CHAPTER 1

Web Design: The Big Picture

For the most part, if you were to venture to your local bookstore, you would find the web design books split into two separate sections: tools/technology and design. Why is this? Aren't design issues important to those learning web design tools? Absolutely! Unfortunately, most tool-specific books on web design are just that: tool-specific. Someone buys them so they can learn about a specific program. Many people have difficulty seeing that the tool itself is just the beginning of web design. It's something you use to realize the creations that, if you've done your job right, have already been completed in your head and on paper.

This paper and brain production (sometimes referred to as predigital production) is the web design topic that appears in those *other* books in the *other* part of your local bookstore's computer section. So, what's the big deal about big picture web design issues anyway? Well, your website will sink or swim based on its *design* (a general term that covers many different issues that will be discussed in this chapter). As a result, even if you are focusing on a specific tool (as you will be with Dreamweaver in this book), it's a good idea to at least have a primer in the basics of web design. Now, this chapter is an *introduction* to the big picture of web design issues; there is no way that I could cover everything you need to know in one chapter. However, this chapter, which is part web history lesson and part web design starter, should give you a basic introduction to web design, along with an idea of what you might want to explore further on your own. At the very least, it will introduce you to some of the most important design issues so you don't inadvertently make a major blunder when you are building your HTML creation.

- **How the Web works**
- **Browser madness**
- **Fundamentals of interactive design**

How the Web Works

As early as the 1940s, computer scientists were moving away from the notion of building computers that mimicked the Newtonian universe. Rather, computers were seen to be instruments to manage and distribute huge amounts of information. Douglas Engelbart, upon reading Vannevar Bush's 1945 famous *Atlantic Monthly* article "As We May Think," noted that "If these machines [could] show you information on printouts, they could show you that information on a screen." Further, he saw a link between the TV-like interface and information processor as a medium for representing symbols. An individual could steer through different information spaces viewing data and graphics in different ways. Most importantly, however, was the fact that Engelbart easily saw the expansion of the medium into a theater-like environment in which one could sit with colleagues and exchange information simultaneously on many levels. The realization of these ideas, however, did not come to fruition for some 44 years.

Why should a humble Dreamweaver user care about the roots of the Web? Well, in order to better understand the medium in which you'll be working, it's helpful to at least have a cursory idea as to how it all came about. After all, most traditional artists (whether they are painters, printmakers, or sculptors) benefit from a solid understanding of the particular form in which they are working.

This section of the chapter will start off with a brief exploration of how the web got started. From there, we'll move on to how Hypertext Markup Language (HTML), the lifeblood of the Web, works.

> It's important to remember that this book doesn't even presume to teach you HTML—that's best learned elsewhere if you are interested. However, your introduction to the big picture web design issues wouldn't be complete without at least a passing look at HTML. And I would be neglecting my duties if I did not say that if you are serious about web design, you should really become familiar with HTML.

Finally, we'll finish off this section with a brief look at the Web, as it exists today. Remember, however, that this section, while important, is just introductory in nature. If you are interested in bypassing it so you can get right on to the real meat of the chapter (the taste of design), go right ahead.

Not So Humble Beginnings

Originally developed as an initiative of the Defense Agency Research Projects Administration (DARPA) in partnership with several prestigious universities and research institutions, the Internet, which at that time was called ARPAnet, was a prototype communications system designed to withstand the inevitable electromagnetic pulse generated by a thermonuclear attack. Yeah, that's right, *the* bomb. It wasn't long before ARPAnet spawned a sister

network for the research community supported and funded by the National Science Foundation (NSF). Called the NSFnet, the network soon found its way into most universities in the United States, Canada, and abroad.

For all of its technological wonder, however, NSFnet was little more than an obscure scientific endeavor. NSFnet, which ultimately became known as the Internet, required adeptness at cryptic computer programs and obscure protocols. Even if one mastered the necessary skill to take advantage of the Internet, where to go and what to do was pretty puzzling, and it wasn't useful for much other than e-mailing.

At its height, NSFnet was a massive library of some of the most advanced information on the planet. The problem rested on the fact that it was extremely difficult to identify and locate a given source of information. It was akin to walking down each isle of an enormous library in the dark, and scanning each book to figure out what was available. Once you found something relevant to your needs, you had to "read" (that is, download) the entire book rather than browse just the parts that were of interest to you. Worse still, once found, a piece of information often referred to another valuable source without providing the means to locate it.

In 1989, tired of the perpetual hunt and peck of the Internet, a researcher named Tim Berners-Lee at the CERN atomic research center in Switzerland proposed software and protocols that would allow computers to browse the information contained on the NSFnet. This one event irrevocably changed the nature of how this global computer network was used.

CERN stands for Conseil Européen pour la Recherche Nucleaire.

Berners-Lee's software (dubbed a *browser*) and protocols created the ability to easily browse information and navigate not only different documents on the same computer, but documents on other computers. Key to the technology was the concept of *hyperlinks*, which were highlighted words or symbols the user could click and be transported to the next "page" in the sequence. Berners-Lee's Hypertext Markup Language (HTML) made hyperlinks possible.

Within months of the release of the early version of the web browser developed by Berners-Lee, web software spread throughout the research community like wildfire. By July 1993, more than 130 servers were web-enabled.

Shortly thereafter, a small group of students at the University of Illinois, among them Marc Andreessen (who later founded Netscape), took it upon themselves to rectify many of the shortcomings of the very primitive prototype web browser available to the public. Most significant among their accomplishments was the addition of a GUI (graphical user interface) to what had been a mostly text-based software and the adaptation of the browser allowing it to function on the Windows operating system. Following this, there was an incredible jump in the number of websites—a phenomenon that can be directly correlated with the release of

the first version of the University of Illinois NCSA (National Center for Supercomputing Applications) Mosaic web browser (see Figure 1.1).

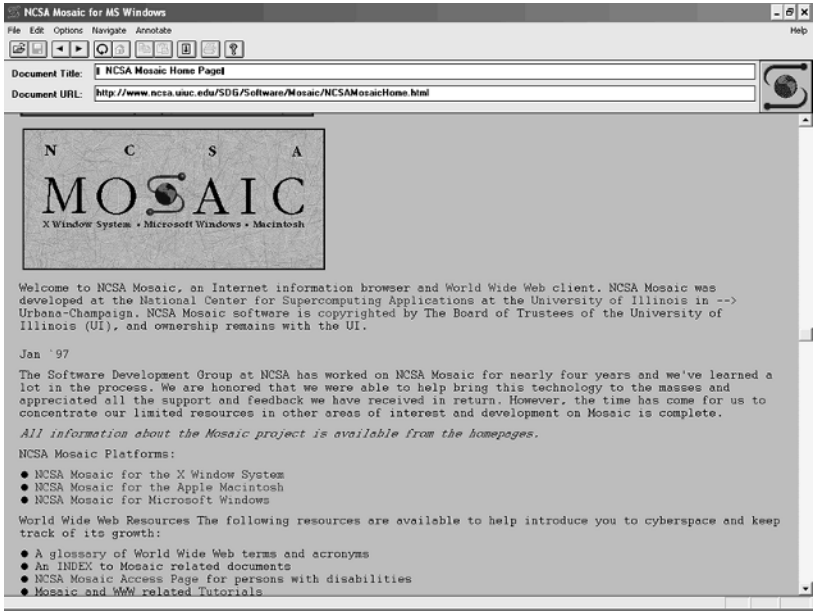
The rest, as they say, is history. NCSA Mosaic changed everything. Not only did the web become graphical, but it became accessible way beyond the academic community. Not long after Mosaic's introduction, Andreessen, along with five others, left the University of Illinois to found the Mosaic Communications Corporation, which later became the Netscape Communications Corporation, one of the first truly commercial web software ventures. Before you could say "holy alleged antitrust, Batman," Microsoft got in on the action, and thus begun the browser wars. However, before we discuss these and other browser-related design issues, there is some additional ground that needs to be covered in this section.

HTML: The Lifeblood of the Web

As mentioned earlier, Hypertext Markup Language (HTML) was one of the things that made the Web possible. In reality, HTML is not a computer language such as C++ or Pascal; it is a system for describing documents. A plain text document is "marked up" using a series of commands called *tags* (see Figure 1.2). A browser interprets the HTML document and displays it. The fact that an HTML document is plain text is significant for a couple of reasons. First, because plain text can be interpreted by any platform (IBM, Macintosh, and UNIX), HTML is truly cross platform and universal. Second, because HTML is plain text, it can be written using the simplest of programs (such as Windows' Notepad or Mac's SimpleText).

Figure 1.1

Mosaic is crude by today's standards, but compared to what was available when it came out, it was an earth-shattering piece of software.



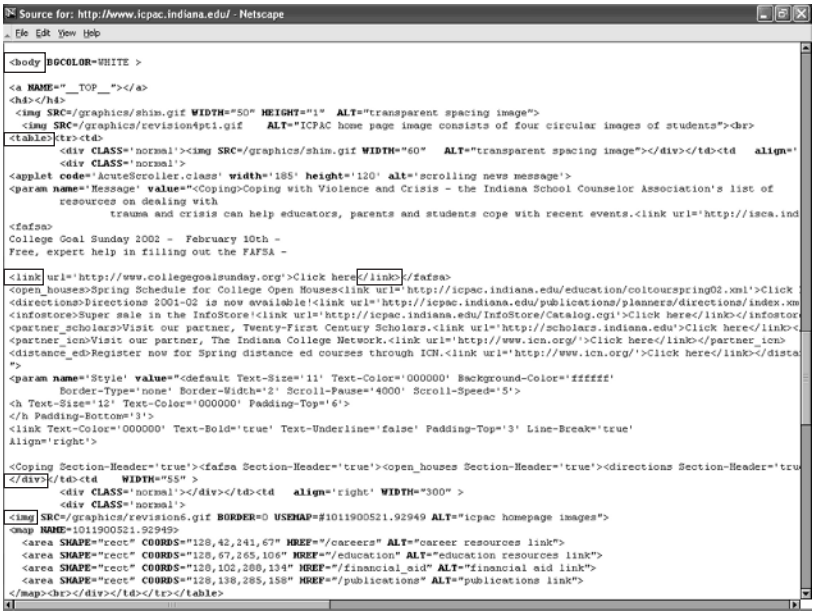


Figure 1.2

A web page is composed of content given structure through the use of HTML tags.

The first incarnation of HTML (version 1) allowed very basic page layout including font size, hyperlinks, and embedding of graphics. It is important to remember that HTML 1 was only the standard insofar as the first popular browser, Mosaic, was designed to interpret it. It wasn't until more browsers appeared on the market that a standard version of HTML was defined. As various versions of HTML were proposed and adopted by the World Wide Web Consortium (W3C, an organization that develops web standards), the language became more powerful. The inclusion of new layout features, such as frames and tables, allowed greater options for the development of web pages. In addition, image maps and new graphics formats increased the integration of images and basic interactivity.

The most recent version of HTML (version 4.01), which has been dubbed Dynamic HTML (DHTML), has some very exciting features that give web designers a lot more freedom than they had previously experienced. While each successive version of HTML offered unparallel opportunities for publishing hypertext documents, the medium was not as malleable as many would have liked. HTML documents were far more difficult to lay out than traditional print documents. In HTML, text could only be arranged in a limited number of ways, making it hard to create a compelling visual experience. In addition, the way HTML treated graphics made it difficult to lay out a document. Finally, the interactivity in HTML documents was

limited to the actual hypertext. It was not until the release of Dynamic HTML that many of these problems were solved.

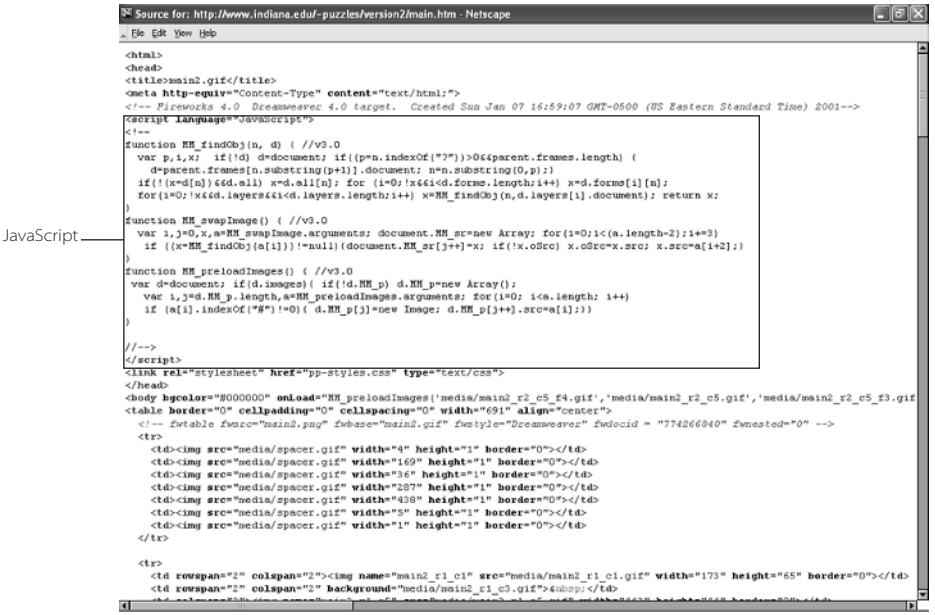
Strictly speaking, Dynamic HTML is really just the most recent version of HTML. The difference, which is relatively substantial, comes from two primary new features: Cascading Style Sheets (CSS) and the ability to integrate scripting languages more efficiently with HTML.

> If you are particularly interested in learning how to use Dreamweaver to create Dynamic HTML, see Part III, "Working with Dynamic HTML."

CSS let the designer precisely define fonts, margins, line spacing, and other elements of an HTML document. Unlike HTML, in which fonts could only be defined in four sizes, DHTML allows type to be defined in point size. In addition, CSS allow designers to specify x and y coordinates to achieve exact positioning of elements within a document. Further, much like traditional print documents made with Adobe PageMaker or QuarkXPress, elements can be stacked one upon another by defining their position with z coordinates. Simply put, CSS provide designers the opportunity to lay out web pages with a greater degree of accuracy, allow much more control over how a given document will appear in a browser, and provide for the stacking and composition of text and images.

DHTML becomes even more powerful when you add in the greater integration of scripting languages. For some time now, designers have been using JavaScript to create simple effects and mouseover events. In most cases, however, the script controlling the event was just replacing one image with another, creating the effect of a dynamic change. With DHTML, JavaScript or Microsoft's VBScript can be used to achieve true dynamic control of page elements. For example, using conventional HTML and JavaScript, a designer can make a headline appear to change when the user runs their mouse over it. What really happens is that the script replaces the image of the headline with another image. With DHTML, the same effect can be accomplished using text and changing the actual font definition tag in runtime. This is extremely important because text is loaded far faster than images are. Figure 1.3 illustrates an HTML document in which JavaScript has been integrated.

> Offered up by the W3C in January 2000, XHTML, a synthesis of HTML and XML, is the heir apparent to HTML. For more information on XHTML, go to `http://builder.cnet.com/webbuilding/0-3881-8-7080997-1.html`.




```
<html>
<head>
<title>main2.gif</title>
<meta http-equiv="Content-Type" content="text/html;">
<!-- Fireworks 4.0  Dreamweaver 4.0 target.  Created Sun Jan 07 16:59:07 GMT-0500 (US Eastern Standard Time) 2001-->
<script language="JavaScript">
<!--
function MM_findObj(n, d) { //v3.0
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document); return x;
}
function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
function MM_preloadImages() { //v3.0
 var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
   var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
   if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

//-->
</script>
<link rel="stylesheet" href="pp-styles.css" type="text/css">
</head>
<body bgcolor="#000000" onLoad="MM_preloadImages('media/main2_r2_c5_f4.gif','media/main2_r2_c5.gif','media/main2_r2_c5_f3.gif
<table border="0" cellpadding="0" cellspacing="0" width="691" align="center">
  <!-- fwtable fwsrc="main2.png" fwbase="main2.gif" fwstyle="Dreamweaver" fwdocid = "774266840" fwnested="0" -->
  <tr>
    <td><img src="media/spacer.gif" width="4" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="169" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="36" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="287" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="438" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="5" height="1" border="0"></td>
    <td><img src="media/spacer.gif" width="1" height="1" border="0"></td>
  </tr>

  <tr>
    <td rowspan="2" colspan="2"><img name="main2_r1_c1" src="media/main2_r1_c1.gif" width="173" height="65" border="0"></td>
    <td rowspan="2" colspan="2" background="media/main2_r1_c3.gif"> </td>
```

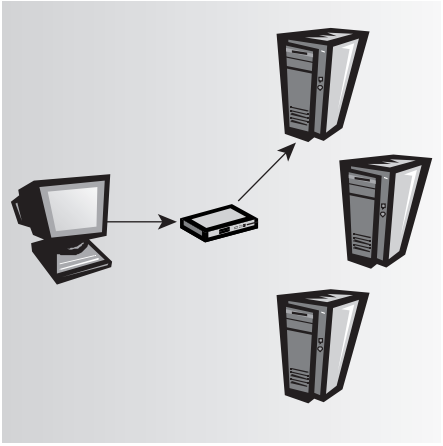
Figure 1.3

The integration of a scripting language such as JavaScript or VBScript lets designers add an incredible range of interactivity into the HTML document.

Waiter, There's a Fly in My Soup

But how does the Web *work*? Well, without getting too crazy and technical, you can think about the Web as a fancy restaurant—yeah, that's right, a restaurant.

Let's start off with a brief exploration, in very dry terms, of how the Web works, and then I'll bring the whole restaurant thing in. To start, it's important to know that the Web works on what is called a client/server relationship. A client, the web browser in this case, is a piece of software that runs on a local computer that communicates with a server in another location. The other half of the equation is the server, which is a computer that performs tasks for other computers, such as sending out e-mail. When you type a URL into a browser (or click a hyperlink), two separate technologies (HTTP and TCP/IP) are used to send a request to a router. The router, which is a piece of hardware that sits in between the user's computer and the web server, takes a look at the request and decides to which web server (there are hundreds of thousands, perhaps millions, in the world) it should be sent. From there, the request is sent on the proper server which then examines it, decides (based on the URL itself) which document is being requested, locates and retrieves the document from somewhere within its directory structure, and sends it back along the route that the request came.



On the way back, the router makes sure the requested document is sent to the client that originally made the request. The last leg happens when the client (the web browser) receives the requested document and displays it for the users' reading (or viewing) pleasure. That's the very basics of how the web works.

A little confused? Don't worry, even a description as stripped down as this can be a tad perplexing. This is where the whole restaurant analogy comes into the picture. Imagine if you were to walk into one of your favorite restaurants and were seated at your favorite table. You peruse the menu and decide upon a delectable soup to start your meal. You put down the menu and signal your waiter. When he comes to your table, you say, "Good evening my good man, I would like to start off with a bowl of gazpacho, please." He nods his head, quickly moves to the kitchen and hands the order to the head chef, who gives the order to one of his newest chefs (it *is* only a bowl of soup). After a while, the soup is made. From there, it is handed back to the head chef, who hands it to the waiter, who takes it to your table, where you sit back and enjoy the beginning of a lovely meal.

OK, let's recap: you tell the waiter what you want to eat: that is, you type a URL into a browser. From there, the browser/waiter sends your request to the head waiter, who fulfills a task not unlike a router, making sure the request gets to the proper place. After your request/order has been passed on to the appropriate chef/server, it is fulfilled, handed back to the router/head chef, who then hands it back to the waiter/browser, who finally gives it to you for consumption. Make a little more sense? Or are you craving some soup?

Browser Madness

We left off in our history of the World Wide Web somewhere in 1994. Netscape Communications Corporation had just released the first version of Netscape, affectionately nicknamed Mozilla. Not long after that, wanting to get in on the action, Microsoft released Internet Explorer, which quickly became Netscape's primary competitor. In addition, around the same time, Sun Microsystems developed a browser called HotJava, and America Online (AOL) developed one called the AOL browser.

> There are a lot more browser alternatives out there than Netscape or Internet Explorer. For a fully featured accounting, check out CNET's browser topic center at `www.browsers.com`.

It didn't take too long for things to get crazy—very crazy. The Web was becoming the hottest thing since sliced bread, and everyone wanted their piece. In an attempt to establish total market domination, the two primary browsers (Netscape and Internet Explorer) adopted a policy under which new versions were released at very regular intervals. Each time a new version was released, it often had a spate of new features (such as the ability to display new HTML content). It seemed like a pretty good situation—new browsers, new features, better

looking and more fully featured web pages. There was one slight hitch: for the most part, the new elements were unique and exclusive to the browser in which they were featured. As a result, there was tons of content floating out on the Web that was only accessible with one browser or another. This constant attempt by Netscape and Microsoft to one-up each other is often referred to as the *browser wars* and was probably one of the most disastrous periods for web design.

For a great look at the history of browser version releases and their associated features, check out Ann Navarro's *Effective Web Design* (Sybex, 2001)

Where does that leave us today? Well, a couple noteworthy things have happened. First, for all intents and purposes, Netscape lost (or, if you aren't keen on putting the last nail in the coffin just yet, is losing) the browser wars. While browser usage statistics are extremely hard to generate (and are often somewhat misleading) it looks like Internet Explorer (in its various versions and incarnations) commands between 58 percent and 65 percent of the market (perhaps even more depending on the source of the statistics), while Netscape has about 10 percent to 12 percent. The remaining slim percentage points are the domain of a myriad of different browser alternative that exist.

For more information on browser usage statistics, check out either Browser Watch at `browserwatch.internet.com` or Browser News at `www.upsdell.com/BrowserNews`.

The second noteworthy thing is that the World Wide Web Consortium was founded. Established in 1994 by many of the individuals who "invented" the World Wide Web, the World Wide Web Consortium (W3C) was mandated to develop standards that promote the Web's evolution and ensure its interoperability.

In theory, the best way to prevent the sort of madness that resulted from Netscape and Microsoft's constant attempts to outdo one another and to ensure that web pages would display consistently and legibly in *any* browser would be to develop standards to which everyone could stick. The primary problem is (and was at the time when the W3C started its Herculean undertaking) that no one owns the Web. Instead, the Web is made up of thousands of individual networks and organizations, each of which is run and financed on its own. As a result, despite the fact that the W3C is a fairly powerful entity, it has no real way to enforce the Web standards that it proposes. Ultimately, it's up to those companies that make the browsers to adopt the standards themselves. The good thing is that, even before the browser wars were winding down, both Netscape and Microsoft were adopting many of the W3C proposed standards as baselines for their browsers. As a result, one could design with the most up to date W3C standard version of HTML and be relatively certain that it would work predictably on different browsers.

The W3C is an interesting entity. Not only does it develop standards for HTML, but it is also constantly developing other cutting-edge technologies for the Web in an effort to prevent any future browser arms races. For more information about the W3C, go to `www.w3c.org`.

However, even today, you'll find that many features (especially those associated with Dynamic HTML (DHTML) display inconsistently from browser to browser. Also, despite the fact that the W3C is the king of web standards, there are companies that integrate unique features into their browsers.

Designing for Today's Web Browser (and Yesterday's, While You're at It)

What's the moral of this story? Well, despite the fact that Microsoft's Internet Explorer seems to be reigning supreme (at least for now), lots of people are using lots of different browsers to surf the Web. If you want to ensure that the maximum amount of people can enjoy your website, you need to stick to the W3C's standards. One of the coolest thing about Dreamweaver is that it was built to design web pages that conform to these standards. Still, it's important that you constantly and consistently test your HTML creation in all of the browsers you foresee it being viewed with.

For more information on how you can use Dreamweaver to preview web pages in different browsers, see the "Previewing Your Work in a Browser" section of Chapter 3, "Setting Up and Managing Your Page."

Fundamentals of Interactive Design

Strictly speaking, the kinds of fundamental issues that you need to address when you're designing your website are relatively universal (at least when it comes to interactive design). Whether you are creating a CD-ROM, a DVD-ROM, a website, an interactive kiosk, or any information-driven application, you need to create something that fulfills the needs of your audience. From the designer's perspective, it would be great if everything created was intended solely for the consumption of the individual who created it—an audience of one so to speak. However, this is far from the case. Even the most insignificant of interactive creations is usually intended for a wider audience than just the designer. Because of this, you need to create something that, in the best case, is all things to all people. As such, there are some *extremely* important issues you should consider during the creative process. This section explores some of these issues.

It's important to remember that what is presented here is only an introduction to the fundamentals of interactive design. However, the discussion will provide you with the necessary foundations upon which to further your own exploration of the topic.

Information Design and Architecture

Information architecture has absolutely nothing to do with traditional architecture. In this section of the chapter, you won't learn how to tell the difference between an Ionic and a Doric column. (This was a big disappointment to one of my Web Design students who thought she could indulge her passion for Renaissance architecture in my class!) Instead, you are going to get an introduction to the art and science of information design and architecture for the Web.

Essentially, information architecture is the process by which a website's content is organized into easily accessible components that support a wide variety of user access techniques (casual browsing, direct searching, and so on). As one would expect, information architecture is intimately related to the navigational system of a site. As a result, you'd have a difficult time designing one without the other.

> This section of the chapter is designed to be a simple introduction to information architecture. Extra resources are a must. You might want to seek out Louis Rosenfeld and Peter Morville's *Information Architecture for the World Wide Web* (O'Reilly, 1998). If you are interested in extending your search to the Web, check out the Argus Center for Information Architecture at `www.argus-acia.com`.

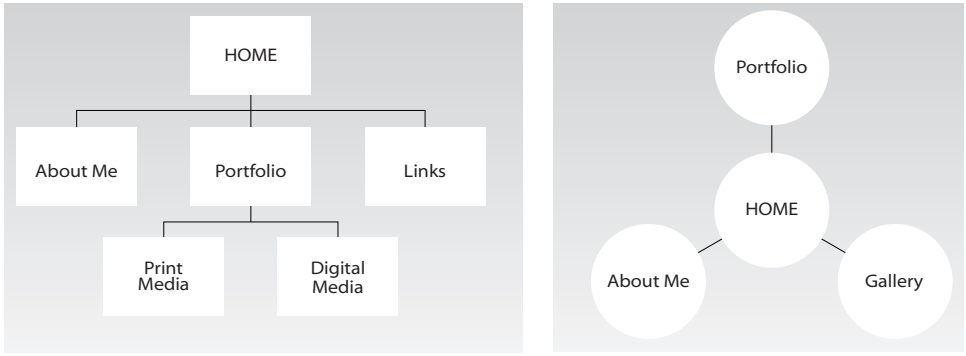
At its most basic, the process by which you develop a site's information architecture is usually a two-step affair. The first step involves organizing your site's information into a variety of categories. You'll need to decide how many sections, subsections, and categories you need; how your site's content will fit into those categories; and how the units of information (individual web pages) will relate to units of information within the same category and to units in other categories. This is arguably one of the most difficult things about designing a site. The way we human beings organize information is largely determined by our cultural context. As a result, what constitutes well-organized information is *extremely* subjective. One must contend with all manner of problems such as linguistic ambiguity (a word or term may mean different things to different people) or different perspectives of how information should be organized.

As with many other issues in web design, the way you organize information will depend largely on your audience. You must put yourself in their shoes and predict the best way to develop an information architecture so that your users' needs are met. Developing a way to organize your content can often be a fairly painful process, but it is absolutely necessary in developing a functional and usable information architecture.

Once you've decided how to organize your information, you need to formalize that structure with something called an information architecture diagram (IA diagram). Designed to provide a visual representation of the information architecture of your site, an IA diagram can take many different forms (see Figure 1.4). Ultimately, the specific style is up to you.

Figure 1.4

There are many different ways to create an information architecture diagram. The example on the left is a reverse branching tree model; the example on the right is the spherical model.



An IA diagram isn't supposed to show the links between various pages within your site. Instead, it represents the hierarchical relationship between sections/subsections, and individual pages.

An IA diagram is also very useful for developing the structure of your site on the server (folders, subfolders, and so on).

Developing a Visual Metaphor

A visual metaphor is often a pretty slippery concept to put into words, but it can be easy to unconsciously interpret when it's used properly. Essentially, a visual metaphor, which is universally applied to an entire website, leverages familiar visual elements (such as images, interface elements, icons, colors, or fonts) to unconsciously reinforce the site's subject matter. The visual metaphor for a major Hollywood movie's promotional website will be completely different from that of an interactive design firm or an online merchant.

For example, in the screenshots depicted in Figure 1.5, the goal was to create a website for the Glenn A. Black Laboratory of Archaeology, a highly prestigious independent research unit at Indiana University, Bloomington. Because of the archaeological theme, the website used an earthy palette of colors consisting of shades of brown, gray, rust, dark green, and tan, combined with archaeologically oriented design elements to visually reinforce its content.

Creating an effective visual metaphor requires some serious brainstorming and inspired free association. You'll need to sit down and think about what kinds of colors, fonts, images, icons, and interface/layout elements unconsciously reinforce the site's content.

Before you even begin brainstorming for your visual metaphor, you need to be rock-solid sure of your audience. You can't effectively design a visual metaphor if you aren't exactly sure of the demographic of those who'll be using the site.

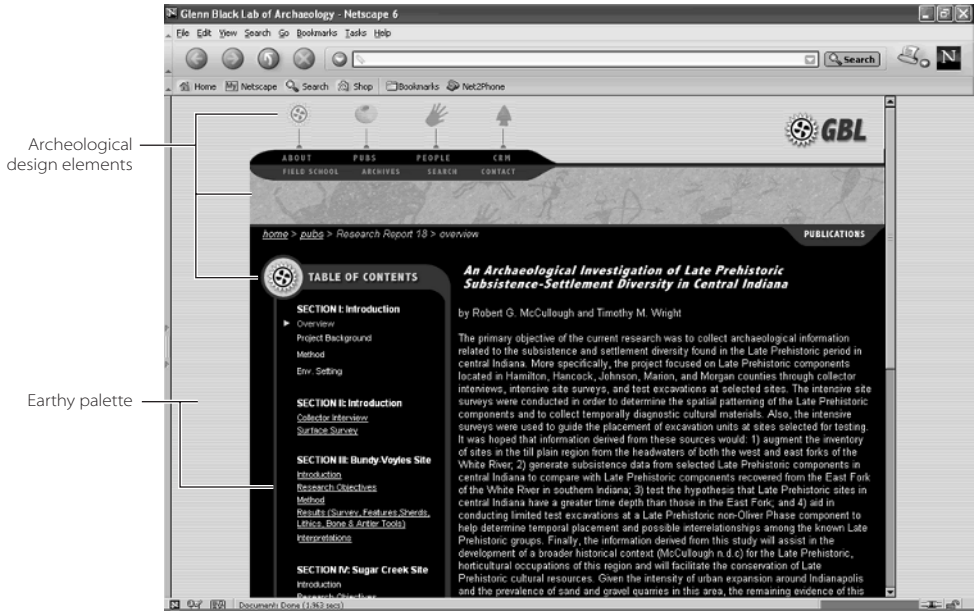


Figure 1.5

The Glenn Black Lab of Archaeology's website employs earth tone colors and archaeologically oriented imagery to create a solid visual metaphor.

Say, for example, you are creating a children's online community site geared towards ages 7 to 10. You might think about using bold primary colors with cartoony interface elements and fonts. On the other hand, the website for a movie would draw its visual metaphor from the film's look and feel (check out the Planet of the Apes website at `www.planetoftheapes.com` for a great example).

There are no hard and fast rules for creating a visual metaphor. The only real guideline is that they should be used wisely: be subtle and don't overdo it. Always have your ideas vetted by individuals not associated with the project; they will often have suggestions or comments you never thought of. Don't get too attached to a visual metaphor because it's quite possible that, given an outside opinion (perhaps one of a prospective user), you'll decide that it doesn't work for your site.

Creating Storyboards and Concept Art

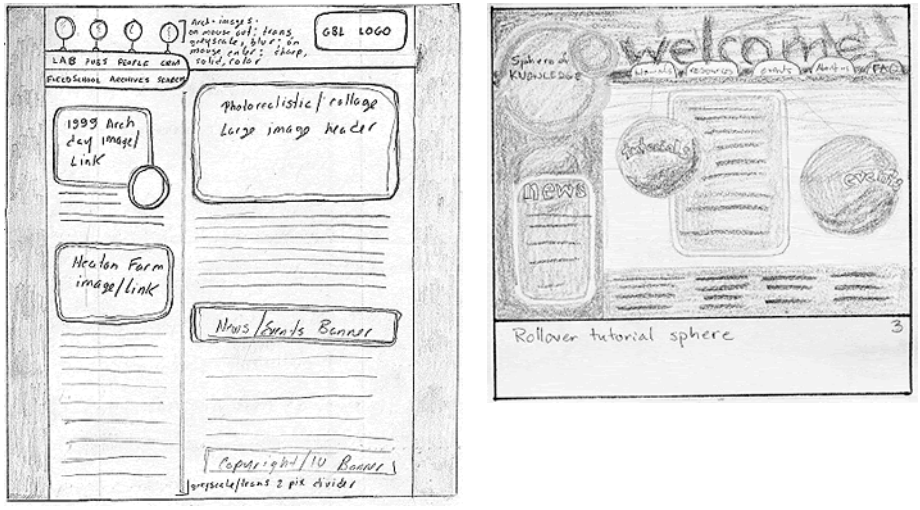
One of the most important steps in the paper and brain predigital production process is creating storyboards or concept art, an idea swiped from the film industry. Generally one of the last steps before you go digital with your grand creation, storyboards are used to visualize your design as a complete entity. With them, you get a chance, among other things, to see how colors interact with one another, how interface elements play off one another, how your navigational system is realized, how your visual metaphor plays out, and whether content is

represented in the best way possible. Storyboards provide you with a painless way of catching any potential design problems before you get to the stage where you build your design in HTML and they become major obstacles. Storyboards are also a great way to play with design ideas and visually brainstorm.

As illustrated in Figure 1.6, there is no hard or fast rule as to how they should look. If you are brainstorming ideas, the back of a cocktail napkin is a good a medium as any. However, if you are preparing a pitch to a potential client, it's a good idea to come up with something more polished and formal. The bottom line is that storyboards, in whatever form they appear, should efficiently communicate your design ideas without too much ambiguity.

Figure 1.6

Storyboards can range from "quick and dirty" (left) to quite formal and polished (right). Whatever their level of quality, they should effectively communicate your design ideas.



You may even want to create your storyboards in a photocopy of an empty browser window. This is a great way to give your client the necessary context.

Several browser templates for the purpose of storyboarding have been included in the Storyboard Templates folder of this book's accompanying CD.

Getting From Here to There: Developing Intuitive Navigation

One could easily argue that designing an intuitive and usable system of navigation is one of—if not *the*—most important goals when it comes to web design. User experience on the web is all about moving in space from one location to another in search of *something*. Whether or not the user knows what they are looking for is moot. It's up to you to crawl inside the heads

of the users, figure out what they want from your site, and then figure out the easiest way for them to get it. If you don't provide a system for them to get from where they are to where they need to be, they'll go elsewhere, and this is the last thing you want.

Don't be fooled into thinking that a navigation scheme is simply buttons and hyperlinks. The best-designed navigation is a highly artful mix of many different things: a pinch of interface design, a dash of information architecture, and a generous dollop of psychology.

As I've mentioned, there is no way that I could effectively condense all that you need to know about designing intuitive navigation into one section of one chapter of one book. However, there are certain general, basic concepts that are both fundamentally important and self-contained enough that they can be discussed.

Keeping Things Consistent

One of the ways human beings define the world around them, and the way which they interact with it, is based on the consistency and predictability of events. When a navigational system works properly, people come to unconsciously rely upon it. For this to happen, the navigational system must be consistent. This means (as shown in Figure 1.7) that the menu must remain in the same location, it must retain the same appearance and contents, and the interface where the navigational elements reside must not change to any significant degree.

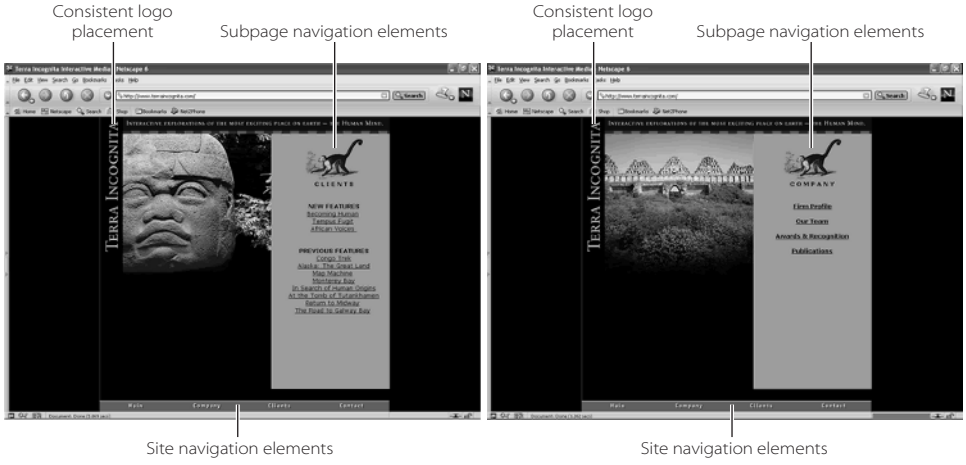


Figure 1.7

These two screenshots are of two different pages within the same site. Note that the navigational elements remain exactly the same.

One of the obstacles to designing a consistent and predictable navigation system revolves around the interplay between navigational elements and interface design at deep levels within the website. Often, as one gets deeper and deeper into a site, a certain point is reached at

which the navigation scheme breaks down due to a lack of foresight. Designers tend to put most of their effort into developing a navigation scheme that will work best in the more consistently accessed areas of a site. As they move deeper and deeper into their site and the amount of information in any given screen increases, they spend less and less time ensuring that the navigation scheme they developed will function properly.

The best-designed websites have a pyramid-shaped information distribution. The top levels of the site contain information that doesn't take up a great amount of screen real estate. As you move deeper into the site and into more specialized information, a larger amount of screen real estate is consumed by the website's content.

It's at this point that chaos often sets in and the all important consistency and predictability goes out the window. To cope with the additional information, designers will toss in additional navigational elements (menus, buttons, and so on) or even alter the existing navigational scheme that worked just fine in the upper levels of the site.

Instead of succumbing to bedlam and anarchy, make sure that when you create the navigational scheme, you think deep into your site's structure. It may seem time-consuming, but it could save you valuable time later. Because your user will probably spend more time in the deeper sections of your site, ask yourself whether what you've laboriously designed will work just as well with the content in the upper sections of your site as with the content in the deeper sections of your site. If you can't answer with a resounding "yes," start again.

Remember, the Web is a nonlinear medium. Users don't always enter your site through the front door: they can just as easily enter through a side or back door into a section deep within your site's hierarchy. Because of this, you should make sure that your entire site maintains a consistent scheme of navigation.

Help Users Quickly Learn Your Navigation Scheme

When you create a website of any kind, you are providing something to your user. Whether it's a mega online bookstore like Amazon.com, a major educational institution like the University of Toronto, or your own personal corner of cyberspace, content is king. You don't want users to have to spend a huge amount of time learning how to locate what they desire. In other words, you don't want an overly complex navigational system to stand in the way of the user and what they want.

The key to easily learned navigational systems lies in several different issues. First, as I just mentioned, your navigational system should be consistent. If you switch the way you require your user to move about the site, they'll have to start from scratch and relearn your navigational scheme: not good. Second, as will be covered shortly, make sure the way your user identifies navigational elements (labels, visual imagery, and so on) is straight to the point and not overly complex or confusing. There is nothing worse than a series of buttons whose labels make no sense. The general rule of thumb (and this is pretty general as rules go) is to create navigational schemes that are not counter-intuitive and thereby difficult to learn.

Providing Clear and Obvious Visual Cues

Because the web is a visual medium, effective navigational schemes should provide clear visual messages. I'm not just talking about the buttons here. Integrating clear visual cues into a navigational scheme requires some very broad (and often subtle) thinking.

COLOR

One of the best ways to provide your users with a quick and easy (and often unconscious) method of identifying exactly where they are located in your site is to use color. You may have noticed that many large sites use a consistent navigational system whose color changes slightly depending on the section or subsection where the user currently resides. This is a very effective technique that, when used properly, creates "signposts" for the user that are easily learned and recognized. When using color to increase the usability of your navigational system, you've definitely got some options. Changing the background color of individual pages is one way, but you can also use color for subtle emphasis—highlighting certain navigational or interface elements like buttons, banners, or header graphics.

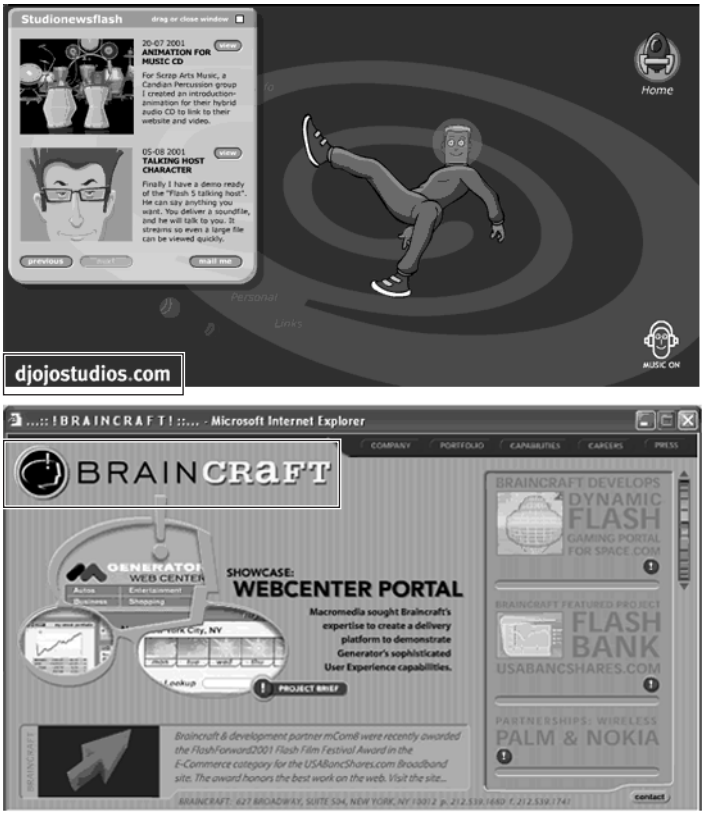
There are, however, some caveats to using color in this way. First, to avoid overwhelming the user with a new color for each subsection, pick a very limited palette and apply those colors to the top level sections of your site. For example, say you're designing your own personal website and you use a nice light rust color for the "About Me" section. To avoid overwhelming the user, you'd also use that color for the "My Favorite Movies," "My Family," and "My Favorite Music" sections, all of which are subsections of the "About Me" page. You should also carefully choose a palette of colors that fits with your visual metaphor.

BRANDING

Consistent and clear branding is also a good way to provide your audience with visual cues. Given the nature of the Web, people have a tendency to quickly jump from site to site with mouse clicks. When your audience is cavorting about your site, you want them to know *exactly* where they are. This is best accomplished with clearly and consistently placed logos, as shown in Figure 1.8.

Figure 1.8

Notice that in both websites, a logo is always prominently displayed to remind the audience exactly where they are.



BREADCRUMB TRAIL

One of the biggest problems in particularly large, content-heavy websites is that people can easily get lost and end up with no clue where they are located, and no idea how to get back to where they were several clicks ago. One of the easiest and most elegant ways (and most cost effective in terms of effort and screen real estate consumed) to work around this problem is to create a simple navigational tool called a breadcrumb trail (Figure 1.9). Essentially, a breadcrumb trail (sometimes referred to as a link buildout) is a horizontal line of hyperlinked words indicating the location of the current page within the site's overall information architecture. An example of a breadcrumb trail would be something like Home → About Me → Favorite Movies. Each item in the trail would be a hyperlink to that specific section or subsection.

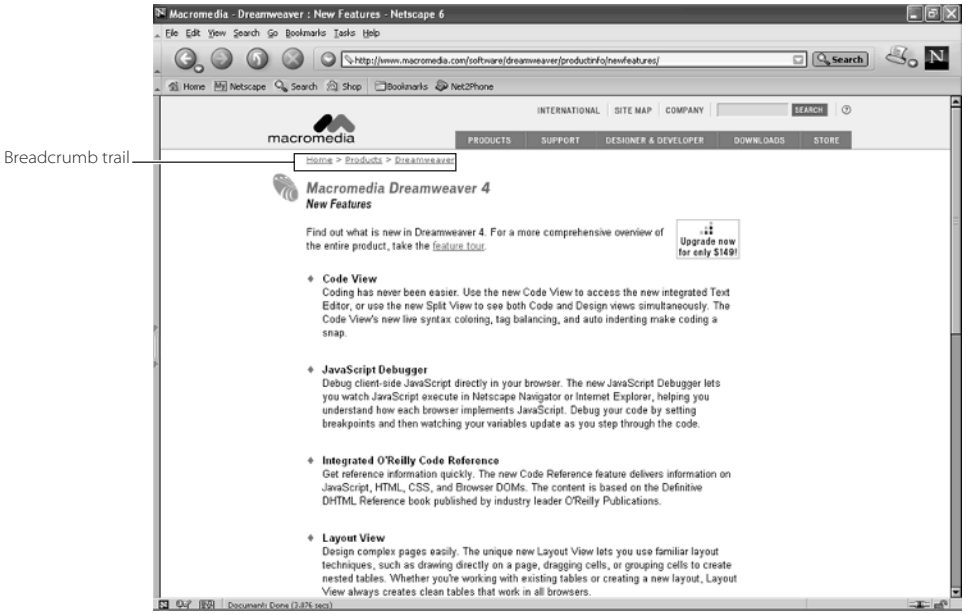


Figure 1.9

A breadcrumb trail provides a clear indication of the position of the current page within the site's overall structure; it also provides an easy way of moving back up that particular section or subsection's hierarchy.

LABELS

One of the most often overlooked methods to provide clear and concise visual cues to your audience is to use effective labels. We're not just talking about ordinary labels here, we're talking about concepts that have been boiled down to their basic understandable components. For example, suppose a section of your site had images of all the photographs you've taken, all the paintings you've painted, and all the sculpture you've sculpted. Instead of having a link or a button that said "Everything I've ever created on film, with canvas, or with clay," you could simply use the word "Portfolio" or "Gallery."

> It's important to remember that many of the conventional web labels used are culturally based. While your average web user in North America wouldn't have any difficulty understanding your intentions, if you used the word "Home" in your navigational scheme, someone from Egypt, the Czech Republic, or Malaysia may have absolutely no clue to what you are referring.

When creating clear labels, you must avoid using what I call *geek speak*, or terminology familiar only to those individuals within a specific field. For example, as an archaeologist, I've created websites where the term "Gray Literature" is used. If you have no experience

in the field of archaeology, you probably don't have a clue what gray literature is. However, there are individuals out there who, despite the fact that they aren't familiar with the strange terms we archaeologists use, would be interested in gray literature. (Gray literature refers to the excavation reports generated by federally mandated salvage archaeological excavations.) To avoid geek speak in this particular situation, I substituted "Gray Literature" with the term "Excavation Reports," which is a lot more understandable to the general public.

VISUAL VOCABULARY: NAVIGATIONAL ELEMENTS

Another good way to provide users with clear visual cues is by using consistent and universally understandable visual vocabulary. There are three general schools of thought when it comes to creating navigational elements (buttons, menus, and so on). The first one tends to emphasize the use of icons or imagery, while the second one emphasizes the use of purely textual-based navigation. The third, which I think is the most rational, encourages the appropriate and contextually suitable use of both text and images as navigational elements.

If you've decided to use a purely visually based navigation system, you are in for some serious obstacles. Using icons or images in navigation is fine, but you must realize that the Web itself has no real standardized conventions for visual vocabulary. So, for instance, if you've created a button on your main page that links to your "About Me" section, what icon do you use? The possibilities are literally endless. The problem especially pops up when you choose an icon that, while significant to you, has no significance for your audience. In this situation, your audience will be faced with a series of acontextual (at least for them) icons with which they are expected to navigate your site. The only true universal solution for this problem is to create navigational elements that incorporate both text and images. If you include text that answers the user's "what the heck is this button for" kinds of questions, you can use funky icons that fit into your visual metaphor.

Multiple Roads from Here to There

Lots of people do things in lots of different ways. People drive differently, have different tastes in movies and music, talk differently, eat differently, and most important to this discussion, use different methods to move about the Web. Some like to wander aimlessly until they stumble across something interesting, while others want to locate specific information as quickly and efficiently as possible. Some people use the newest browser on a fast machine with a fast Internet connection, while others have an older browser on a slower machine with a slow Internet connection. Some people use text browsers or screen readers. Get the point? It's up to you to try to accommodate all of these "profiles" so that you don't alienate possible visitors.

A screen reader is a piece of software that "reads" the content of a website for those who are visually impaired or blind.

To ensure that your website is accessible to as wide an audience as possible, you have to create a navigational system which supports many different personal styles. For instance, provide a low bandwidth version for those whose Internet connection and computer is on the slower side. You can also employ a series of different tools, such as a search feature, a site map, and a traditional text and icon menu so that a user can choose how to move about your site.

Designing with Bandwidth in Mind

The way users connect to the Internet plays a major role in how websites should be designed. Remember, when you view a web page in your browser, you are downloading the HTML file and all the associated media (images, Flash movies, and so on) onto the hard drive of your computer. As a result, the size of the file and speed a web page downloads is directly proportional to the speed of Internet connection the user is employing.

I'm sure you've heard your share of "bandwidth this" and "bandwidth that" in the media. What exactly is bandwidth, anyway? Basically, it's a measure of the amount of data that can be sent across an Internet connection over a certain unit of time (second, minute, and so on).

Bandwidth is usually measured in terms of KBps—kilobits per second. Most standard modems range from 14.4 to 56KBps, while high-speed connections (such as cable models, DSL, or T1 lines) range from 64 to 1500KBps (and even higher).

An Internet connection is the method by which you connect to the internet itself and can range from a modem to DSL (digital subscriber line). Each method has a different level of bandwidth, which can range from very low (downloads data from the Web slowly) to very high (downloads from the Web quickly). You can think about your Internet connection as a pipe through which the material you're downloading is shoved. A faster connection means a larger pipe, which means more stuff can be downloaded at a quicker rate. A slower connection, on the other hand, means a smaller pipe, which means things are downloaded at a far slower rate.

So, what does this all mean to you as a designer? Like all other file types, HTML files consume a specific amount of computer memory. Granted, because HTML files are plain text, they are usually quite small. However, when one starts adding images and other multimedia (Flash and Shockwave movies, audio, or digital video, for example), a web page can get quite

large. The larger a web page is, the longer it will take to download. As a result, particularly multimedia-rich websites are often time consuming to download for those users who don't have high speed connections. Would *you* want to hang around, waiting to download a web page whose contents are pretty much unknown? Probably not. That's why you should put a lot of thought into the ultimate size of the web pages you create.

> You might think everybody has fast connections these days. While the amount of people connecting to the Internet with high-speed connections is growing by leaps and bounds, dial-up modems still reign supreme. Some statistics show that in 2001, more than 17 million people in the United States were connecting to the Internet with high-speed connections, while about 64 million were using 56K modems, 15 million were using either 28.8K or 33.6K modems, and about 3 million were using 14.4K modems. (These statistics are just for the United States. Statistics for Canada and Europe, which are more "wired" than the U.S., were not included. Also, don't forget the rest of the world!)

As a designer, you've got to make a decision about your audience. Do you want to include cool, bandwidth-consuming features such as Flash, Shockwave, or digital video and run the risk of alienating those people who have a slower connection? Or do you want to curtail your multimedia leanings and design for the most common denominator (which, given the numbers, is probably a 56K modem)? Ultimately, it's all about balance: cool features for limited audience, or not-as-cool features for a wider audience?

If you plan on creating a site more suitable for high-bandwidth connections, it is vital that you give your user some indication of this early on in their navigation of your site (usually on the top-most page). This way, those individuals with low-bandwidth connections don't get unwittingly stuck having to download an overly bandwidth-intensive site. One way to work around the problem of low-bandwidth users accessing a high-bandwidth site is to create two versions of your site: one for high bandwidth access and one for lower bandwidth access.

Inspiration: Design and Technique

Based in the great city of Vancouver in the wonderful province of British Columbia in the beautiful country of Canada, Atomic Cartoons specializes in creating offbeat and hilarious Flash-animated shorts for distribution over the Web. Entertaining the masses, the Atomic Cartoons website features incredibly well-engineered graphics with stylish simplicity that scream creative talent (see Figure 1.10).

The beautifully constructed visual elements used in the site indicate that the good folks at Atomic Cartoons have a keen understanding of all aspects of interactive design.

Figure 1.10
Atomic Cartoons

Summary

You've covered a lot of ground in this chapter. You should now have an idea of some of the more important web design issues and topics, so it's time to dig into Dreamweaver. In the next chapter, you'll be taking your very first steps in Dreamweaver, starting off on the right foot by exploring the creative environment where you'll be spending a fair amount of time.