepad. However, you can also use specialized CSS editors that offer shortcuts to the creation of a style sheet. With editors like Microsoft's Visual Studio or TopStyle Pro from Magia Internet Studio, you can, for example, choose a text color from a palette, drag and drop a graphic from a toolbox, or select a font size from a list. Then the editor automatically translates your choices (generally made by dragging and dropping or clicking with the mouse) into the text descriptions that make up a style sheet. For such activities as moving page elements around to find the most attractive layout, mouse dragging can be a real time-saver.

If you open a CSS file, and you've been using Microsoft's Visual Studio on your computer, by default, the CSS is displayed in Visual Studio, as shown in Figure 1-1.

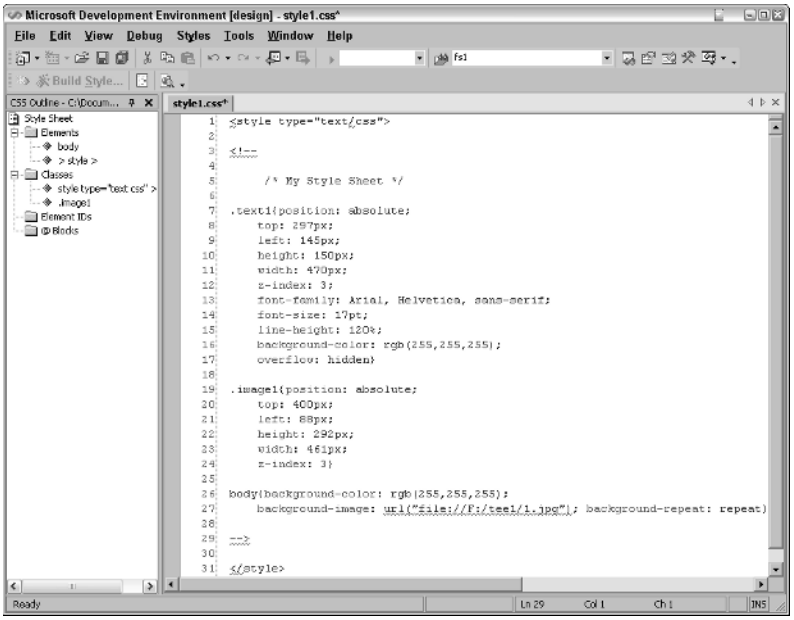

Figure 1-1: CSS files can be managed within Microsoft's Visual Studio.

Many Web programmers use Visual Studio and its ASP.NET features to create richly interactive Web sites. But artists and designers can also use Visual Studio to create CSS files. As you see in Figure 1-1, this style sheet specifies a text box, an image, and a background image. On the left is an abstract view (an "outline") of the style sheet; on the right is the actual, editable code. If

you want to add some more style definitions, nothing could be simpler. You *don't have to write the code yourself*. Just click the Build Style button and the Style Builder dialog box opens, as shown in Figure 1-2:

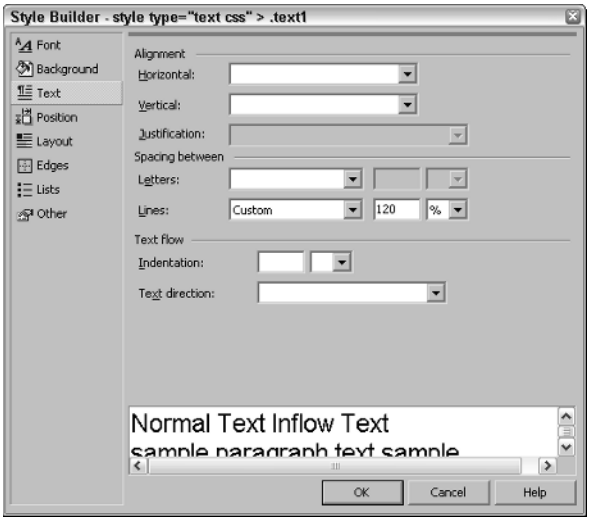

Figure 1-2: Create new styles the easy way, with Style Builder, a feature of Visual Studio.

Click any of the categories on the left — such as Background, Position, or Lists — and you see a new dialog box with additional options.

To start from scratch and create a brand-new style sheet, choose File➪New➪File and double-click the Style Sheet icon.

To create a new style by associating it with an existing HTML element such as <h2>, choose Styles➪Add Style Rule. The Add Style Rule dialog box opens, as shown in Figure 1-3:

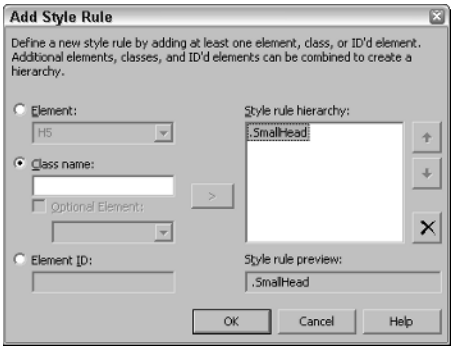

Figure 1-3: Use this dialog box to introduce new styles.

Remember that when you add CSS to your Web design bag of tricks, you don't simply abandon HTML. Instead, CSS allows you to modify HTML's tags. In some cases, however, you might find that you want to use some of CSS's features rather than the traditional HTML. For instance, many people think it's better to use CSS positioning tools rather than relying so heavily on tables like classic HTML. Similarly, you might decide to abandon the venerable HTML `<font>` tag in favor of the more powerful and refined text descriptors available in CSS.

Above all, don't be intimidated. CSS is not conceptually difficult, nor is it hard to use in practice.

Getting Practical

Perhaps the single biggest leap of faith that Web page designers must make is to *think beyond HTML* when using style sheets. Many computer programs support HTML — even Office 2003 applications such as Word have Web Page Preview and Save As Web Page options on their File menus. But with CSS, you need to augment your current Web page design habits and tools into a bit of abstract thinking. Academics would say that CSS is primarily a system that allows you to define abstract classes that can be applied with practical results in Web page design. I say that CSS makes design easier.

Look for CSS features in your current software

Doubtless you've used at least one application to build HTML that ends up as a Web page. If you're comfortable with a particular Web page design tool, go ahead and continue using it. But check to see if there are any features in your current software that support CSS. Search the product's Help Index for CSS — possibly you've never noticed a CSS tool sitting right there all the time.

Resources on the Web

As an alternative, you can use popular programming editors like Visual Studio, or dedicated CSS editors, to analyze existing Web pages and abstract CSS style sheets from them — or build a CSS file from scratch. If you don't yet have access to any CSS tools, take a look at the following tip:

You can find many CSS designer tools — some for free — on the Internet. Check out the list of CSS authoring tools at this W3C Web site: `www.w3.org/Style/CSS/`. On the topic of CSS Resources, you can often find useful answers to your questions about CSS at this newsgroup: `comp.infosystems.www.authoring.stylesheets`.

Of course, there's always Microsoft's Web site. At the time of this writing, Microsoft's main CSS index at this address: `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsintro7/html/vxtskWorkingWithHTMLStyles.asp`. However, Microsoft rarely retains a name or address for its technologies and documentation for long. So you'll likely need to look for CSS using the Search feature on the main Microsoft Web page at `www.microsoft.com`.

Also, considerable information (although somewhat stiffly, academically presented) is available at the Web Style Sheets home page sponsored by W3C at `http://www.w3.org/Style`.

Finally, take a look at Chapter 19, where several additional online CSS resources are described.

Avoiding Browser Compatibility Problems

Early Web page designers faced a peculiar problem: Different Web browsers interpreted HTML in different ways. In those early days, Netscape was still widely used. Fortunately — at least for Web designers — today, more than 95 percent of the people visiting Web sites use Internet Explorer (IE). What's more, most of them can be expected to use a recent version of IE. This makes a designer's life easier. You can expect that most people will see your Web pages as you intend them to be seen — as long as you stick to the specifications and capabilities of the current version of IE.

Many books on CSS spend quite a bit of time listing and describing the incompatibilities between Netscape and Internet Explorer — demonstrating the differing ways that these browsers implement CSS features. Another potential source of incompatibility derives from the differences between operating systems, namely PC and Mac.

But consider the worst case: What happens if a CSS feature is used in your Web page code, but it isn't supported by a user's browser or OS? Nothing happens. Unlike programming languages — with their often baffling error messages that can scare users — browsers are designed to hide problems from their users. The user never sees an error message saying: "This CSS style is unsupported by Netscape." Instead, whatever special effect you were trying to achieve by redefining an HTML element with CSS is simply ignored. If you had redefined `<h1>` as a blue headline, that redefinition is ignored and the default black is just used instead. If CSS is controlling positioning, however, the results can be less benign. But, again, the damage is limited to those few people not using IE.

Browser compatibility isn't nearly as much of an issue now as it was a few years ago when Netscape was more popular. And Mac computers, too, represent a small portion of today's computer market. Fair or not, the browser wars and the OS wars have settled into at least a temporary truce — and Web designers can benefit from the single primary platform they can build for.

Nonetheless, some Web site designers *must* wrestle with the compatibility problem. If the issue concerns you, take a look at Chapter 17, where I discuss various strategies you can employ to at least minimize — if not prevent — the damage done to your great designs by minor or simply out-of-date browsers.

Getting Dramatic with Filters

To give you a taste of how effective and powerful special browser effects can be, take a look at a few filters you can add to your Web pages. *Filters* are a set of special animated effects that Microsoft built into Internet Explorer.

Type this into Notepad or your choice of CSS editor:

```
<html>
<head>

<style>

div.box {width: 300px; height: 200px; padding: 30px;
         font: 46pt times new roman;}
</style>

</head>
<body>

<div class="box" style=" filter:
           progid:DXImageTransform.Microsoft.Alpha

(Opacity=100,
FinishOpacity=0, Style=1, StartX=0, FinishX=0, StartY=0,
FinishY=100)">

Hey...you can modify opacity.</div>


</body>
</html>
```

Now save it to a file named opac.htm, and then double-click on that filename in Windows Explorer. Your Internet Explorer window should open, displaying the text with its opacity adjusted from 0 (can't see through it at all) to 100 (can see through it completely), as shown in Figure 1-4.

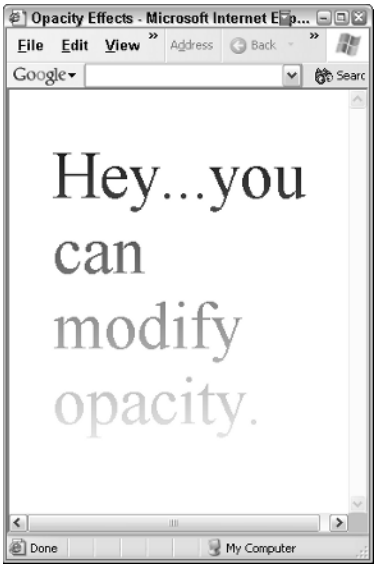

Figure 1-4: You can adjust opacity to suit yourself by applying a CSS style.

Notice in Figure 1-4 that a gradient is created, gradually fading the text. In effect, what's happening here is that the background (by default white) is simply showing through more and more because the text is increasingly transparent.

So what, you might say? I can get the same effect by writing some text in a graphics program like Photoshop, and then applying an opacity gradient. I can then save this text as a graphic file and just import it into my Web page as an image. Sure, you can do that. But creating a CSS style to accomplish the same thing has several advantages. You can apply that style easily to any additional text blocks in your Web site by merely using additional `<div>` tags. What's more, with a little extra programming, you can cause these kinds of effects to become animated — to be dynamic. For instance, you could use the opacity filter to fade some text, or a graphic, slowly in or out of the page. You could allow items to gently fade in or out in response to something the user does with the mouse. Or how about having entire sections or pages fade relatively rapidly as a transition effect to the next section or page? Lots of cool effects can be achieved when you add a little scripting and some timers to various filters.

Sure, filters are only built into Internet Explorer (version 4 and later). And, technically, they're not exclusively a CSS effect — although CSS does make using them easier. But so what? Filters are increasingly being used by Web page designers as a way of competing with television. Just as adding color was a real improvement over black and white Web pages, so, too, is animation a significant improvement over static pages.

Try a new trick. To create a different gradient — a circular one this time — just make two little changes to your HTML code (shown in boldface). You want to change the opacity `style` attribute from `1` to `2` (this changes the gradient to a circular effect), and provide a color background so you can more easily see the circular radiation of the gradient:

```
<html>
<head>

<style>

div.box {width: 300px; height: 200px; padding: 30px;
         font: 46pt times new roman;}
</style>

</head>
<body>

<div class="box" style="background: green; filter:
           progid:DXImageTransform.Microsoft.Alpha

(Opacity=100,
FinishOpacity=0, Style=2, StartX=0, FinishX=0, StartY=0,
FinishY=100)">

Hey...you can modify opacity.</div>


</body>
</html>
```

Save this HTML to an .htm file, and then double-click it in Windows Explorer to load it into IE. You see that the gradient has become circular, as shown in Figure 1-5.

You'll find lots of special effects you can employ built into IE, several of which you explore in Chapter 13. There are shadows, inversions, reversions, conversions, and a few mild perversions thrown in for the Goth crowd.

As an example of the great variety of effects available via filters, consider transition wipes. Transition wipes — just one of many kinds of effects you can use — provide smooth connections between two elements. These wipes are used in films and video as a way of moving from scene to scene, indicating moving through space or the passage of time, or some other transitional behavior — such as between reality and a dream. You can use them in Web pages for similar purposes. To give you an idea of just some of the effects you can employ, Table 1-1 shows a list of transition wipes.

Figure 1-5: Change the opacity Style to 2 to create circular gradients.

Table 1-1 Transition Wipes

Transition Wipe	*Value*
Box in	0
Box out	1
Circle in	2
Circle out	3
Wipe up	4
Wipe down	5
Wipe right	6
Wipe left	7
Vertical blinds	8
Horizontal blinds	9
Checkerboard across	10

(continued)

Table 1-1 ***(continued)***

Transition Wipe	*Value*
Checkerboard down	11
Random dissolve	12
Split vertical in	13
Split vertical out	14
Split horizontal in	15
Split horizontal out	16
Strips left down	17
Strips left up	18
Strips right down	19
Strips right up	20
Random bars horizontal	21
Random bars vertical	22
Random	23

You can combine or blend various effects to generate new effects. I won't say "The possibilities are endless — you're only limited by your imagination," because that's the single most tedious cliché of the computer age. But you can sure do some great visual stuff with CSS and Internet Explorer. If you can't wait to get to the discussion of dynamic animation tricks, flip over to Chapters 13 or 16 and dive in.