McGraw-Hill Osborne

Online support at webdesignref.com Includes tools and samples for use on your own site

The Complete Reference Web Design

Second Edition

Provides up-to-date Web design solutions for home and corporate users

Covers modern Web building practices, and features practical and focused examples

Explains common design theories, techniques, technologies, and practical applications

Thomas A. Powell

Instructor, UCSD Computer Science Department, and best-selling author

Web Design: The Complete Reference Second Edition

About the Author

An Internet professional for numerous years prior to the introduction of the Web, Thomas Powell brings an interesting combination of networking and technical expertise to the Web design community. In 1994 he founded PINT, Inc. (www.pint.com), a web development firm with headquarters in San Diego, which serves numerous corporate clients around the country.

Powell is also the author of numerous other Web development books, including the bestsellers: *HTML: The Complete Reference*, *JavaScript: The Complete Reference*, and *Web Site Engineering*. He also writes frequently about Web technologies for *Network World* magazine.

Mr. Powell teaches Web design and development classes for the University of California, San Diego Computer Science and Engineering Department, as well as the Information Technologies program at the UCSD Extension. He holds a B.S. from UCLA and an M.S. in Computer Science from UCSD.

About the Technical Editor

Fritz Schneider is a software engineer at a major Internet search engine. He is co-author of JavaScript: The Complete Reference and served as the technical editor for HTML: The Complete Reference. Schneider holds a B.S. in Computer Engineering from Columbia University and an M.S. in Computer Science from UC San Diego.

Web Design: The Complete Reference Second Edition

Thomas Powell

McGraw-Hill/Osborne
New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

McGraw-Hill/Osborne

A Division of The McGraw-Hill Companies

0-07-222851-2

The material in this eBook also appears in the print version of this title: 0-07-222442-8

TERMS OF USE

DOI: 10.1036/0072228512

Contents at a Glance

Contents

Acknowledgments

When you take the time out of your life to write a doorstop-sized book like this one, you tend to rely on a lot of people's assistance. I'll mention only a few of them here to avoid adding any more pages to this already massive tome.

First, as always, the folks at Osborne were a pleasure to work with. Megg Morin somehow puts up with me year after year and the books keep getting done. Without Megg, I probably wouldn't be a prolific author. Tana Allen also provided great assistance in editing and project management, while Carl Wikander provided rigorous copy editing. Finally, thanks to Julie Smith for enduring the ghastly long phone calls necessary to help ferret out proof problems.

My technical editor Fritz Schneider did an excellent job. Having seen the other side of the fence as my co-author on the *JavaScript: The Complete Reference*, he didn't let me get away with much.

My employees at PINT provided dozens of right hands for me and deserve special mention. First, Mine Okano has helped run another book project and done an excellent job at it. I am not sure she expected this when she came to work for me, but she's done a great job. Dan Whitworth also continues to tackle book projects, and probably wonders what he did in a previous life to deserve fixing my poor grammar. Catrin Walsh and Kim Smith lent some valuable assistance in the site production, usability, and site testing content. Other PINT employees, including Jimmy Tam, Rob McFarlane, Maria

Defante, Eric Raether, Cathleen Ryan, Meredith Hodge, Nigel Paxton, David Sanchez, Dave Andrews, Melinda Serrato, Michele Bedard, Candice Fong, Cory Ducker, Anh Gross, Kun Puparussanon, Huijuan Yin, Marcus Richard, Kevin Griffith, Christine Lawson, and numerous others, helped out directly on edits or just kept the projects rolling while I was busy. Joe Lima, Allan Pister, Christie Kennedy, and Jared Ashlock deserve some praise for getting some of my outside software project duties taken care of as well.

The students in my undergraduate and extension classes always make good points and many of their ideas are incorporated into new editions. Daisy Bhonsle deserves to be singled out for always helping with my books by catching errors and making suggestions for improvements.

Somehow I find a way to have time outside of the Web for friends, family, and home. My wife Sylvia made sure I didn't work all day on the weekends, and our Schnauzer puppy Tucker kept both of us very busy with his antics. However, now that I'm finished with the book and the dog is housebroken, with any luck a restful Wakaya visit or other trip will be up next.

Finally, the most thanks go to the thousands of readers around the world who have purchased my various Web technology and design books. It is really a great pleasure to get such positive feedback and see folks putting this information to good use.

Introduction

A thick Web design book without glossy paper and pictures! Who would have thought it would be published? That's exactly what I set out to do a few years back and it seemed to make sense to enough readers that now it has even been massively updated. Why engage in such a fool's errand? Simply because there are plenty of Web design books out there that provide color snapshots of well-implemented sites or short discussions of the cool features in today's trendy sites. However, given the fluid nature of the Web, the interesting sites have often changed by the time the ink has dried on the pages, leaving only a paper record of what the site used to be like. Worse yet, what *is* left only tells part of the story. It often hides the usability problems, the technical execution problems, and the slow loading pages. Even so, I often turn to such resources as they provide a great deal of visual inspiration. But they tell only half the story—and I will try to tell the other half in this book.

The goal here is to talk about what makes sites work beyond the trends of the latest font or visual treatment. Usability will certainly be a major concern, but so will correct construction. I'll try to speak from the experience I gained from building hundreds of sites over the years with my firm. Some of the projects worked well and others didn't, and I found that I learned not only from my successes, but also from the failures of both my own projects and those I have observed or rescued. Experience is truly the best teacher in an industry as young as Web design. I'll try to make sure to teach the fine balance between designer wants and user needs, between form and function, and between uniqueness and consistency, all while respecting what is possible to execute in the chaotic medium known as the Web.

After reading this book, you'll truly appreciate how Web design is a fluid mixture of art and science, inspiration and execution, and ultimately, of frustration and elation. You may excel on the visual side of a site only to fail in the technology or delivery aspects. Web design is all-encompassing and the investment in understanding deeper medium and technical issues will pay huge dividends in future projects.

Yet as you read this book, you might not always agree with what I have to say. You may even find that some of the rules and suggestions are not perfectly consistent. However, that may be the point—to get you to think and not dismiss something out of hand. Instead, ponder why such rules and suggestions were developed before you throw caution to the wind. Great designers, regardless of medium, bend or break established rules on purpose. Real breakthroughs rarely come due to ignorance or arrogance.

Unfortunately, I won't be able to guarantee a proven step-by-step process that ensures a great Web site. Some things really do take practice. Building numerous sites and browsing even more sites is required to excel at Web design. However, I can say that if you do read this book, you'll have at least half of what you need to make great sites. The rest will be up to you and your creativity. So get out there and show the Web what you can do!

Using This Book

The book is used as a textbook for a course in Web design theories and practices as well as a reference book. The first section provides foundation information about common Web design principles, usability issues, core Web technologies, and development practices. The second section focuses primarily on site organization, navigation, and usability concerns. The final section addresses execution issues with focus on best practices. The appendices of the book provide compact reference material on HTML, CSS, fonts, colors, and other Web issues. Such an organization should make this book not only useful to understand major Web design issues, but to keep around for future consultation. The Web site at www.webdesignref.com provides support for the book including examples, reference materials, related links, and of course errata. More novice Web designers should read the book sequentially as chapters build on one another. However, experienced designers may find that single chapters or sections can be read safely in isolation if they are familiarizing themselves with a particular topic or attempting to fill in knowledge gaps.

The text does assume that readers are fairly fluent in core Web technologies like HTML, CSS, and JavaScript and can use basic graphics manipulation tools like PhotoShop or Fireworks. Readers interested in better understanding the core Web technologies may find *HTML: The Complete Reference (*www.htmlref.com) and

JavaScript: The Complete Reference (www.javascriptref.com) also useful. The three books together provide a complete discussion of the theory and execution of the popular client-side Web technologies that are not tied into the use of a particular Web tool. Tutorial books on the various editors and other Web tools can of course be utilized in conjunction with any of the books.

Good luck to you!

Thomas A. Powell
tpowell@pint.com
Summer 2002

The Complete Reference

Web Design

Part I

Foundation

This page intentionally left blank.

The Complete Reference

Web Design

Chapter 1

What Is Web Design?

Most discussions of Web design get off track in short order, because what people mean by the expression varies so dramatically. While everyone has some sense of what Web design is, few seem able to define it exactly. Certain components, such as graphic design or programming, are a part of any discussion, but their importance in the construction of sites varies from person to person and from site to site. Some consider the creation and organization of content—or, more formally, the *information architecture*—as the most important aspect of Web design. Other factors—ease of use, the value and function of the site within an organization's overall operations, and site delivery, among many others—remain firmly within the realm of Web design. With influences from library science, graphic design, programming, networking, user interface design, usability, and a variety of other sources, Web design is truly a multidisciplinary field.

Defining Web Design

There are five areas that cover the major facets of Web design:

- **Content** This includes the form and organization of a site's content. This can range from the way text is written to how it is organized, presented, and structured using a markup technology such as HTML.
- **Visuals** This refers to the screen layout used in a site. The layout is usually created using HTML, CSS, or even Flash and may include graphic elements either as decoration or for navigation. The visual aspect of the site is the most obvious aspect of Web design, but it is not the sole, or most important, aspect of the discipline.
- **Technology** While the use of various core Web technologies such as HTML or CSS fall into this category, technology in this context more commonly refers to the various interactive elements of a site, particularly those built using programming techniques. Such elements range from client-side scripting languages like JavaScript to server-side applications such as Java servlets.
- **Delivery** The speed and reliability of a site's delivery over the Internet or an internal corporate network are related to the server hardware/software used and to the network architecture employed.
- **Purpose** The reason the site exists, often related to an economic issue, is arguably the most important part of Web design. This element should be considered in all decisions involving the other areas.

Of course, the amount each aspect of Web design influences a site may vary according to the type of site being built. A personal home page generally doesn't have the economic considerations of a shopping site. An intranet for a manufacturing company may not have the visual considerations of a public Web site promoting an action movie. Precisely what is meant by the expression "Web design" seems to be fluid; our discussion must take this into account, but at the same time provide ideas concise enough for the designer to keep in mind at all times. We'll start first with abstract definitions and get more concrete as we move on.

The Web Design Pyramid

One way to think of all the components of Web design is through the metaphor of the Web pyramid shown in Figure 1-1. Content provides the bricks that build the pyramid, but the foundation rests solidly on both visuals and technology, with a heavy reliance on economics to make our project worth doing.

As Web designers, we try to plan our sites carefully, but construction is difficult. The shifting sands of Web technology make it challenging to build our site; construction requires teamwork and a firm understanding of the Web medium. Even if we are experts able to construct a beautiful and functional Web site, our users may look at our beautiful construction with puzzlement. Designers, or their employers, often spend more time considering their own needs and wants than those of the site's visitors. Our conceptual Web pyramids may become too much like brick-and-mortar pyramids—impenetrable

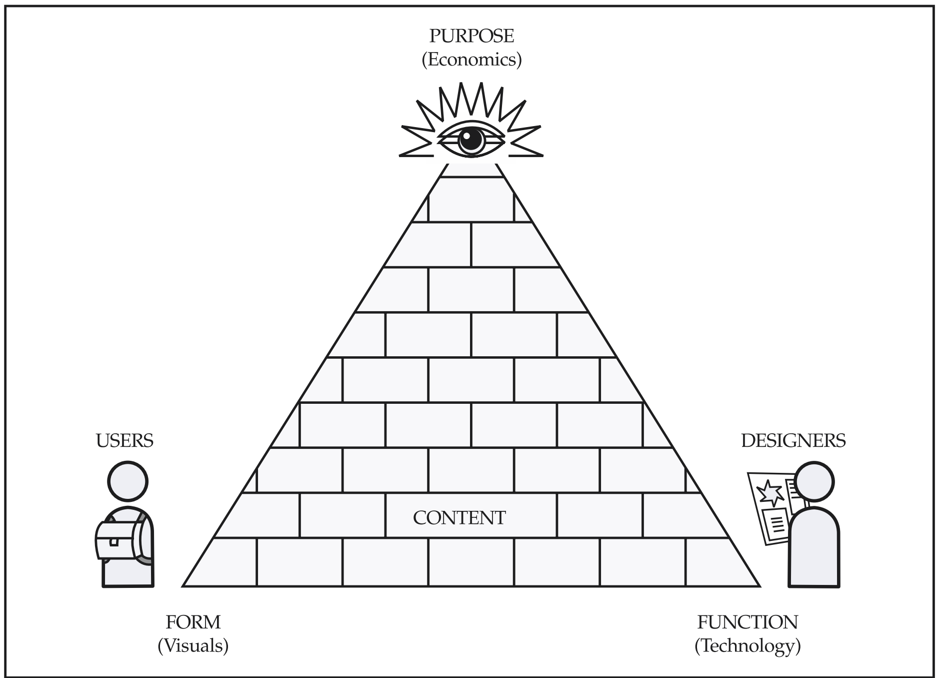


Figure 1-1. *Web pyramids: the facets of Web design*

tombs that leave us wondering if the users who strike out over the Web to reach our monuments can even find the door. Do they even understand the point of the site?

While Web development challenges aren't quite on the level of those faced by the ancient Egyptians, building a functional, pleasing Web site that can stand the test of Internet time is certainly not easy. The pyramid provides a simple way for designers to think of all aspects of Web design in interplay, but does little to provide a deeper understanding of the Web medium.

The Medium of the Web

While the Web pyramid analogy is a very abstract way of describing Web design, it is a useful tool for showing the interplay of the various components of Web building. A more practical way to discuss Web design is to think of the various components of the Web medium, as shown in Figure 1-2.

Today's Web sites are primarily a basic client-server network programming model with three common elements:

The server-side This includes the Web server hardware and software as well as programming elements and built in technologies. The technologies can range from simple CGI programs written in PERL to complex multi-tier Java based applications and include backend technologies such as database servers that may support the Web site.

The client-side The client-side is concerned with the Web browser and its supported technologies, such as HTML, CSS, and JavaScript languages and ActiveX controls or Netscape plug-ins, which are utilized to create the presentation of a page or provide interactive features.

The network The network describes the various connectivity elements utilized to deliver the Web site to a user. Such elements may be the various networks on the public Internet or the private connections within a corporation—often dubbed an *intranet*.

Complete understanding of the technical aspects of the Web medium, including the network component, is of paramount importance in becoming a great Web designer, and much of this book will focus on these details. The Web pyramid diagram again reminds us of the important user component, as Web design really is a networked programming pursuit with certain user-focused issues.

Web sites are used as a communication mechanism between a site's owners and its users, and occasionally between its users and each other. Site owners usually set the message and define the basic rules of interaction, while users are those who visit the site and attempt to use the content or facilities presented there. The communication path between site owner and visitor can vary. Site owners often set information for users to consume, in somewhat of a one-way interaction. Other times users can post

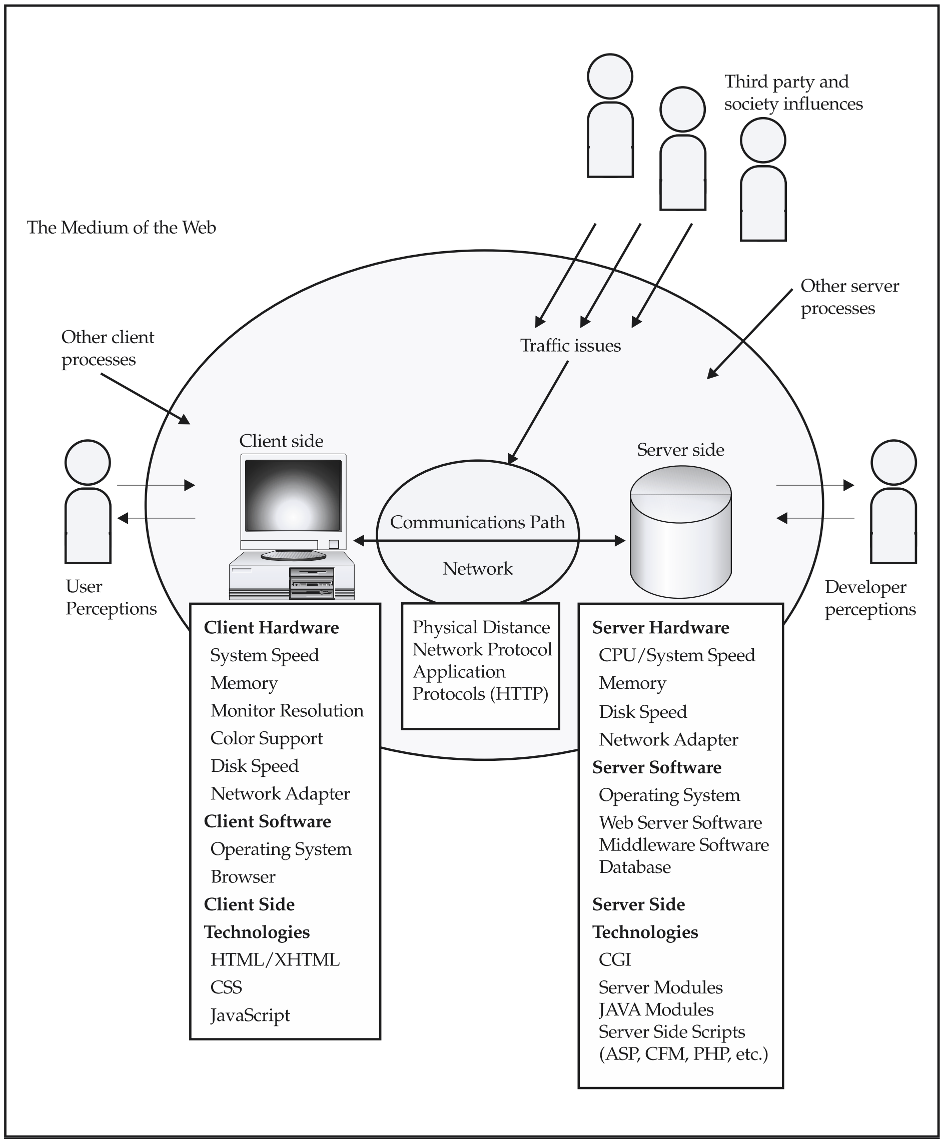
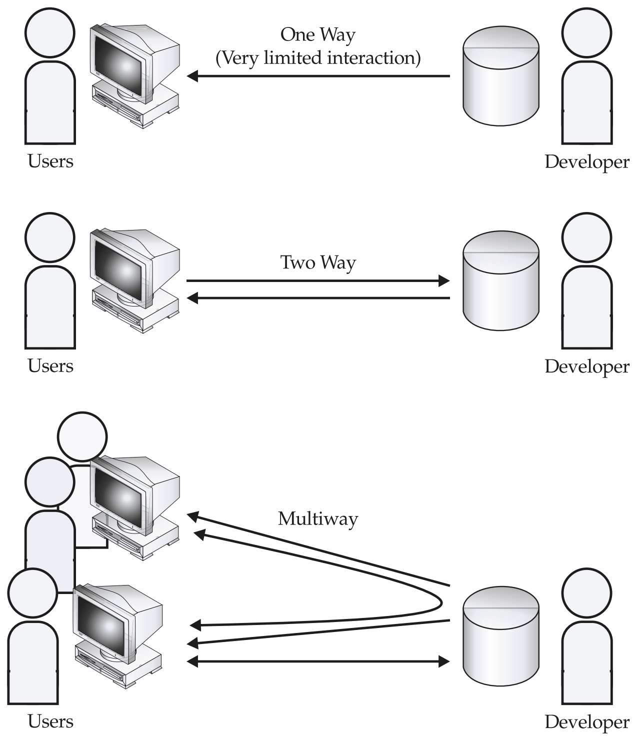


Figure 1-2. *Components of the Web Medium*

information for site owners or even other users, creating more of a multi-way communication path, as illustrated here:



During any communication, most users are generally unaware of the medium when things are working correctly. While users are affected by the medium, they often do not distinguish the individual components such as network, HTML, style sheets, and JavaScript—unless something goes wrong. In the negative case of a slow site, or one that causes visual or functional errors, the user may notice the medium but still may not distinguish which aspect of it is causing the problem. Users tend to see not the parts themselves, but the sum of them. This makes it important to think of sites as a whole, in order to understand how users see them.

Types of Web Sites

Users tend to view Web sites, and thus Web site design, by the function of the site or by its visual appearance. It is important to be able to describe sites this way; however, there are many more ways to categorize them. While the possible categories of sites may appear endless, we can safely group sites in a few general ways. We'll start first with the abstract and then move to visual categorizations.

Abstract Groupings

First, consider if a site is information focused or task focused. Sometimes we may describe this distinction as one between a site that is *document-centered* and one that is *application-centered*. Document-centered or informational sites provide information for users, but they provide very limited interactivity (other than allowing the user to browse, search, or sort the information presented). Sites that are task or applications oriented allow the user to interact with information or accomplish some task, such as transferring funds from a bank account or buying a new sweater. Hybrid sites do a little of both; these are becoming more common as the line between information and application blurs. Figure 1-3 plots the continuum from a simple static document-oriented site

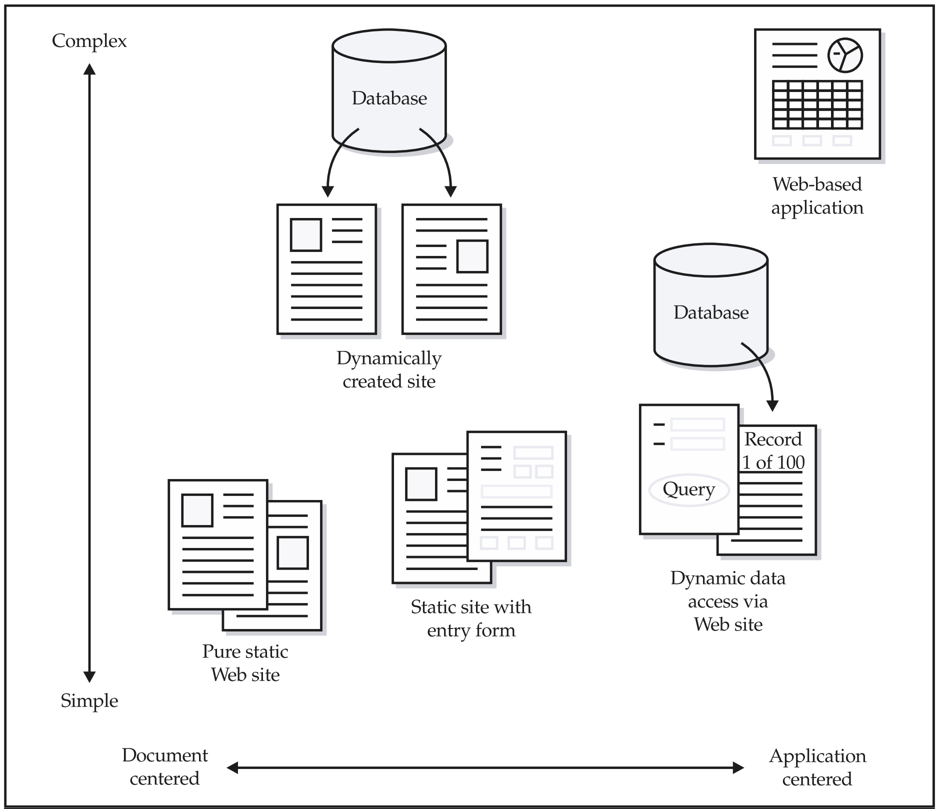


Figure 1-3. *The range of Web sites*

(often called a "brochureware" site) to full-blown software applications. This abstract grouping suggests that there is a transition from more document- or print-oriented Web sites to more interactive programmatic Web sites. This is indeed true; the intersection between the two philosophical camps is a source for much of the contention—and innovation—in the Web design community.

Another way we might group sites is within the following broad categories:

- **Informational sites** These sites provide information about a particular subject or organization (the "brochureware" sites). These are the most common Web sites on the Internet and often take on aspects of the other site categories over time.
- **Transactional sites** This type of site can be used to conduct some transaction or task. E-commerce sites fall into this category.
- **Community sites** These provide information or transaction-related facilities, but focus on the interaction between the visitors of the site. Community-based sites tend to focus on a particular topic or type of person and encourage interaction between likeminded individuals.
- **Entertainment sites** These sites are for game playing or some form of amusing interaction, which may include transactional, community, and informational elements.
- **Other sites** Included here are artistic or experimental sites, personal Web spaces such as Web logs (also called blogs), and sites that may not follow common Web conventions or have a well-defined economic purpose.

We might also group sites based upon the organization that is running, or in some sense paying for, the site. Within this type of categorization we see five major groupings:

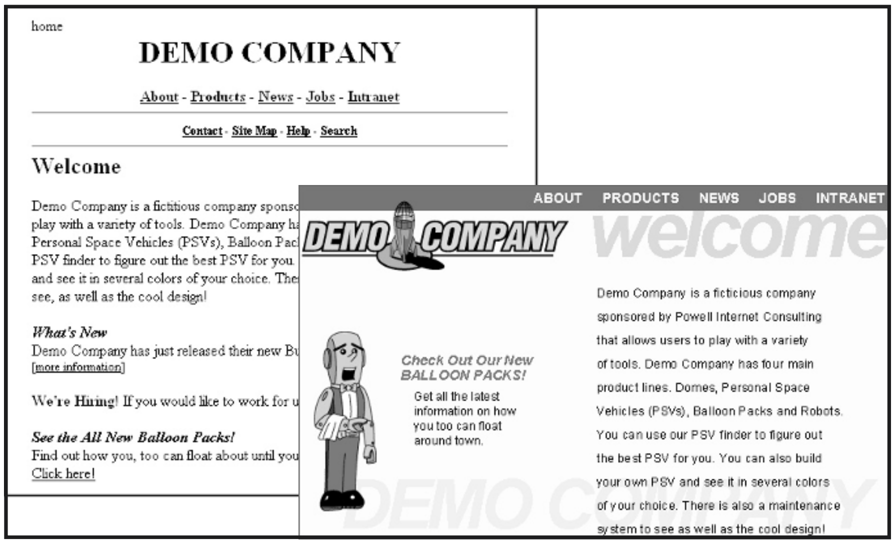
- **Commercial** A site in this group is built and run by an organization or individual for commercial gain, either directly through e-commerce or indirectly through promotion for some off-line purchase of goods or services.
- **Government** This site's parent entity is ultimately a government organization, and the purpose of the site is to satisfy some social or legal need.
- **Educational** This type of site's parent entity is some educational institution (perhaps government related), and it is used to support learning or research goals.
- **Charitable** A charitable site exists to promote the goals of a nonprofit organization or the charitable activities of an individual or organization.
- **Personal** The site exists at the sole discretion of some person or group for any number of reasons, usually as a creative outlet or form of personal expression.

Categorization can be difficult. For example, educational sites might really fall under the governmental category. Some sites in the personal category may arguably belong in the charitable or commercial group, depending on the reason for the person putting the site together. Now we turn to the more visual characteristics of sites, with a few sample categories of sites commonly seen on the Web.

Visual Groupings

As we group sites visually, we may see a range from those which rely more heavily on text and those which focus more on graphic presentation or imagery. The four most common design schools on the Web are:

- **Text oriented** These are sites designed with a focus on textual content. Such sites, as shown in Figure 1-4, are relatively lightweight, download-wise, and often somewhat minimalist in design.
- **GUI style** These are sites that follow certain graphical user interface (GUI) conventions from software design, such as top-oriented menu bars, icons, and pop-up windows. GUI-oriented sites range from simple GUI devices added to a primarily text-oriented site to full-blown Web applications with customized user interface widgets. Figure 1-5 shows some examples of GUI style Web sites.


home
DEMO COMPANY
About - Products - News - Jobs - Intranet
Contact - Site Map - Help - Search
ABOUT PRODUCTS NEWS JOBS INTRANET
DEMO COMPANY
welcome
DEMO COMPANY

Welcome

Demo Company is a fictitious company sponso
play with a variety of tools. Demo Company ha
Personal Space Vehicles (PSVs), Balloon Pac
PSV finder to figure out the best PSV for you.
and see it in several colors of your choice. The
see, as well as the cool design!

What's New
Demo Company has just released their new Bu
[more information]

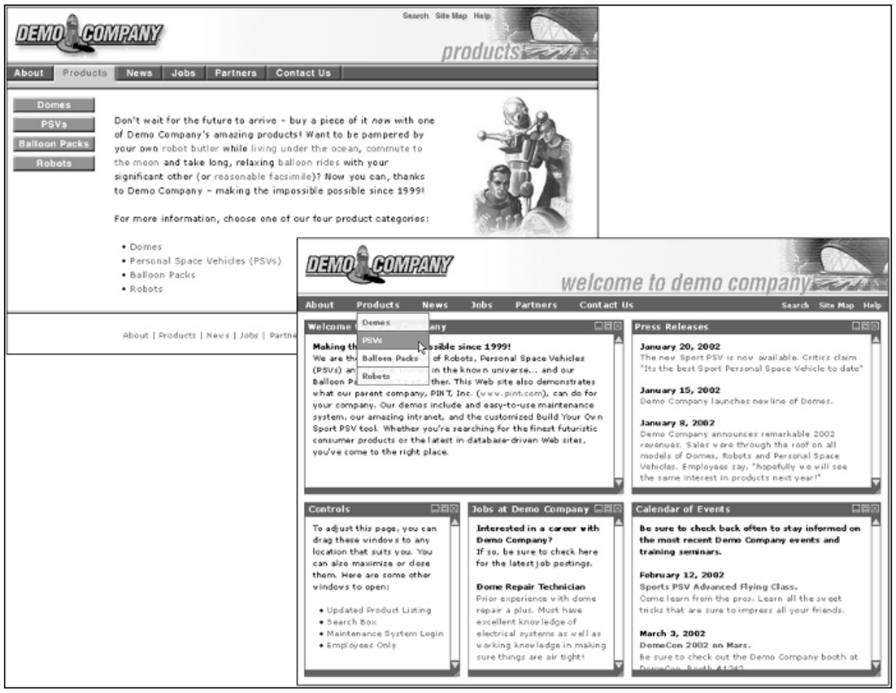
We're Hiring! If you would like to work for u

See the All New Balloon Packs!
Find out how you, too can float about until you
Click here!

Check Out Our New BALLOON PACKS!
Get all the latest information on how you too can float around town.

Demo Company is a ficticious company sponsored by Powell Internet Consulting that allows users to play with a variety of tools. Demo Company has four main product lines. Domes, Personal Space Vehicles (PSVs), Balloon Packs and Robots. You can use our PSV finder to figure out the best PSV for you. You can also build your own PSV and see it in several colors of your choice. There is also a maintenance system to see as well as the cool design!

Figure 1-4. *Text-oriented sites*



Don't wait for the future to arrive – buy a piece of it *now* with one of Demo Company's amazing products! Want to be pampered by your own robot butler while living under the ocean, commute to the moon and take long, relaxing balloon rides with your significant other (or reasonable facsimile)? Now you can, thanks to Demo Company – making the impossible possible since 1999!

For more information, choose one of our four product categories:

- Domes
- Personal Space Vehicles (PSVs)
- Balloon Packs
- Robots

Making th ssible since 1999!
We are th of Robots, Personal Space Vehicles (PSVs) an in the known universe... and our Balloon Pa ther. This Web site also demonstrates what our parent company, PINT, Inc. (www.pint.com), can do for your company. Our demos include and easy-to-use maintenance system, our amazing intranet, and the customized Build Your Own Sport PSV tool. Whether you're searching for the finest futuristic consumer products or the latest in database-driven Web sites, you've come to the right place.

January 20, 2002
The new Sport PSV is now available. Critics claim "Its the best Sport Personal Space Vehicle to date"

January 15, 2002
Demo Company launches new line of Domes.

January 8, 2002
Demo Company announces remarkable 2002 revenues. Sales were through the roof on all models of Domes, Robots and Personal Space Vehicles. Employees say, "hopefully we will see the same interest in products next year!"

To adjust this page, you can drag these windows to any location that suits you. You can also maximize or close them. Here are some other windows to open:

- Updated Product Listing
- Search Box
- Maintenance System Login
- Employees Only

Interested in a career with Demo Company?
If so, be sure to check here for the latest job postings.

Dome Repair Technician
Prior experience with dome repair a plus. Must have excellent knowledge of electrical systems as well as working knowledge in making sure things are air tight!

Be sure to check back often to stay informed on the most recent Demo Company events and training seminars.

February 12, 2002
Sports PSV Advanced Flying Class.
Come learn from the pros. Learn all the sweet tricks that are sure to impress all your friends.

March 3, 2002
DomeCon 2002 on Mars.
Be sure to check out the Demo Company booth at

Figure 1-5. *Web Designs with a little GUI or a lot of GUI*

- **Metaphorical** Metaphor sites borrow ideas from "real life." For example, a site about cars might employ a dashboard and steering wheel in design and navigation. A metaphor-designed site, as shown in Figure 1-6, tends to be extremely visual or interactive. This may be frustrating to some users and engaging to others.
- **Experimental** Experimental designs attempt to do things a little differently than the norm. Creativity, unpredictability, innovation and even randomness are often employed in sites following the experimental design style, as shown in Figure 1-7.

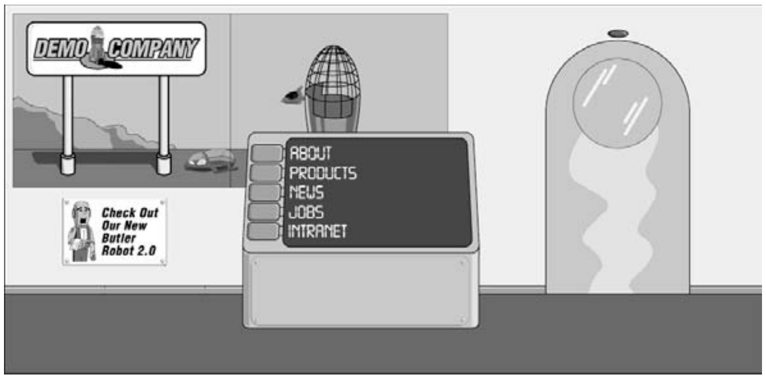
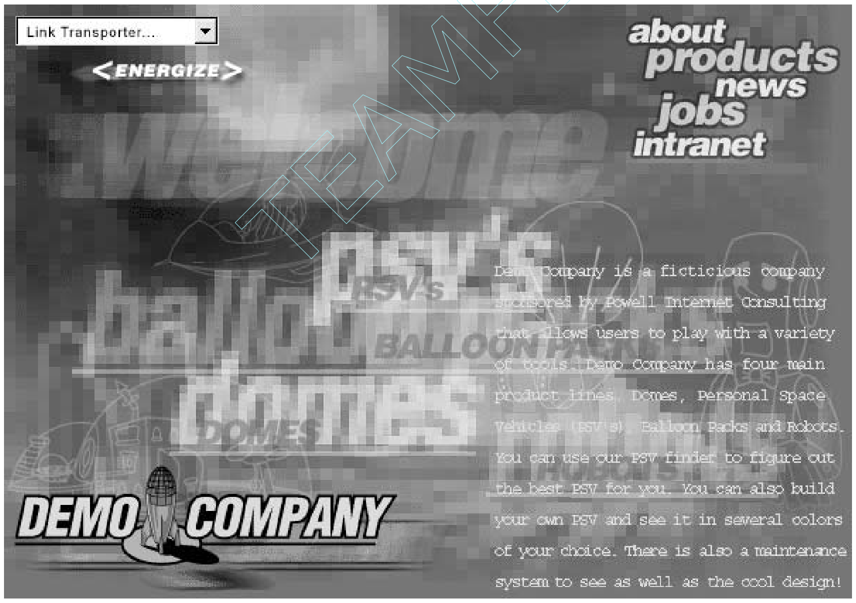
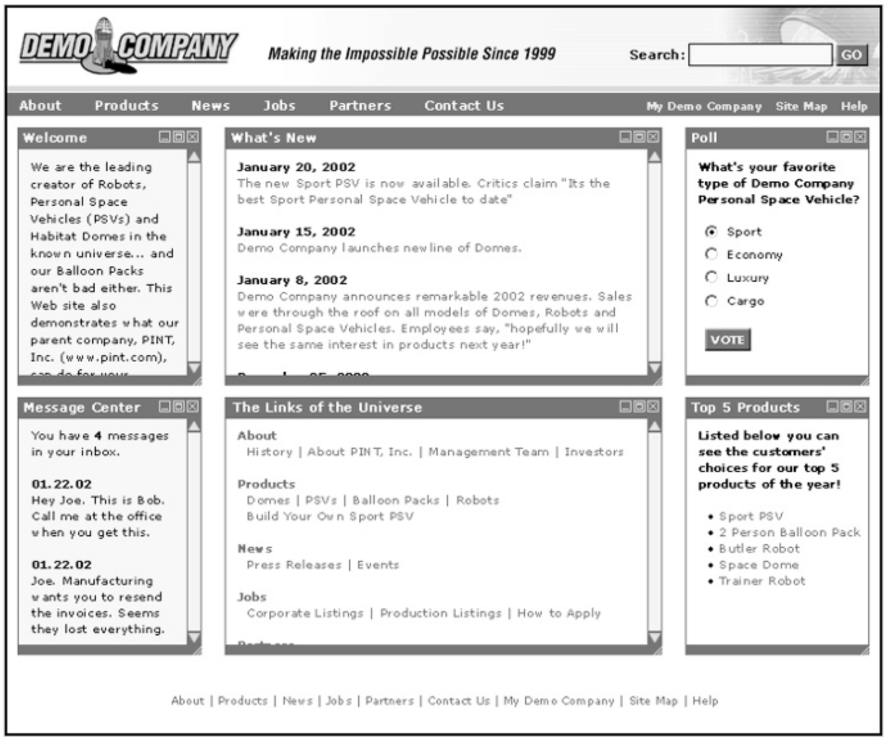


Figure 1-6. *Metaphorical design*



Demo Company is a ficticious company sponsored by Powell Internet Consulting that allows users to play with a variety of tools. Demo Company has four main product lines, Domes, Personal Space Vehicles (PSV's), Balloon Packs and Robots. You can use our PSV finder to figure out the best PSV for you. You can also build your own PSV and see it in several colors of your choice. There is also a maintenance system to see as well as the cool design!

Figure 1-7. *Experimental design*



Welcome

We are the leading creator of Robots, Personal Space Vehicles (PSVs) and Habitat Domes in the known universe... and our Balloon Packs aren't bad either. This Web site also demonstrates what our parent company, PINT, Inc. (www.pint.com),

What's New

January 20, 2002
The new Sport PSV is now available. Critics claim "Its the best Sport Personal Space Vehicle to date"

January 15, 2002
Demo Company launches new line of Domes.

January 8, 2002
Demo Company announces remarkable 2002 revenues. Sales were through the roof on all models of Domes, Robots and Personal Space Vehicles. Employees say, "hopefully we will see the same interest in products next year!"

Poll

What's your favorite type of Demo Company Personal Space Vehicle?

[✓ Sport]
Economy
Luxury
Cargo

VOTE

Message Center

You have **4** messages in your inbox.

01.22.02
Hey Joe. This is Bob. Call me at the office when you get this.

01.22.02
Joe. Manufacturing wants you to resend the invoices. Seems they lost everything.

The Links of the Universe

About
History | About PINT, Inc. | Management Team | Investors

Products
Domes | PSVs | Balloon Packs | Robots
Build Your Own Sport PSV

News
Press Releases | Events

Jobs
Corporate Listings | Production Listings | How to Apply

Top 5 Products

Listed below you can see the customers' choices for our top 5 products of the year!

- Sport PSV
- 2 Person Balloon Pack
- Butler Robot
- Space Dome
- Trainer Robot

About | Products | News | Jobs | Partners | Contact Us | My Demo Company | Site Map | Help

Figure 1-8. *Portal style design*

Of course, on the Web we find mixtures of form or potential new categorization of sites. For example, how would you categorize a *portal site*, such as the one shown in Figure 1-8, that provides a wealth of content, navigation choices, and even community related-facilities in a single page? This is certainly a design style that is used in a great deal of sites. We see the potential rise of other design categories when we look at Web site genres such as e-commerce sites, particularly strict "catalog and cart" sites, as well as online personal journal sites called "Weblogs" or "blogs." We'll take a closer look at these design ideas in later chapters.

A Clearer Definition of Web Design

So, after all this discussion, what exactly is Web design? It is obviously a very user-centered multidisciplinary design pursuit that includes influences from visual arts, technology, content, and business. A succinct definition follows.

Web Design: A multidisciplinary pursuit pertaining to the planning and production of Web sites, including, but not limited to, technical development, information structure, visual design, and networked delivery.

Because Web design is so multidisciplinary, it is often appropriate to pull ideas and theories from related fields. Indeed, we've been doing that even in the very first pages of this book. Some people, however, take this approach a little too far, developing their sites in a manner similar to print pieces or adopting so many software GUI interface conventions that the user becomes confused. While Web design borrows heavily from other design pursuits, there are significant differences. For example, the medium is very different than print because more function is provided—not unlike software. Delivery issues and content effects make Web sites different from traditional software applications as well. Web design isn't just adoption of old ideas. It's something altogether new.

We shouldn't say the Web is totally different either. There are plenty of people who do that as well. The Web is so revolutionary, they say, that none of the old rules hold. This is complete nonsense. Despite the proclamations of pundits, new media forms have always adopted conventions from other forms and invented new ones of their own. Furthermore, no new form has completely eliminated any other. Radio, magazines, newspapers, television, and other entertainment media all continue to exist in some form or other despite emerging technologies and new media forms. The Web certainly isn't so new that we should throw out any valuable concepts we learned before. It does, however, have its own principles. We should strive to understand other media design concepts and modify them to fit the Web. The rest of the introduction will present some of the themes of Web design and conclude with a "roadmap" for the rest of the book.

Web Design Themes

When discussing Web design, we see similar themes come up over and over again. Whether it's the political struggle between a corporation's marketing department and information technology group over site ownership, or a graphic designer trying to convince a client of the appropriateness of a particular look or multimedia technology, these themes are at the heart of the matter. These issues often result in rather heated discussions among designers, as well as between designers and their clients both inside and outside corporate Web teams. While there is no simple answer to some of these issues, they are relatively easy to describe.

Generally the major themes behind modern Web design include:

- Designer needs versus user needs
- The balance of form and function
- The quality of execution
- The interplay between convention and innovation

In the abstract sense, these themes are not at all unique to the Web medium. Artists like Leonardo DaVinci certainly struggled at times to balance the desires of patrons and even his viewing public with his own needs. Commercial artists producing something like a magazine advertisement or billboard have to balance the demands of visual look with successful and clear communication. Execution varies in any discipline, but in one as young as Web design, the effects are more evident. Lastly, the rules of convention and the desires of innovation are as common as the struggle of a young person rebelling against convention, the middle age designer discovering the wisdom of the masters, and the old designer trying to rediscover his or her innovative youth. Despite the general nature of these themes, their specific details vary with each medium. It will be valuable to introduce each here before we encounter them later on. We start with the most important issue first: user-centered versus designer-centered site design.

User-Focused Design

A common theme of Web design is the focus on users. Unfortunately, a common mistake made in Web development is that, far too often, sites are built more for designers and their needs than for the site's actual users. Always remember this important tenet of Web design:

Rule: YOU are NOT the USER.

What you understand is not what a user will understand. As a designer, you have intimate knowledge of a Web site. You understand where information is. You understand how to install plug-ins. You have the optimal screen resolution, browser setup, and so on. When you build your site around your own visual characteristics and skill levels, you often will confuse the actual users of the site. You must accept the fact that many users will not necessarily have intimate knowledge of the site you have so carefully crafted. They may not even have the same interests as you.

Given the importance of the users' interests and desires, it might seem appropriate to simply ask the users to design the site the way they want. This seems to be a good idea until you consider another basic Web design tenet:

Rule: USERS are NOT DESIGNERS.

Not everyone is or should be a Web designer. Just as it would seem foolish to let moviegoers attempt to direct a major motion picture on the basis of their having viewed numerous movies, we should not expect users to be able to design Web sites just because they have browsed a multitude of sites. Users often have unrealistic requirements and expectations for sites. Users will not think carefully about the individual components of a Web site. In summary, users are not going to have the sophisticated understanding of the Web that a designer will have.

That said, the key to successful, usable Web site design is always trying to think from the point of view of the user. *User-centered design* is the term given to design that always puts the user first. But what can we say about users? Is there a typical user? Does a "Joe Average Internet" exist that we should design our sites for? Probably not, but we certainly should consider certain traits, such as reaction times, memory, and other cognitive or physical abilities, as we design sites. An overview of cognitive science helps us understand basic user capabilities; we will discuss this topic further in the next chapter. Remember, however, that while users may have similar basic characteristics, they are also individuals. What may seem easy to one user will be hard for another. Sites that are built for a "common" user may not meet the needs of all users. Power users may find a site restrictive, while novice users find it too difficult. Users are individuals with certain shared capacities and characteristics. Sites should take account of the relevant differences while focusing on the commonalities, as stated by the following Web design tenet:

Rule: Design for the common user, but account for differences.

Lastly, we can see that the differing needs of the user and the designer raise an issue of control. Control over a visit to a site is an unwritten contract between the designer and the visitor to how the experience will unfold. Often, sites provide little user control, forcing the user to view content in a predetermined order with little control over presentation or technology. Rarely do we find the exact opposite occurs, where the site gives users ultimate control over visitation, allowing them to choose what to see and how to see it and even allowing them to add to or modify the site's contents. However, most sites do allow the user some choices and the ability to control experience, but always under the influence of the designer's requirements. We'll revisit some of the general ideas of control and user experience throughout the book.

Form and Function

A key problem with Web design is that sites often do not balance form and function. Under the influence of modernism, many designers have long held that the form of something should follow its function. Consider that the form is one base of our Web design pyramid analogy, while function is the other. Function without form would be boring: while the site may work, it won't inspire the user. Conversely, even if the form is impressive, if the function is limited, the user will be disappointed. There needs to be a clear and continuous relationship between form and function. Put simply, the form of a site should directly relate to its purpose. If the site is marketing-driven, it might be very visual and even incorporate heavy amounts of multimedia if it helps to accomplish our goals. However, if the site is clearly a task-based one, such as an online banking site, it might have a much more utilitarian form. Of course, determining the appropriate form for a site requires that the function of the site be clearly defined. Unfortunately, for many Web sites the ultimate function of the site isn't always clearly conveyed. Even worse, the relationship of form and function for the site is not always clearly established.

Rule: Make sure the visual form of a site relates to its function.

It is likely that there will be a continual struggle between form and function, despite the fact that in nearly all cases the only side the designers should be on is that of their users. In fact, there really need be no disagreement. Form and function do not always have to fight; they complement each other nearly all of the time. A nice-looking design makes a functional site much better, while great functionality will make up for a deficiency in "look and feel" over time.

Seasoned designers understand this balance and practice the idea of holistic design by following the rule that the correct execution and integration of all facets of the site will outweigh the value of a single component. In fact, the real difference between a Web designer and a mere Web builder is that the former is capable of not only executing the individual parts of a site correctly but can also breathe extra "life" into the project as a whole.

Execution: The Easy Part?

HTML, XML, CSS, JavaScript, Java, Flash, browser compatibility, server capacity, and all the other components of Web development are the easy part of Web design. While learning a new technology might take some time and effort, it is generally quite easy to say whether some HTML or other technology is used correctly or not. However, today's sites are riddled with execution problems, ranging from simple typos to significant technical compatibility, delivery, and usability problems.

A Web site should only be considered excellent if it is useful, usable, correct, and pleasing. The meaning of each of these considerations is somewhat subjective, except in the case of correctness. For a site to be well designed, its execution must be excellent. This means that the site must not *break* in any way. The HTML must be correct and the images saved properly so that the page renders itself as the designer intended. Any interactive elements, whether in the form of client-side scripts in JavaScript or server-executed CGI programs, must function properly and not result in error messages. The navigation of the site must work at all times. Broken links accompanied by the all too familiar "404: Not Found" message are not the sign of a well-executed site. Errors, in fact, should be handled, and the site should fail gracefully, if at all. While execution seems like an obvious requirement for excellence, too many sites exhibit execution problems to let this consideration go unmentioned:

Rule: A site's execution must be close to flawless.

Why are execution problems rampant in Web sites? Simple: this is a young industry with changing standards. Consider state-of-the-art Web design from a few years ago and you'll see the difference. Further, most Web professionals often didn't have the background in computer science, networking, hypertext theory, cognitive science, and all the other disciplines that might affect the quality of the produced site. Some naïve designers even ignore the inherent differences in the emerging Web medium by not addressing problems of varying resolutions, color reproduction, bandwidth limitations, and so on. A Web designer who overlooks these types of technical characteristics of the Web is like the print designer who will not admit that ink bleeds on paper—great Web designers must know and respect the medium, which includes everything from browsers and bandwidth to programming and protocols.

Rule: Know and respect the Web and Internet medium constraints.

So, given the environment of Web design, we end up with today's assortment of sites, from those that are standards-compliant, lightweight, user-friendly, informational, and task-rich to those that are browser-specific, unusable, or multimedia bandwidth hogs touted as "next generation" designs. Yet does this comparison suggest that all good sites are the same? Not necessarily.

Conformity versus Innovation

Many Web designers feel that design theories and site design categorization increase conformity and stifle innovation. It is true that rigidly following design templates such as "top-left-bottom" layout or adhering to such common practices as putting organizational logos in the left corner of a Web page will limit some page design choices; designers have misunderstood the reason for these conventions. Consider that, while it might be possible to design books with triangular pages, few books are done this way. The cost of production, the awkwardness, and the reader's unfamiliarity with such a shape could make a triangular book a risky proposition. Most books are square or rectangular and have a distinct cover, title page, table of contents, chapter breaks, and so on. Are these conventions stifling to the book designer? Few would say they are; a great deal of creativity is still possible within the given constraints of a modern book. The same should be said for Web design. Graphical User Interface (GUI) design for software programs has influenced what is considered standard for Web user interfaces, but new ideas have also emerged. Designers need to respect conventions of navigation choices, navigation placement, colors, and so on. These ideas do not limit design; they simply constrain sites to recognizable forms so that users do not find the sites they visit to be completely different.

Rule: Appropriately respect GUI and Web interface conventions.

All these general "designing theories" set the stage for learning Web design, but when you apply them to a real site the theories will become much more specific. In short, we have a lot of ground to cover, so let's get started.

Learning Web Design

Reading a book like this is useful in uncovering the theories and commonly held practices of Web design, but more is required if you are to ever achieve mastery of Web design. Always remember that learning the basics of Web site development is not necessarily difficult, but do not underestimate the time and effort it will take to become an accomplished designer. This is no different from carpentry, painting, writing, illustration, or just about any skill you can think of. So make sure you set reasonable expectations for yourself as you learn.

One useful approach to learning Web design is by evaluating the efforts of others. We can look at what is done right and what is done wrong and try to emulate the good and fix the bad. Beware, however: it is not always easy to evaluate and compare site designs. Far too often people compare that which is not comparable. You would never compare a video game with a word processor, yet both are software programs. Why, then, do we compare experimental sites with corporate sites, or e-commerce sites with Web design agency portfolio sites? Far too often, this type of comparison is done in the Web design community. Sites and books put forward a variety of sites as absolute yardsticks of great design. Yet, obviously, not all sites will have the same issues as those that the "excellent design" rules were derived from. What is cool or clever for one site may be an absolute disaster for another. A great example is the splash page shown in Figure 1-9. A *splash page* is the term used to describe an entry page to a site—one that comes before the actual home or core page of the site.

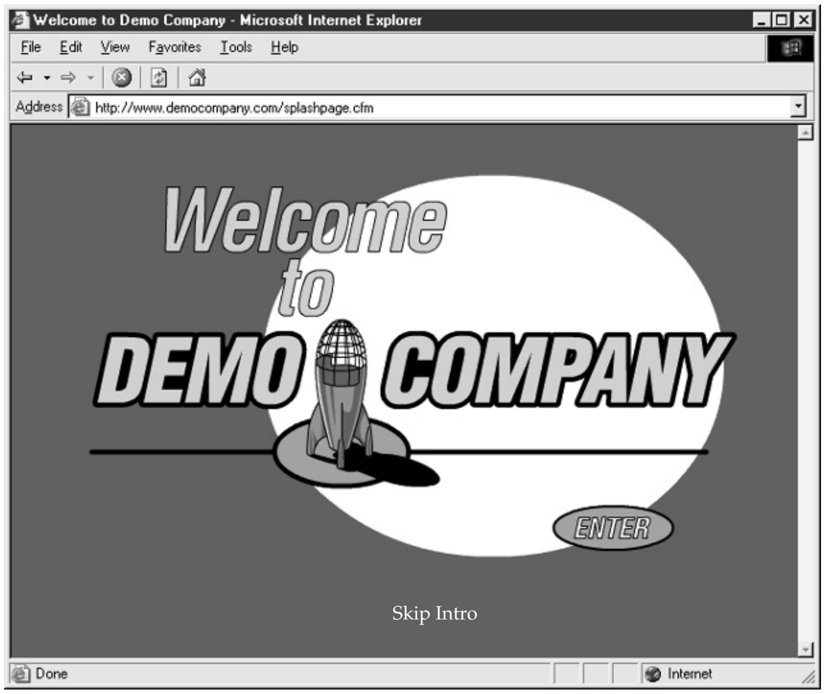

Welcome to Demo Company - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.democompany.com/splashpage.cfm
Welcome to DEMO COMPANY
ENTER
Skip Intro
Done
Internet

Figure 1-9. *A splash page*

A splash page is often used to set the tone for the site and may consist of an interesting animation, preloading sequence, or some form of "installation" information in regards to what technology is required or what the user's expectations should be. While splash pages can be effective, very often they are not. The mere mention of the phrase "skip intro" results in hearty chuckles among many designers. Yet the much maligned splash page may just happen to have some uses. Some movie and entertainment sites have found such sequences to be an integral aspect of their sites. Just like a movie without opening credits, these sites would be incomplete without a splash page. This simple example illustrates the most dangerous problem facing those learning Web design—namely, assuming there is only one form of good Web design. Often, it seems that the only absolute in a fluid discipline like Web design is that there is no absolute.

Rule: There is no form of "correct" Web design that fits every site.

As you read this book, you'll notice that various rules and suggestions are presented. These are fairly safe and well thought out, but their real value comes from understanding the motivation for them, not from blindly applying them. The importance of this distinction will become apparent once you see that many of the "rules" seem at odds with other rules. Exhibiting good judgment that strikes a balance between conflicts is a key attribute of a great Web designer.

A discussion of site evaluations that attempts to cover all aspects of Web design from taste to technical implementation can be found in Chapter 5, and a checklist useful during such site evaluations is presented in Appendix B. Yet do not fall into the trap of becoming a professional critic. Certainly it is important to point out what not to do by finding flaws in sites or criticizing what is bad, but spending too much time discussing bad Web design may not be fruitful, particularly when you consider that there is no accounting for poor taste. It is easy to criticize, but it is much more difficult to take your acquired knowledge and apply it to a site of your own.

In the final analysis, the best approach to learning Web design is obviously by doing. Reading about site design theory or reviewing sites simply isn't a replacement for building sites of your own. Yet before you set out constructing a site, learn the core principles of Web design as well as the building and evaluation procedures that will help you construct your Web sites well.

Summary

Pinning down exactly what is meant by the term Web design can be difficult. At best we can see that Web design is a multidisciplinary pursuit that consists of five primary components: content, visuals, technology, delivery, and purpose. However, theories of exactly how these components should mix together vary from person to person as well as project to project. Striking a fine balance between form and function, user and designer, content and task, and convention and innovation is the lofty goal of the Web designer. The good designer knows that scales should not tip too far one way or another and tries to avoid the absolutisms of “correct” Web design. Yet not everything in the field of Web design is so abstract—many specifics can be found. Correct mastery of the technical medium and knowledge of various details and conventions are mandatory for aspiring Web practitioners. We begin the discussion of the core aspects of Web design in the next chapter, which focuses on user-centered design.

Chapter 2

User-Centered Design

As discussed in Chapter 1, Web sites are often developed from one particular philosophical reference point. Sometimes this point of reference is content centered; other times, it is technology-centered. Even more frequently, it is graphics-centered. However, the real emphasis when building sites should always be the user. Keeping users in mind and always trying to meet their needs should be the key focus of user-centered design.

Understanding users needs isn't easy. While users may share common capabilities such as memory or reaction time, each user is still a distinct individual. Sites should be built for common user capabilities, rather than for the extreme novice or power user. Sites should be accessible to all and be able to account for the differences exhibited by individuals. Building a usable Web site is challenging, since what is usable to one person may be problematic for another. The likelihood of building a user-centered site is greatly improved through user interviews, testing, or even iterative design. Always be wary, though, of falling into the "user trap." While a site should always be built for users, the desires of the site's creators must also be met, even though these may be somewhat at odds with the desires of the site's users. The fine balance of power between user and designer is not always easily achieved.

Usability

Everyone has a vague idea of what it means for something to be *usable*. People will talk at length about how Web sites are supposedly user friendly, intuitive to use, or simply "usable." What, exactly, does it mean for something to be usable? First, consider the concept of *utility* in connection with two e-commerce sites that sell books and offer the same basic features. Both allow the user to search or browse for books, read information on books, purchase books, and track their orders. If both sites have basically the same features, they have the same utility—meaning they can do the same thing. Given that the sites have a few basic functions, you may find it easier to perform the same task on one site than the other. In this case, we can say that one site is more usable (has greater utility) than the other. Unfortunately, it is difficult to agree on what is usable. Plenty of people have attempted to characterize what usability is. Consider the following definition adopted from an ISO standard definition of *usability*:

Definition: Usability is the extent to which a site can be used by a specified group of users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

Consider each piece of the definition. First, note that we should limit the group of users when talking about usability. Recall that usability will vary greatly depending on the user.

Next, usability should be related to a task. You should not consider a site to be usable in some general sense. Instead, discuss usability within the context of performing some task, such as finding a telephone number for contact, purchasing a product, and so on. Usability is then judged by the effectiveness, efficiency, and satisfaction the user experiences trying to achieve these goals.

Effectiveness describes whether or not users are able to actually achieve their goals. If users are unable to, or only partially able to, complete a task they set out to perform at a site, the site really isn't usable. Next, usability is related to efficiency. If users make a great number of mistakes or have to do things in a roundabout way when they visit a site, the site isn't terribly usable. Last, the user must be satisfied with the performance of the task.

Many other definitions of usability exist. Some usability professionals suggest that usability can be concretely defined. Maybe it could be computed as some combination of the completion time for a typical visit and the number of errors made during the visit. From the user's point of view, that might not mean much. Users might just be concerned with how satisfied they were after performing a task. Many usability experts, such as Jakob Nielsen (http://www.useit.com), tend to have similar definitions more in line with the ISO one. For example, Nielsen suggests that the following five ideas determine the usability of a site:

- Learnability
- Rememberability
- Efficiency of use
- Reliability in use
- User satisfaction

By this definition, a site is usable if it is easy to learn, easy to remember how to use, efficient to use (doesn't require a lot of work on the part of the user), reliable in that it works correctly and helps users perform tasks correctly, and results in the user being generally satisfied using the site. This still seems fuzzy in some ways, and conflicts arise easily in the usability area. For example, a site that is easily learnable by a novice user may be laborious to use for a power user. Because people are different and come with different levels of capabilities and Web knowledge, not everyone is going to agree on what is supposedly usable. A site that is easy to one user may be hard for another.

Rule: There is no absolute description of what constitutes a usable site.

Even without considering user differences, we may find that usability varies according to how a single user interacts with a site. Usability also often depends on the medium of consumption—textual content viewed on the screen may be more usable in a large size, but when it is consumed on paper, it might be better smaller. If you have tried to read large amounts of small-size content online, you know it can be difficult. People tend to find that it is much easier to read it on paper. Some experts have suggested that people read much slower onscreen and tend to scan more than read content online. In this case, the medium of consumption—screen or print—has affected the usability of the content. In the case of the Web, the medium, which includes networks, browsers, screen sizes, and technologies like HTML, often contributes in a large way to usability problems. Throughout this book, the mantra of "know thy medium" should be repeated over and over.

Rule: Usability depends on the medium of consumption.

What is considered usable often varies between sites. An entertainment site would have different usability constraints than a commercial one. Further, the user's familiarity with a site—as well as how often the user accesses the site and for what purpose—will affect the site's perceived user friendliness. Consider how people may feel about the usability of a site that they have never been to before and are only marginally interested in, as opposed to one that they frequently visit or must use. They may be much more forgiving of errors in the site they need to use or have come to use than in the one they are just casually interested in. In short, a "throwaway" single-time-visit site has different usability constraints than a site a user relies on day to day.

Rule: Usability depends on the type of site as well as the user's familiarity with it.

This idea might seem a tad unusual, but it shouldn't. People often come to believe inefficient ways of doing things are perfectly acceptable. Be careful about getting too scientific when talking about usability (measuring page clicks, mouse travel, errors rates, and the like). How users "feel" about the experience when they come away—their satisfaction with the site or the task performed—is really the most important thing. For some people, how they feel may not always be logical or even totally related to what happened during the site visit. Consider how many people gain satisfaction from performing difficult tasks; they may feel that way about some sites as well. Also, people let organizations that they are familiar with outside the Web get away with things at their sites that a new company can't, simply because they trust the name brand of the older firm. On the other hand, don't assume that the occasionally illogical user can be used as an excuse to produce a site that is hard to use. A site that requires the user to learn a new way of doing things, is inflexible, results in errors, or just doesn't work will generally result in poor user satisfaction. Improve usability and users will be happier.

Rule: Usability and user satisfaction are directly related.

To understand how to make something usable, you must understand users. The next few sections will discuss usability in light of user capabilities and tendencies. The conclusion of the chapter will revisit these subjects and present a few rules of thumb that can be applied during Web site design to improve a site's usability.

Who Are Web Users?

Site designers often make the common mistake of oversimplifying or completely ignoring the capabilities and desires of users. In some cases, concerns about designing the site with a particular browser or bandwidth in mind replace any serious thought of the user. Don't design your site for Netscape—design for people who happen to use the Netscape browser. Always remember the following very important Web design rule:

Rule: Browsers don't use sites, people do.

Fortunately, most designers don't go the extreme of completely forgetting the user, but often they do oversimplify who the site's users are. Far too often, sites are built for some elusive stereotypical Web user—the modem user accessing via AOL, perhaps. This user is just a nameless person surfing the Internet to be enticed into visiting the site and performing whatever task the designer desires. The reality is that users are not automatons with the same capabilities and desires, but individuals with a wide range of physical capabilities, needs, wants, expectations, and goals. Real Web users have bad days or can't always figure things out sometimes, just like the rest of us.

Suggestion: There are no generic people. Always try to envision a real person visiting your site.

While it may not be possible to create a perfect stereotypical user to design Web sites for, there are some general things that can be said about users. The first thing is to think about how today's typical user interacts with a Web site. Until alternative browsing environments such as cell phones or PDAs become much more commonplace, a user of your site is almost certainly sitting at a desk or table with a computer. Users sit at most a few feet from a monitor and generally use a keyboard and a mouse to interact with a Web site shown on the monitor. Primarily, they are using their eyes to access the information on the screen, though sound may also come into play. The stimulus from the site is filtered, and choice items may be consumed or, more accurately, committed to short- or long-term memory. The information they consume then may cause them to react by, for instance, clicking a link or entering data into a form. This simplified view of a user interacting with a site is shown in Figure 2-1.

Note *Adding in the constraint of mobility, the user's environment can really change usability. If you consider a user who is walking while browsing on a PDA or cell phone, you can see how the environment can affect usability.*

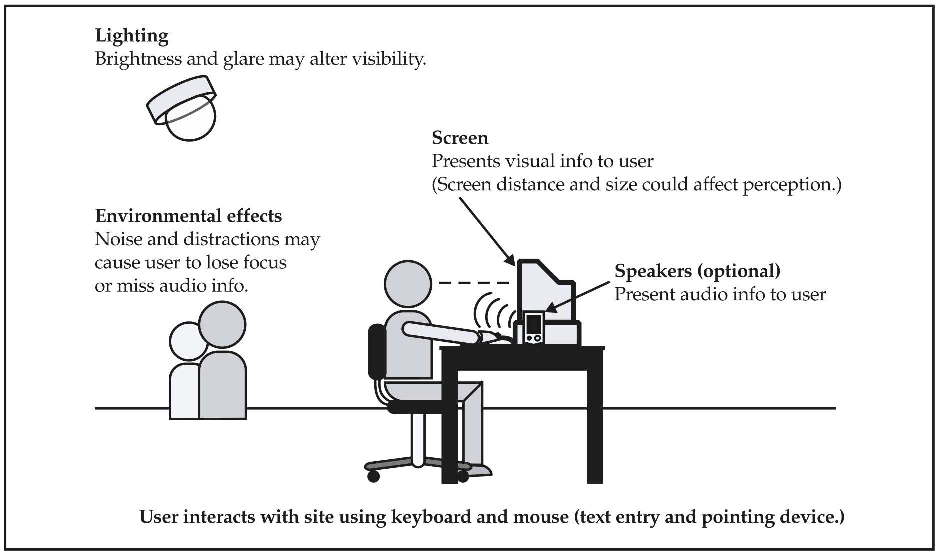

Lighting
Brightness and glare may alter visibility.
Screen
Presents visual info to user
(Screen distance and size could affect perception.)
Environmental effects
Noise and distractions may
cause user to lose focus
or miss audio info.
Speakers (optional)
Present audio info to user
User interacts with site using keyboard and mouse (text entry and pointing device.)

Figure 2-1. *Typical environment of user interacting with a site*

You can describe how people tend to react to the world around them, including Web sites, in the following way. First, they encounter some sensation that is stored in memory. Then they try to understand the sensation, which is filtered both consciously and unconsciously. Information from past experiences may be called into action, influencing how they perceive things and possibly helping them decide what to do. From this perception, they may perform an action—or possibly take no action—that will later result in more sensations to be interpreted. This simplified action/reaction/action loop is shown here:

$$\begin{tikzcd}
\text{Sensation} \arrow[rr] \arrow[dr] & & \text{Perception} \arrow[r] \arrow[dl, bend right] & \text{Action} \arrow[lll, bend right] \\
& \text{Memory} \arrow[ur, bend right] & &
\end{tikzcd}$$

Do not think that people can be simplified to a formula where a stimulus is provided that results in an action. People are more complicated than that. People are capable of learning things, and information they encounter is committed to memory that can be used to modify what they do. Further, people aren't perfect. Problems may occur, such as not remembering things properly. Different people perceive stimuli differently. Not everyone sees color quite the same way, for instance. Despite its simplification, the model does force designers to consider how people interact with the world—which includes their Web site. Common user characteristics such as sensation and memory need to be well considered, at least in a general sense, when building sites.

Common User Characteristics

There are no generic people, but people tend to have similar physical characteristics. Most people tend see about the same, are capable of remembering things, and react to stimuli in about the same way. However, remember that people are individuals. There will be some users who will be able to see much better than others. There will be people who can memorize hundreds of links and be able to quickly filter them, and others who will be overwhelmed when presented with more than two choices. There will also be a few users who react much faster or much slower to information than the average user. However, as with all aspects of Web design, we should aim first for the common user and make sure to account for differences. Let's first consider common user characteristics such as vision, memory, and stimulus reaction.

Vision

The first aspect to consider about users is how they receive information from a Web site. The primary way most users consume data from Web sites is visually. They look at a screen and consume information in the form of text, color, graphics, or animation. The user's ability to see is obviously very important. Consider, for example, users with poor eyesight. Unless the text is very large and the contrast between foreground and background elements very distinct, they may not be able to effectively interact with the content of the site. Unfortunately, many sites seem to assume that users have nearly superhuman vision, as text is sized very small, or a minor degree of contrast is used between foreground and background elements. A simple example of some of contrast and sizing problems can be found at http://www.webdesignref.com/visionissues.htm.

In order to avoid troublesome color combinations, designers should be aware of how color is perceived by the human eye. Three factors affect how color is perceived:

- **Hue** the degree to which a color is similar to the basic colors—red, green, and blue—or some combination of these colors.
- **Saturation** the degree to which a color differs from achromatic (white, gray, or black).
- **Lightness** the degree to which a color appears lighter or darker than another under the same viewing conditions.

Users with vision that is somewhat color deficient are often unable to differentiate between colors of similar hue when those colors are of the same lightness and saturation. For example, someone with the most common color deficiency—red-green color blindness—has trouble distinguishing between red and green when the red and green are close in saturation and lightness. Such color vision issues can be troublesome when you consider the difficulty in distinguishing between red and green traffic lights. Does the color-deficient user really know when to stop or go? In the real world, probably so, since the red light is always the top light— but on the Web, things aren't always so cut and dried. If links are similar in hue, lightness, and saturation, it might be difficult for someone to determine which links have been clicked and which have not.

Web page designers can avoid vision issues for users if they follow a few simple rules. First, make sure not to use text or graphic combinations that have a similar hue. Instead of using light blue on dark blue, use blue on yellow or white instead.

Suggestion: Avoid using text, graphics, and backgrounds of similar hue.

It is possible to get in trouble when using colored text on backgrounds with similar saturation. For example, instead of using a grayish blue text on a rose color background, where both colors are close to achromatic gray, use white text on a rose background, or vice versa.

Suggestion: Avoid combining text, graphics, and backgrounds of similar saturation.

The most obvious problem is when contrast is not great enough. Designers need to consider that dark text on a dark background or bright text on a bright background just may not be readable on all monitors or by people with color or vision deficiency. Instead of using a light blue text on a pale yellow background, use blue text on a white background. Or, black text on a white background is always a safe bet. Yellow and black contrast very well, and, therefore, they are used on road signs that are very important to read. However, before changing your Web site to this color combination, consider that design shouldn't be thrown completely out the window just because of usability concerns.

Rule: Keep contrast high. Avoid using text, graphics, and background of similar lightness.

A very important use of color in a Web page is link color. In general, you should really avoid modifying link colors in any way. However, if you do modify link colors, make sure to avoid using link state colors of similar hue, similar saturation, or similar lightness to the background or to one another. For example, avoid links that change from red to pink. For some reason, designers seem to favor such types of combinations. Instead, consider using links that change from dark blue to pink, similar to the normal link state. Be careful with the background color, as it may interfere with link readability. Because of this, white is a good background color. However, if a sacrifice has to be made with color contrast, make the visited state color the one with the contrast problem, since these are links the user would generally be less interested in.

Links, as well as normal text, often have problems with backgrounds. In particular, avoid patterned backgrounds with multiple hues, saturations, or levels of lightness.

Backgrounds like speckles or texture patterns tend to make poor backgrounds; instead, choose a subtle pattern or simple color.

Suggestion: Avoid using busy background tiles.

To make pages more readable and to deal with users who might have some color or vision deficiency, Web designers should make sure colors that are meant to distinguish items are significantly different in two areas (for example, hue and lightness). By following this rule, if the user is color deficient in one area (for example, red-green hue), he or she can still distinguish the item by another attribute, such as its lightness or saturation.

Rule: Make sure colors that are meant to distinguish items like links are significantly different in two ways, such as hue and lightness.

For more in-depth discussions of vision, color, and imagery on the Web, refer to Chapters 12, 13, and 14.

Memory

Memory is critical to a user being able to utilize a site. If users are unable to remember anything about a site as they browse it, they will become hopelessly lost, since they will not be able to recall if they have been someplace before. However, any user's memory is far from perfect, and users don't consciously spend time trying to memorize things. Users tend to always follow a simple rule: try to do minimal work for maximal gain. Simple human nature suggests that a user is not going to spend a great deal of time to figure something out unless there is a potentially good payoff.

Rule: Users try to maximize gain and minimize work.

Of course, what is considered a good payoff will vary from person to person. Some people like to solve complex puzzles just for personal satisfaction. For them, the payoff is an intense feeling of accomplishment from solving a problem. However, let's assume that users are generally not going to exhibit such behavior; rather, they will only work hard if they know they need to or if there is a really good payoff that will result. If you want a blunt or somewhat negative way to remember this idea, just assume users are lazy! More general rules of thumb about how users tend to act will be presented later in the chapter. The previous rule is simply presented to tie in with a few ideas about memory.

Now, assuming users will not like or even avoid Web sites that require them to work too hard, forcing them to memorize things is not a good idea. To illustrate this idea in practice, consider the interface of an automated telephone banking system. When you call the bank, you are prompted for your account number and then read a list of items and corresponding keys to press—"Press 1 for balance, press 2 for transfer, press 3 for payments... ." If you encounter such a system and are unfamiliar with all the choices, using the system can be difficult. You may find that you will try to remember a choice presented in your mind until all the choices have been presented. If too many choices are presented, you might not be able to recall the range of choices or you might even forget which item you chose and have to listen to the choices again. Now, if the same information were presented on a small text menu, it would be much easier to find the item. You would just look over the list and pick the appropriate one. The voice example requires you to recall the choices, which is very difficult. In general, it is easier for users to recognize choices than to recall them. Because users who make mistakes will then tend to favor easier-to-use systems, we should always try to rely on recognition over recall.

Rule: Recognition is easier than recall.

There are plenty of examples of how recognition is easier than recall. Students generally consider a multiple-choice test to be easier than a fill-in test. You must study, of course, for each (assuming the tests are created correctly), but the amount of memorization required is much higher for the fill-in test. The multiple-choice test doesn't require the depth of memory because you will see the answer and recognize it (hopefully) with only a minimal amount of "recall effort."

It turns out that many of the rules and suggestions presented in this book ultimately are related to this idea of recognition being easier than recall. For example, consider the idea of modifying link color. If we turn off link coloring so that links never look visited, we are forcing users to recall whether they have selected a certain link before. If the links do change color, users simply have to recognize the different color to know they have been there before.

Rule: Do not make visited links the same style or color as unvisited ones.

Another important aspect of memory to consider is that it isn't perfect. Users are not going to memorize things easily and often will have only partial memory or a flawed memory of something. Just as in real life, repetition will lead to improved memory. For example, frequent users or power users may actually rely on memorization of the location of objects on the screen, but most users will have only vague memories of how link choices or pages are organized. However, when people are memorizing things, it is known that image memory is one of our most robust forms of memory. It is far easier to retrieve pictures or even words or ideas that evoke pictures than it is to retrieve abstract ideas without visual cues from memory. It is often far easier to remember a person's face than it is to remember the person's name. Given that users will generally find it easier to remember visuals, it would be wise to make pages that should be remembered visually different from the rest. For example, in site navigation, a home page serves as a safe zone for a user. Using a distinct image or a different color is important in making the home page memorable. However, do not assume the user to have perfect memory—don't make the home page only subtly different from the other pages or expect the user to notice or memorize text items.

Suggestion: Make pages that should be remembered visually different from the rest.

Another aspect of memory that is important to the usability of Web pages is the amount of information a person can recall from short-term memory. Let's return to the automated phone banking system example. When users hear the choices, they have to memorize them. If too many choices are presented, they might forget an item. This is an example of short-term working memory. In a sense, we need a little scratch space in our brain to remember something for a few moments. This memory does not hold a great number of items and is highly volatile. Cognitive scientists have long been interested in short-term memory and have conducted many experiments where participants are presented random objects or words and asked to quickly look at them or to make choices from them to test short-term memory. What is found is that participants are able to recall a range of seven items, plus or minus two, from short-term memory. What this means is that when given five to nine items, the user will be able to recall all the items for a short period of time and have them equally present in mind for choosing among them.

The implication of users being able to remember quickly 7 (±2) items on Web design may or may not be profound. If you present a user with a set of links, shouldn't you limit the choices to from five to nine? It would seem you should—if you want the user to choose from the choices "fairly." For example, if you present a list of dozens of what may appear to be randomly ordered links to a user, you will find that the user will have a tough time picking from them. You may notice that users will tend to favor extremes. In practice, the author has seen this happen on Web sites. For example, a large music site faced a problem in that bands listed in the site having names beginning with A or Z had a much higher download rate than anything else. What was happening is that users had little knowledge of the bands, so they would scan the lists and—unless something jumped out at them—they tended to choose the first or last items in the list to see what happened. They really couldn't remember all the names of the bands that were interesting as they went along—there were just too many of them. If you want users to easily choose from a list of things that are equally important, you should limit your set of choices to between five and nine items.

Suggestion: Limit groups of similar choices such as links to between five and nine items.

However, do not go overboard with the five to nine items idea. Some usability experts, in fact, believe this rule has no place on the Web. This seems unwarranted, given the support for the rule both from long-term human capabilities studies and from GUI practices, which tend not to put 100+ choices on a single screen. However, there is some merit to the idea of not putting too much stock in the 7 (±2) rule. Consider that some designers might be tempted to use this rule to suggest that pages should have only five to nine links on them. However, this could be rather limiting if you have a lot of content. Users can focus on items progressively. Consider, for example, being presented five to nine distinctly different clusters of links on a page. Maybe the clusters are labeled and colored so the user chooses a cluster after looking at each. Once in the cluster, there are five to nine links. In this sense, there might be as many as 81 links on a screen, and the user will still be able to use them easily. When looking at well-designed pages with numerous links, you hopefully will see fewer than 100 links and notice that the clustering used an organization method, such as alphabetical, to avoid memorization.

Memory rules of thumb can also be applied to clicks. It appears that users are able to remember about three pages presented sequentially. Anything more than that and there tend to be gaps in memory. For example, as users click through dozens of pages, they will probably remember a variety of pages but not all sequentially. The memorable pages may be visually different enough to trigger recall. Usually, such distinguishable pages are termed *landmarks*—the most obvious landmark page in a site being, of course, the home page. However, if you want users to remember a path, they tend to remember only about three page views sequentially—and maybe fewer if the pages look nearly identical. Therefore, you should not expect a user to memorize a sequence or path longer than three items without repeated use. The number of markers showing location and path in today's sites and the user's continual reliance on the Back button and browser's history mechanism demonstrate how tenuous sequential memory tends to be. Because of these memory constraints, we tend to see many sites trying to reach content within three clicks or complete transactions in as few screens as possible.

Suggestion: Aim for memorization of only three items or pages sequentially.

This is by no means a complete discussion of memory, but it does serve to remind Web designers that, in order to make a site easy to use, we need to limit the amount of memorization going on. The less effort the users expend trying to recall what sequence of buttons they pressed or what choices they may have seen, the better.

Response and Reaction Times

If you have watched people browse around Web sites, it is obvious that some people are faster than others. Some users appear to cut quickly through page content and make choices rapidly, and are frustrated with even the slightest download delay. Others struggle to keep up and seem to have the patience of Job when it comes to waiting for pages to load. However, over time you'll come to find that people's patience for Web page loading will go away, particularly as they become more frequent users. Consider, for example, how long it takes for users to become annoyed at an automated teller machine that has not returned their money to them. The entire transaction may only take a few seconds, but customers are quickly annoyed. But when automated tellers first came out, a wait of even 30 seconds to a minute seemed tolerable compared to waiting in a long bank line.

Tip: Users tend to be more patient with something they are unfamiliar with or that is a novelty.

We see this novelty/patience dynamic on the Web all the time. Sites that could be considered single-visit sites, like movie promotion sites or designer portfolios, get away with huge download times. These sites could be termed single-visit or "throwaway" sites, since the user is unlikely to return. Splash pages, excessive animations, and long downloads are less annoying to a user who hasn't seen them before, but patience wears thin on return visitation. Consider that even when a splash page has a "skip intro" button, a return visitor will still be frustrated with having to even make such a choice. The very fast loading design of successful, heavily frequented sites, such as portals or e-commerce sites, shows that patience wears thin. The needs and desires of the first-time visitor, who in some sense could be considered a novice user of the site, are different than the frequent or expert user of a site. However, users do not have infinite patience, and they are getting more and more impatient as they get used to what facilities the Web, or a particular site, provides. In general, we find the following rule to hold true:

Rule: The amount of time a user will wait is proportional to the payoff.

The better the payoff, the longer the user will wait. Users who get something for free or who are stealing some desirable piece of software or music seem to be willing to wait an eternity. Consider users who illegally download software, songs, and movies from the Internet with a modem. They'll literally spend hours searching for and downloading songs when they could have gone out, worked at a near-minimum-wage job, and earned enough to purchase an entire CD in a similar period of time. Of course, this imbalance will certainly change with the increase in bandwidth—much to the annoyance of the music industry. But it remains true that if you are going to expect a user to wait for a page to load, there better be something useful there.

The amount of time users will wait will vary based on the individual user, his or her personality, and the potential benefit of waiting. However, there are some things we can say about response and reaction times for users in general. Some usability experts (for example, Jakob Nielsen, www.useit.com) relate that studies about response times report similar results. Common response times and user reactions are summarized in Table 2-1.

Time Elapsed	Probable User Reaction
0.1 second	When something operates this fast or faster, it appears instantaneous or nearly instantaneous to the user. Unfortunately, due to bandwidth and technology constraints, few Web pages will exhibit this level of responsiveness in the near future.

Table 2-1. *Response Time and User Reactions*

Time Elapsed	Probable User Reaction
1.0 second	When something reacts in around a second, there is no major potential for interrupt. The user is relatively engaged and not easily distracted from what is happening on the screen.
10 seconds	This is suggested to be the limit for keeping the user's attention focused on the page. Some feedback showing that progress is being made is required, though browser feedback such as a progress bar may be adequate. However, do not be surprised if the user becomes bored and decides to move on to something else.
> 10 seconds	With a delay this long, the user may actually go about other business, look at sites in other windows, talk on the phone, and so on. If you want users to continue to pay attention, you will have to give them constant feedback about progress made and some sense of when the page will be finished (as when downloading software, the browser lets users know how much time is left to complete the download).

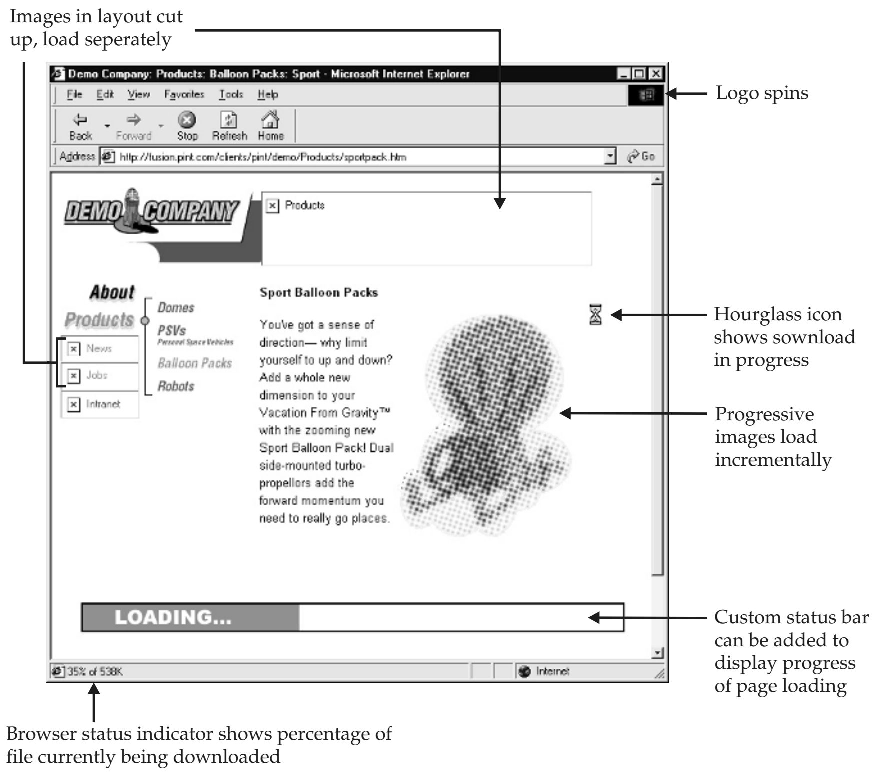
Table 2-1. *Response Time and User Reactions* (continued)

When it comes to the Web, there is generally little chance of going too fast for the user. Most of the time, it takes more than a few seconds even on a broadband connection to download something. However, be careful once something like a Flash file is downloaded. If the user has a faster processor than you, the program may end up running much faster on their system than expected, so much so that the user might not be able to keep up. On occasion, you may notice how animations used in some Web pages appear to travel at a rate only a superhuman could read.

Tip: Be careful with overly fast response times of downloaded objects.

In most cases, a Web site will probably not outpace the user; in fact, it may be much too slow for the user's liking. Because users may get impatient, you need to make sure that they are given some indication of the progress being made. The browser itself actually gives a great deal of feedback about the progress being made. When loading a page, a browser will generally convert a cursor to a wait indicator (such as an hour glass), spin or pulsate a logo (generally in the upper-right corner of the browser), provide a progress meter towards the bottom of the screen, and display messages about objects being loaded in the status bar at the bottom of the screen. The Web designer will design pages to provide even more feedback. For example, the designer may build pages so text loads first or pieces of the page are loaded one at a time. Often, designers will cut images up into multiple pieces, so the user will see a little bit loaded at a time. Also, designers often use images that load in a progressive fashion from an unclear one to a sharp one so that the user is able to get a general sense of a complete fuzzy picture early on and watch its loading progress, if necessary. Figure 2-2 illustrates all these progress indicators in action.

For page loads that only take 10–20 seconds, the feedback given by the browser and incremental loading of a page should be enough to let the user know something is going on. However, when loading takes longer, you should give the user more information. For example, many sites that use binary technologies like Flash use a special loading page complete with a status bar showing progress. Such progress meters can also be created using technologies like JavaScript. However, don't bother with a progress bar


Images in layout cut up, load seperately
Logo spins
Demo Company: Products: Balloon Packs: Sport - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address http://lusion.pint.com/clients/pint/demo/Products/sportpack.htm
Go
DEMO COMPANY
Products
About
Products
News
Jobs
Intranet
Domes
PSVs
Balloon Packs
Robots
Hourglass icon shows sownload in progress
Progressive images load incrementally
LOADING...
Custom status bar can be added to display progress of page loading
35% of 538K
Internet
Browser status indicator shows percentage of file currently being downloaded

Sport Balloon Packs

You've got a sense of direction— why limit yourself to up and down? Add a whole new dimension to your Vacation From Gravity™ with the zooming new Sport Balloon Pack! Dual side-mounted turbo-propellors add the forward momentum you need to really go places.

Figure 2-2. *Browser and site feedback shows progress of download*

or other forms of feedback unless load times are around 30 seconds or more. (With the proliferation of broadband Internet access, this time will certainly diminish; even now, many broadband users are likely to get impatient around 20 seconds or less, and in due time even 10 seconds may seem like a long wait.)

Rule: When response times such as page loads take more than 30 seconds, try to provide your own feedback to the user, such as a load-time progress bar.

If you are building a static site, there are some simple tricks to let the user know about a longer wait for an object. For a very large image download, besides interlacing the image or having it show up progressively sharper, it is also possible to use a trick with the **<img>** tag's **lowsrc** attribute. You could load a low-resolution version of an image first, or even a graphic message stating the image is loading, like so:

```
<img src="hirezpicture.jpg" lowsrc="lowrezpicture.jpg" height="1000"
width="1000" />
```

Or, you might have a message display instead. Some designers have even experimented using the **alt** attribute of an image to show file size or a loading message, like so:

```
<img src="hirezpicture.jpg" alt="Loading picture of Mars (800K)"
height="1000" width="1000" />
```

Of course, it is probably better to reserve the **alt** text for its primary purpose—providing an alternative rendering for users without images. Another HTML or CSS trick that can be used to let a user know about a long download is to use a background image with a message on it that says a page is loading, which is eventually covered up by content that is being downloaded. Other forms of loading screens can be created in both JavaScript and Flash. An example using these techniques is shown in Figure 2-3.

When attempting to create a site that appears responsive to a user, remember that time is what matters the most. How users actually perceive a page loading will not necessarily equate to the bytes delivered. A user who isn't paying for bandwidth isn't going to care if 1K or 100MB is delivered, as long as it appears fast to them.

Rule: Time matters more to a user than bytes delivered.

Because time is so important to a user, it is important to take advantage of every second. Consider that the general way users navigate the Web is to look at a page scan to find an appropriate link, click, and then wait for the page. Once the page loads, they then look at the page to find the next link or spend time consuming the content. Notice the time is split between user "think time" and download time. The reality is that for most users, the think time for navigation pages is pretty small compared to the wait

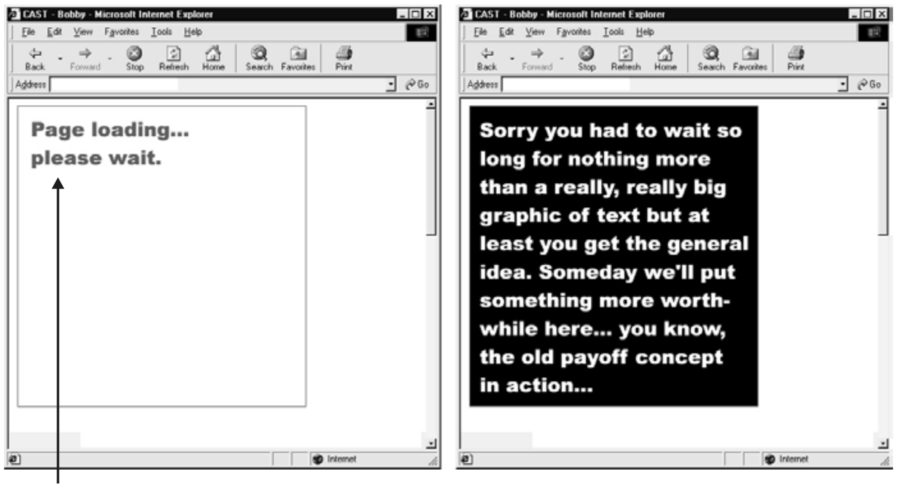

CAST - Bobby - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Print
Address
Go
Page loading... please wait.
Sorry you had to wait so long for nothing more than a really, really big graphic of text but at least you get the general idea. Someday we'll put something more worth-while here... you know, the old payoff concept in action...
Internet
Alt text or background image showing status

Figure 2-3. *Let the user know a long download is in progress*

time. For content pages, however, the user may spend a great deal of time looking at the page. One way to improve responsiveness is to take advantage of the thinking time by downloading information to be used later on. This is often called *preloading* or *precaching*. Assuming you are able to preload most or all of the next thing to be looked at by the user during the think time, the next page load time could be significantly reduced. Somewhat like the magician who has the result of a trick set up in advances, downloading during idle moments can produce a nearly mystical appearance of speed.

Suggestion: Improve Web page response time by taking advantage of user "think time" with preloading.

A variety of browser accelerator tools have been built in an attempt to improve Web responsiveness by preloading pages linked from the current page. The only problem with this approach is that many pages have so many outbound links that it is difficult for the browser to predict the page the user will load next. The best way to improve the odds of caching the correct "next page" is to look at the common paths users take through a site by examining a log file and then putting in code to preload pages along these paths. However, this just improves the odds. The only time you can really guarantee that preloading will improve things is when the user is navigating a linear progression of pages.

The responsiveness of a Web site is a key aspect of a user's feeling of the site's usability. Beyond loading of pages, consider that time is important to a user even after a page has loaded. For example, if a page loads quickly but users can't figure out what is going on in the page within about ten seconds, they can become just as frustrated as waiting for a simple page to download. Aim for what might be called the "ten-second Web page." A ten-second Web page is one where the user gets the gist of the page in about one minute and can decide after that whether to consume the content more seriously or not.

Tip: Aim for a ten-second limit for the user to determine the basic gist of a page's content or purpose after loading.

Dealing with Stimulus

Users are constantly being bombarded by stimuli from our sites. The text, the links, the graphics, animation, even sound all create a cacophony of information that the user tries to distill meaning from. Because of the continual stimulation, we need to filter out some of the data, and we do this both unconsciously and consciously. Three primary ways it is thought that people filter sensation data include thresholds, something dubbed the "cocktail party effect," and sensory adaptation.

Thresholds

Rather than deal with every minute change that happens, we tend to notice only something that exceeds a particular threshold. For example, if on a Web page an object moves very slowly—say a pixel every few seconds—we may not notice at first because the speed of its movement is below our absolute threshold. However, over time we may notice the movement. Thresholds are tough to predict. Depending on their psychological state, users may be able to detect something under normal conditions; but, if they are tired or distracted, they may not be able to notice the difference between two similar but different colors or fonts that have been used to separate navigation forms.

When designing pages, designers should always consider thresholds. Thresholds suggest making objects or pages noticeably different from each other so that users will be easily able to understand their difference. For example, consider if link and text color in a page are too similar. The user may have to carefully inspect underlined text to make sure that it is a link and not just underlined text, because text colors are only subtly different. In other words, they might not always be sure what's a link and what isn't without putting in at least some degree of effort. Designers should strive not to force the user to spend time and effort trying to interpret the differences between objects on a page, since it both is frustrating and takes time away from the main goal of getting the user to consume the content or perform a task. Consider the threshold effect when trying to differentiate objects on a page.

Suggestion: Make page elements obviously different if they are different.

Things need to be just different enough for the user to notice. If the designer is too subtle, however, the user may not be able to tell. And if you go overboard, the design may backfire. It would be easy enough to always put site buttons in bright colors and content in dark colors, but this could be annoying to the user. The next two ideas show how users tend to filter out information when being bombarded with excessive stimuli.

Cocktail Party Effect

The cocktail party effect describes how people are able to concentrate on important data when being bombarded by nonessential stimuli. People at a cocktail party can concentrate on their own conversation despite being in a room filled with numerous other conversations. Don't dismiss the other conversations as background noise. If the listener stopped and focused on another conversation, he or she probably could hear certain parts of it. However, the threshold effect is also in play during a cocktail party. If the person you are trying to listen to speaks too softly, if the proximity of other conversations is too close, or if the volume of other conversations is too loud, the listener will be overwhelmed by the outside stimuli.

Web page designers should consider that, as in cocktail party conversations, the user might want to concentrate on only a small portion of the information on a page. The rest is background noise that has to be filtered out. If there is too much going on, users will not be able to effectively concentrate on what they want and become frustrated. Therefore, we should try to section things off just as in a cocktail party, so the user can effectively concentrate. A good site has lots of choices but provides the visitors the ability to focus on what they want to see. Toward this end, we might consider grouping similar items together and separating groups of items with a lot of white space. Also, within text, we might convey important points in a bullet list or a pull quote, or highlight them with a background color. Always strive to limit noise—namely, competing objects on a page. If you don't, and the site is like the cocktail party that gets too loud, users won't be able to filter out information that isn't important to them.

Suggestion: Limit page noise and segment page objects so that they don't compete so much visually that users are unable to focus on what they are interested in.

Thresholds and the cocktail party effect present a balance between having too little of a difference and too much. Don't become so concerned with trying to get an absolutely perfect balance of stimuli—just try to get it about right. You may consider erring in favor of a little too much, since people are very adaptable, as shown by the next cognitive science idea.

Sensory Adaptation

Sensory adaptation occurs when users become so used to a particular stimulus that they no longer respond to it—at least not consciously. Think of the watch on your wrist. You probably don't notice it normally. Take the watch and put it on your other wrist and you'll notice it for a while, but eventually you'll get used to it. That's sensory adaptation. Life is filled with things that people adapt to: the ticking of an alarm clock, the clothes you wear, the loudness of the music coming from your car stereo, and so on. Life on the Web is no different. Users adapt to Web stimuli quickly. That continually animated GIF that grabbed the user's attention once or twice quickly fades into the background.

Probably the most interesting sensory adaptation is the rise of so-called "banner blindness." People are becoming so used to the shape and location of banners that they are just tuning them out. Experiments as well as click rate studies show that people don't look at banners terribly attentively. Animation added to the mix improved things, but it, too, has succumbed to sensory adaptation. Rich banner ads complete with sound and complex interaction are being experimented with to see if they can regain user attention. And we have pop-ups that are quickly swatted away by users as fast as they spawn. The bottom line is that users will decide what they want to focus on. Designers may want users to focus on something such as a banner ad or a download button, but in order to grab their attention, they will have to continue coming up with new tricks as users adapt to stimuli over time.

Rule: Sensory adaptation does occur on the Web. If you want a user's full attention, you'll have to vary things significantly and often.

Sensory adaptation suggests that the numerous fonts, animations, and colored regions on a page may go unnoticed over time. This doesn't mean that we should completely avoid using things to stimulate the user, but we should not be as reliant on them, since they lose strength with use. Sensory adaptation really suggests that, in order to get a user's full attention, we have to "wake them up" with something different. A little bit of surprise can be useful to make the user pay attention. However, be careful with this idea. In general, users will want to peacefully go about their business and will expect pages to look and act consistently. We shouldn't disturb them, but should let them focus on the task or content at hand. If you bombard the user all the time, they will feel uncomfortable because of the lack of consistency, and they may become so annoyed that they leave.

Movement Capabilities

Once the user has absorbed information they have been provided, they will eventually react to it and make some choice. While someday voice interfaces may become commonplace, today's Web sites are generally manipulated using the keyboard or mouse. Therefore, we should always attempt to minimize user efforts using these devices. Few sites consider that users may prefer using the keyboard or arrow keys, instead of a mouse, to move through choices in a page. While many form pages are optimized for quick navigation via the keyboard, other pages may not be.

Rule: Try to optimize keyboard access for all pages in a site, not just form pages.

Consider also the work users perform moving their mouse around the screen. Moving the pointer around the screen takes effort, and a button or link press may take up to a few seconds if a user has to move a long distance or focus on clicking a very small button. In fact, the time it takes a user to press a button is governed by something called Fitt's law (Fitts, 1954). Fitt's law basically states that the smaller the button to press and the farther away it is, the longer it will take to perform the action. This seems logical, since users tend either to overshoot small click targets because they moved too fast or to take extra time to clock the button more carefully.

Fitt's law would suggest that to improve speed of use and thus efficiency, we should first bring things closer together. First, we might consider reducing the amount of mouse travel between successive clicks. Notice how efficient a wizard-style interface is, since after clicking "Next" the successive "Next" button tends to be directly under or very close to the current mouse position. There is no reason we couldn't apply this to navigation elements. Try to keep successively clicked buttons close together. Navigation bars tend to encourage following this plan, anyway.

Rule: Minimize mouse travel distance between successive choices.

However, with the Web, we can't always be sure that the user will press another button within the page as their next choice. In fact, quite often the user may move to a browser button such as the Back button rather than rely on internal site navigation nearby. Given some users' preference for the browser Back button, designers should try to minimize the mouse travel to the Back button. The question is, travel from where? We should assume that the user will probably hover over the navigation bar or near the scroll bars most of the time. While we can't decrease the distance from the scroll bars, which will tend to be far away from the Back button in the upper left of the screen, there is no reason that we should not consider putting primary navigation buttons on the left or top portions of the screen. Doing so will minimize the distance from a primary selection area and the heavily used Back button, thus reducing mouse travel and increasing the speed at which the site can be used.

Rule: Minimize mouse travel between primary page hover locations and the browser's Back button.

Fitt's law would also suggest that we make clicking targets larger, particularly if they are far away. Some designers find this design suggestion troublesome because it suggests making big huge buttons, which would take up a great deal of screen real estate as well as potentially making the site look like it was designed primarily for novice users. Big buttons also bring too much attention to the interface. However, buttons should be made big enough for users to mouse to them relatively quickly—and spaced out well enough so they are able to click them without accidentally click an adjacent choice.

Rule: Make clickable regions large enough for users to move to them quickly and press them accurately.

General user capabilities are not all that we need to consider when discussing what ideas affect usable Web design. We must also consider the world the user inhabits and the user's general and unique characteristics and experiences.

The User's World

People truly are the centers of their own universe, in the sense that they perceive everything initially from their own point of view. Consider the idea of how a user might perceive the Web site shown in Figure 2-4. The user lives in the real world.

Users are affected by their environment: the physical conditions of their location, the noise around them, the visual quality of the monitor they are using, and so on. From their world, they access your Web site via the medium of the Internet and the Web, which includes things like network connections, servers, browsers, and so on. Once on the Web, they navigate about and visit sites. If they decide to actually interact with a site, they finally begin to consume or react to the content presented.

The presentation and navigation layers in Figure 2-4 could be interchanged considering that a user's ability to navigate Web space is greatly affected by the way it is presented.

Suggestion: Always remember that you need to bring a site into the user's world, not the other way around.

The preceding suggestion is an important one. Designers will naturally believe that they have set the rules for their sites and that users are just visitors. While this may be true, users tend to interpret things from their own perspective. Each user will have his or her own opinions, capabilities, environment, and experiences, all of which will influence how the site is interpreted. A fine balance between what the user thinks and

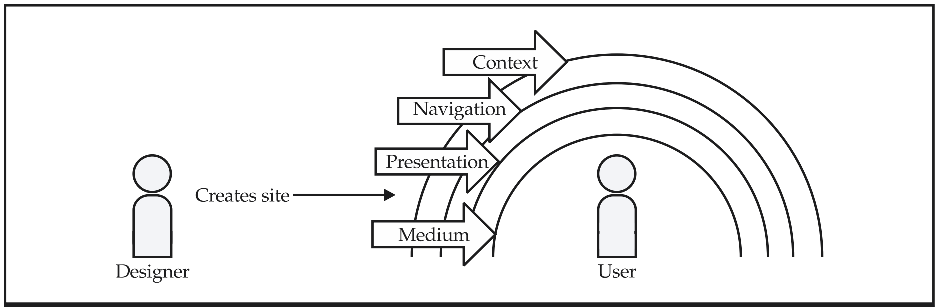


Figure 2-4. *The user's universe*

wants and what the designer thinks and wants has to be struck. This will be discussed in more depth later in this chapter.

User Environments

The user is heavily influenced by what could be called their *environment of consumption*. For example, consider a user in a public place such as an airport using a public Internet kiosk to remotely access their e-mail. The user is standing up—it might be crowded and noisy—waiting to dash off to the plane. Because of this environment, the user may not be tolerant of long waits, excessive menus, or anything that slows down the task at hand. Further, due to the noise, the user may not be able to always hear sound cues. Last, because the user is standing up, the amount of time they might spent during the whole online session will certainly be significantly less than a normal session at the office. When designing for users, always think about where the user is accessing the site from. Table 2-2 details some of the possibilities.

The environment will greatly affect the user's view of what is "usable." For example, color combinations that contrast acceptably indoors might be troublesome outdoors. Designers must take into account the environment of consumption.

Location	Characteristics
Office	Generally computer-based access Single user Relatively quie Should be primarily work or task focused, at least during primary work hours Often high speed
Home office or bedroom	Generally computer-based access Single user Noise level variable, but often quiet Purpose may be work or play Access could be anytime Speed of access varies dramatically from modem to high speed
Home living room	Access may be from set-top box or video game console Distance from device may be farther Use may be less input oriented (reduced typing) Noise level variable May be group-oriented access or single user Access probably more entertainment related Printing may not be an end result

Table 2-2. *Common User Environments Characteristics*

Location	Characteristics
Cybercafe	Probably computer-based access Cost may influence usage Noise level variable Use is probably entertainment or research Speed of access probably high May be group-oriented access or single user Security or privacy may be a concern
Public kiosk	Cost may influence usage Noise level variable User may be standing Use will be less input oriented (reduced typing) Use is probably task oriented, focused on e-mail or access to very important information Access to location-related information may be a high priority Security or privacy may be a concern
Car	Probably noncomputer-based access (PDA or smart phone) Use will be less input oriented (reduced typing) Focus will not be primarily on the access if user is the driver Use is probably task oriented or limited to very important information Access to location-related information may be a high priority Speed and quality of access is probably low
Mass transit or plane	Probably either noncomputer-based access (PDA or smart phone) or a laptop User may be standing or sitting Use could be entertainment or work Access to location or time-sensitive information may be a high priority Speed and quality of access is probably low Security or privacy may be a concern
Outside	Probably noncomputer-based access (PDA or smart phone) Screen glare could be a significant problem Use will be less input oriented (reduced typing) User may be standing or moving Noise level variable Use is probably task oriented or limited to very important information Access to location-related information may be a high priority Speed and quality of access is probably low

Table 2-2. *Common User Environments Characteristics* (continued)

Rule: Account for the characteristics of the probable environment in which the user will access a site if possible.

General Types of Users

There are three levels into which users can be classified to reflect their knowledge of how to use a Web site: novices, intermediates, and experts or power users.

A novice user is one who may have little knowledge of a site or even of how the Web works. A novice user will need extra assistance and may prefer extra clicks with extra feedback to accomplish a simple task. An example of an interface tuned to novices would be a wizard that automates some common task.

At the other end, power users are those users who understand the Web or a site very well. Power users should be considered in two distinct categories: frequent and infrequent visitors to the site. A power user who frequently visits a site to utilize advanced features such as sophisticated searching, may directly form their own URLs, and memorize the position of objects within a page or the site. A power user who is an infrequent visitor to a site may not be familiar with the site's structure but will expect certain facilities, such as search, to be available to navigate a site. Power users will need relatively little handholding and will desire to click less and consume more. Obviously, the distance between a power user and novice user is great. A site geared too much toward one audience or the other will certainly annoy—the power user if the site has been dumbed down, or the novice user if the site is geared mostly toward power users.

The third group of users, the infrequent intermediate user, is actually the largest category of users on the Web. Most users are infrequent intermediate users because they pretty much understand how the Web works, but may not know how to navigate a particular site in a very efficient manner. Infrequent intermediate users do not continually revisit the site; if they do, they will probably eventually become a power user. Because site usage tends to be dominated by intermediate users, you may consider designing the site for the capabilities of these users. However, doing so may lock out novice users and bore or restrict advanced users.

The best approach to building a site for basic user groups is to build a site that provides features that cater to all users. Software applications do this, so there is no reason a Web site cannot. A software application may provides keyboard shortcuts and other features, such as customizable interfaces, for power users while also providing icons and menu systems for intermediate and novice users. Help systems and wizards are other features mostly geared toward the novice user. A Web site could provide features like a clean URL system, advanced search facility, and personalization features for an advanced user. A site with consistent navigation bars that have button labels similar to other sites (About, Products, Careers, and so on) is very friendly to novice and intermediate users, and it can also have dynamically built “bread crumb”–style navigation lines, popular with advanced users. Last, a Web site could provide help systems, maps, and alternative forms of access such as simple text links for the novice.

Suggestion: Aim to create an adaptive Web site that meets the requirements of novices, intermediates, and advanced users.

In the perfect world, there is no reason that a Web site can’t be built to meet the needs of all general user groups. However, time and cost constraints may limit the number of features that can be added to some Web sites. In such cases, it is probably best to aim for the largest group of users: the intermediate. This may lock out some novice users unable to figure the site out. There is an argument to be made for aiming at the lowest common denominator in a user. The problem with this is that if you start building only for the complete novice, you can quickly alienate users who know what they are doing.

Suggestion: Design for the intermediate user if an adaptive Web interface is not possible.

Even if an adaptive interface is built, there is bound to be a user who doesn’t understand or like the site we have built.

Tip: Remember there will always be users who don’t like or get a site, no matter how good it is.

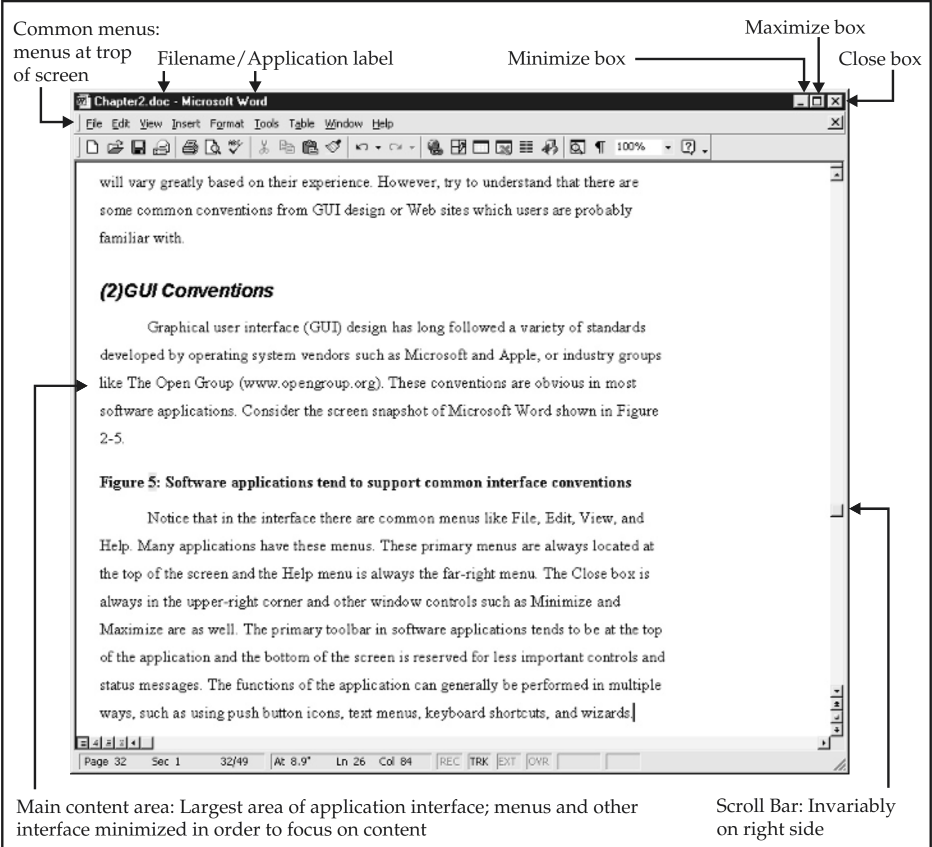
Users are individuals with different tastes and opinions. They will have different experiences, capabilities, personalities, age, gender issues, and cultural issues. Some individuals may have disabilities that prohibit them from using a Web site that most users find easy to use. Users bring what they know from the real world and from other Web sites to your site. They may expect to use symbols from the real world, such as those for navigation. However, they may also bring knowledge of how Web sites work that they gained from visits to many other sites. Knowledge of how traditional software applications work may also be brought into play. Remember, as mentioned early in the chapter, that users bring the site into their world—they don’t visit the universe of your Web site. Your site is just a speck in an overall universe of Web sites. In fact, it could be said that most of the time users are not at your site. Some call this the 99 percent rule, since 99 percent of the time, users are probably not using your site. You should, therefore, make sure that your site follows any conventions and meets expectations set up by other sites.

Rule: Users bring past experiences with the world, software, and the Web to your site. Make sure your site meets their expectations.

You need to make sure that your site acts like other sites or software users have used and meets their general expectations. Remember the rule of consistency: if you do things differently from how everybody else does, you can’t rely on a user’s past knowledge and you force the user to learn something new. Of course, the challenge with real users is that expectations will vary greatly based on their experience. However, try to understand that there are some common conventions from GUI design or Web sites that users are probably familiar with.

GUI Conventions

Graphical user interface (GUI) design has long followed a variety of standards developed by operating system vendors such as Microsoft and Apple, or industry groups like The Open Group (http://www.opengroup.org). These conventions are obvious in most software applications. Consider the screen snapshot of Microsoft Word shown in Figure 2-5.


Common menus: menus at trop of screen
Filename/Application label
Minimize box
Maximize box
Close box
Chapter2.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
100%
Page 32 Sec 1 32/49 At 8.9" Ln 26 Col 84 REC TRK EXT OVR
Main content area: Largest area of application interface; menus and other interface minimized in order to focus on content
Scroll Bar: Invariably on right side

will vary greatly based on their experience. However, try to understand that there are some common conventions from GUI design or Web sites which users are probably familiar with.

(2)GUI Conventions

Graphical user interface (GUI) design has long followed a variety of standards developed by operating system vendors such as Microsoft and Apple, or industry groups like The Open Group (www.opengroup.org). These conventions are obvious in most software applications. Consider the screen snapshot of Microsoft Word shown in Figure 2-5.

Figure 5: Software applications tend to support common interface conventions

Notice that in the interface there are common menus like File, Edit, View, and Help. Many applications have these menus. These primary menus are always located at the top of the screen and the Help menu is always the far-right menu. The Close box is always in the upper-right corner and other window controls such as Minimize and Maximize are as well. The primary toolbar in software applications tends to be at the top of the application and the bottom of the screen is reserved for less important controls and status messages. The functions of the application can generally be performed in multiple ways, such as using push button icons, text menus, keyboard shortcuts, and wizards.

Figure 2-5. *Software applications tend to support common interface conventions*

Notice that in the interface in Figrue 2-5 there are common menus like File, Edit, View, and Help. Many applications have these menus. These primary menus are always located at the top of the screen, and the Help menu is always the far-right menu. The Close box is always in the upper-right corner, and other window controls such as Minimize and Maximize are there as well. The primary toolbar in software applications tends to be at the top of the application, and the bottom of the screen is reserved for less important controls and status messages. The functions of the application can generally be performed in multiple ways, such as using push button icons, text menus, keyboard shortcuts, and wizards.

GUI conventions are very useful to know, particularly when designing forms and other interactive elements of a site. In later chapters on implementation, we'll discuss the use of GUI widgets and the difference between Web and GUI interfaces. The Web has not been able to develop conventions that are as well understood as those for software applications. There are two main reasons for this. First, software applications are often defined significantly by the operating system they are written for. Microsoft has great influence on how applications written for Windows should work. Apple can dictate conventions for Macintosh software. Second, the ability to author and distribute software applications is restricted to a much smaller group of people than in Web design. Many Web designers lack any formal understanding of GUI conventions and may actually shun them in favor of artistic freedom. This struggle is fortunately changing, as the focus on user-oriented site design becomes more popular.

Web Conventions

While Web sites may not exactly follow GUI usability conventions, they do have a loose set of conventions. Straying from the way that most Web sites work is a dangerous idea. Unless you happen to be running an important day-to-day use site like an internal site, a heavily trafficked site like Amazon, or a portal like Yahoo!, you will probably not be able to introduce any conventions of your own. In fact, if users come to expect that a company logo in the upper left-hand corner of the screen will return them to the home page, you had better do this in your site. If you don't do this, you may surprise the user, which could cause a negative reaction. Forcing the user to learn a new idea also could cause a negative feeling.

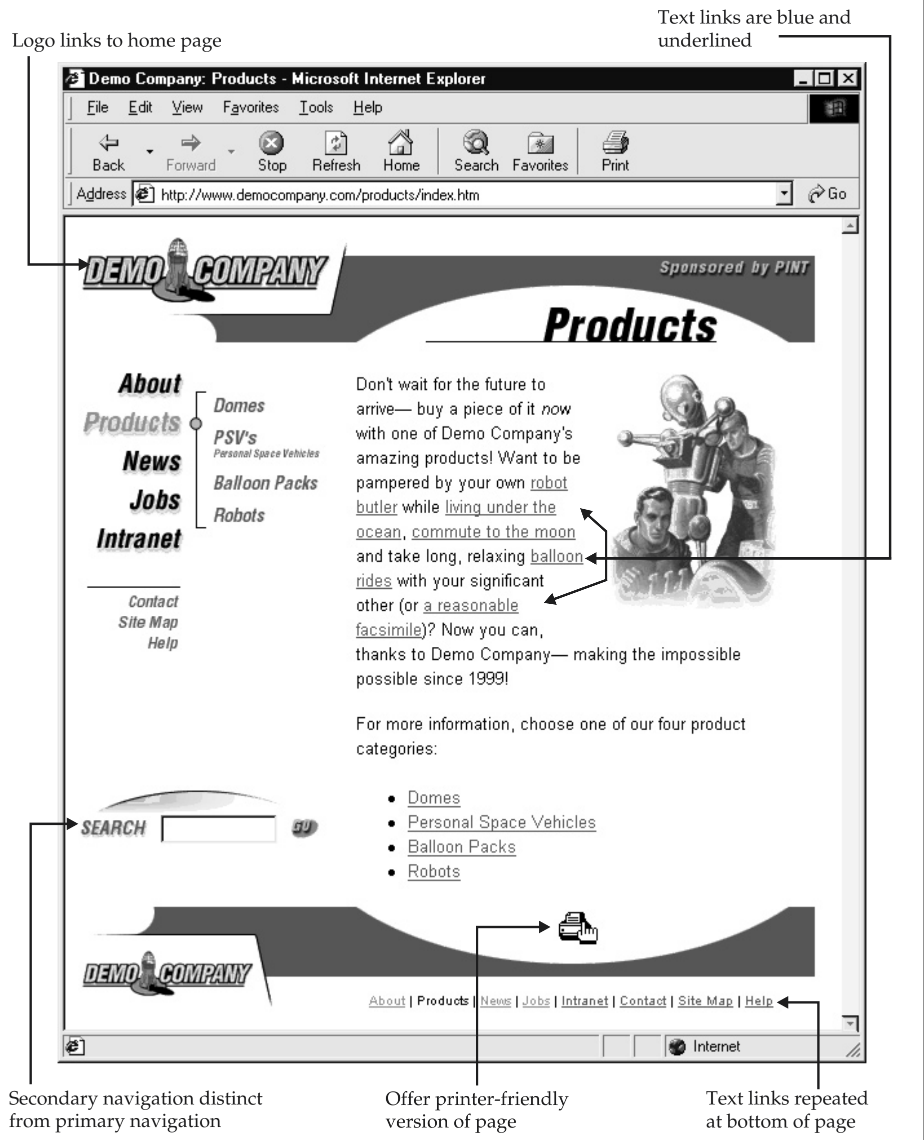
Rule: Do not stray from the common interface conventions established by heavily used sites.

Web conventions, unfortunately, are difficult to pinpoint. A few well-known ones are summarized in Table 2-3.

Convention	Description
Upper-left corner logo signals home page return.	Users tend to expect a corporate logo to return them to the home page. Most sites put this in the upper-left corner. An explicit Home button, as well as a ToolTip, is a good idea.
Text links are repeated at the bottom of a page.	Most sites like to repeat text navigation at the bottom of a page, particularly if top or side navigation is a graphical form.
Back-to-top link used on long pages.	While sites will provide text navigation to move to the next page, a back-to-top link or arrow is generally included at the bottom of the page to quickly jump the user up the page.
Special print forms used for heavily printed pages.	Increasingly, sites are providing special printer-friendly versions either in a stripped HTML form or even in an Acrobat form. This is most commonly found on sites that distribute large volumes of content.
Shopping cart in the upper right.	Typically, the shopping cart icon or link is found in the upper-right corner of the screen.
Clickable items are blue and underlined.	Fight it all you want, but most text links are blue and underlined. While many users may be able to understand nonunderlined links or different colors, the best way to signify that something is pressable is making it blue and underlined. Be careful with creating logos or other content that are blue, as users may actually try to click on it.
Secondary navigation elements such as a site map or search are presented separately from sectional navigation.	Because site maps, site indexes, help systems, and search facilities are navigation aids, most sites have tended to put less emphasis on links to them. However, given the rise in popularity of search features, content-rich sites may emphasize search facilities.

Table 2-3. *Some Common Web Conventions*

Figure 2-6 illustrates some of the common Web conventions used in a page within the DemoCompany site.


Logo links to home page
Text links are blue and underlined
Demo Company: Products - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Print
Address http://www.democompany.com/products/index.htm Go
DEMO COMPANY
Sponsored by PINT
Products
About
Products
News
Jobs
Intranet
Domes
PSV's
Personal Space Vehicles
Balloon Packs
Robots
Contact
Site Map
Help
SEARCH GO
About | Products | News | Jobs | Intranet | Contact | Site Map | Help
Internet
Secondary navigation distinct from primary navigation
Offer printer-friendly version of page
Text links repeated at bottom of page

Don't wait for the future to arrive— buy a piece of it *now* with one of Demo Company's amazing products! Want to be pampered by your own robot butler while living under the ocean, commute to the moon and take long, relaxing balloon rides with your significant other (or a reasonable facsimile)? Now you can, thanks to Demo Company— making the impossible possible since 1999!

For more information, choose one of our four product categories:

- Domes
- Personal Space Vehicles
- Balloon Packs
- Robots

Figure 2-6. *Web conventions in practice*

The problem with Web conventions is that they are moving targets. New conventions may be invented and sweep across the Web like fads. For example, frames and splash pages used to be popular, but they have somewhat fallen out of favor. Conventions are not always well considered and may often have more to do with novelty than usability. However, this shouldn't lead you to invent new conventions or avoid those that are current. The best way to keep up with current conventions is to simply browse the well-trafficked e-commerce and content sites often and look for common features. If users are exposed to features there, such as single-click ordering, it isn't going to be difficult to explain to them how it works on your site. Don't assume that everyone understands common conventions or that all users will be able to use current conventions. Some users will have special needs.

Accessibility

There is no way to account for all the small differences between people. In fact, we only aim to create sites that *most* people like. This may lead us to stereotype groups of users (like casual female surfers under 18, and so on), but this may be an approach we have to make. Yet, this does not mean that you should go out and build a site catering to the largest demographic group of users hitting your site. Try to please as many distinct groups as possible by making your site as accessible as possible. Don't forget that some people may have difficulty if you assume that all users have perfect physical and technical capabilities.

Providing accessibility for people who may have deficiencies involving sight, hearing, or other physical capabilities isn't just a nice idea anymore—it may actually be required for some organizations, particularly government agencies—and many companies could incur serious liability if they do not account for all users. For example, Section 508 of the 1986 Federal Rehabilitation Act requires that the federal government include solutions for employees with disabilities when awarding contract proposals. This would also eventually apply to systems such as intranets, extranets, and most likely public Web sites. Also, the 1992 Americans with Disabilities Act (ADA) states that firms with 15 or more employees provide reasonable accommodation for employees with disabilities. This could apply to intranets or extranet creation!

But making a Web site accessible is something that should be done, not because of some law or to avoid future litigation, but because doing so could result in a much better Web site for everyone. Very often, creating systems that are accessible to all users also creates benefits for all users, regardless of capability. For example, the so-called talking books, initially considered for the blind, fostered books on tape. Also consider that easy ramps to access buildings, and curb cutouts made for wheelchairs, make walking easier for all and tend to reduce the number of people falling flat on their face after crossing the street or severely twisting their ankles as they step off the curb.

The W3C (http://www.w3.org) has long advocated designing sites for maximum accessibility and promotes the Web Accessibility Initiative (http://www.w3.org/wai). The WAI is concerned not only with creating sites that are accessible to people with disabilities, but also with making sites that are accessible by anyone who might be operating in a different environment than what a designer considers "normal." Remember that users will not necessarily be using a fast connection and a large monitor like yours—or if you aren't using a fast connection with the latest and greatest, your users just might be! From the W3C guides, you should always consider that users may have different operating constraints:

- They may not be able to see, hear, or move easily, or may not be able to process some types of information easily (or even at all).
- They may have difficulty reading or comprehending text because of language problems.
- They may not be able to use a keyboard or mouse because of access method (such as a cell phone) or physical disability.
- They may have a less-than-ideal access environment, such as a text-only screen, a small screen, a screen without color, or a slow Internet connection.
- They may be accessing the site in a nonstandard environment where they may be affected by environmental factors—accessing the Web in a noisy cybercafe or as they drive a car, for instance.
- They may have an older browser or a nonstandard browser or operating system or use an alternative form of user interface, such as voice access.

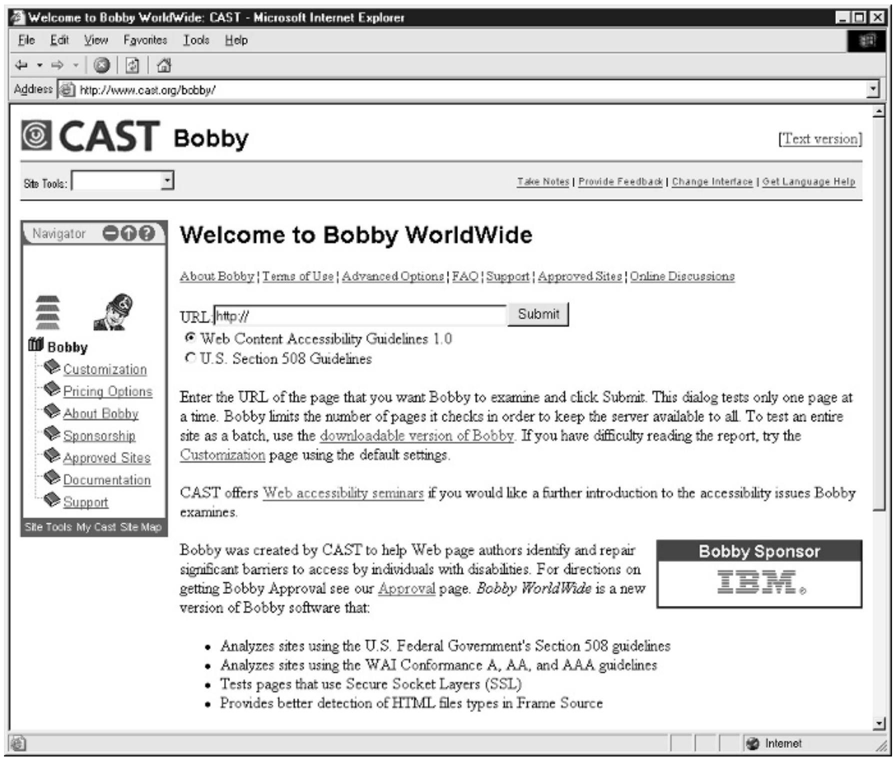
To deal with these issues, the W3C has issued a few suggestions to improve the accessibility of a site. These are summarized here:

- **Provide equivalent alternatives to auditory and visual content** In other words, don't rely solely on one form of communication. If you use picture buttons, provide text links. If audio is used, provide a text transcript of the message, and so on.
- **Don't rely on color alone** As discussed earlier in the chapter, not everyone will be able to view colors properly; so if color alone is used to convey information, such as what constitutes a link, people who cannot differentiate between certain colors and users with devices that have noncolor or nonvisual displays will not be able to figure out what is being presented. In general, you need to consider avoiding color combinations with similar hues or not enough contrast—particularly if they are likely to be viewed on monochrome displays or by people with different types of color vision deficits.
- **Use markup and style sheets, and do so properly** Basically, make sure to use HTML for structure and CSS for presentation. Especially avoid using proprietary markup or presentation elements and avoid using technology that may not render the same way in different browsers.

- **Clarify natural language usage** Make sure to define terms and use markup that indicates acroymns, definitions, quotations, and so on. In other words, use more logical markup. Further, make sure to clearly indicate the language being used in the document so that a browser may be able to switch to another language.
- **Create tables that transform gracefully** In short, don't use tables for layout—use them for presenting tabular data such as a spreadsheet. When tables are used, provide a clear caption, column headings, and other indicators of the meaning of cell contents.
- **Ensure that pages featuring new technologies transform gracefully** This is a key idea discussed throughout the book. Basically, make sure that, if you are going to push the limit of design, any new technologies degrade gracefully under older browsers. For example, if you are relying on JavaScript, does the page still work without it on? Or at least evor gracefully?
- **Ensure user control of time-sensitive content changes** Make sure that moving, blinking, scrolling, or autoupdating objects or pages may be paused or stopped by the user. Besides being highly annoying, such distractions may actually make it difficult for users to focus on the site.
- **Ensure direct accessibility of embedded user interfaces** If you use an interface within the page—for example, a Java applet that has its own internal interface—make sure that it, too, is accessible.
- **Design for device independence** Try to build interfaces that can work in multiple devices, including those with different screen sizes, different viewing devices (cell phones as well as computers), and different manipulation devices (keyboard or mouse and keyboard). A particularly important consideration is just making sure that a site doesn't rely solely on the mouse for navigation. Some users may find mouse movement difficult, and power users may actually prefer to use the keyboard for navigation.
- **Use interim solutions** Because not all browsers will support the same technologies or standards completely, make sure to provide alternatives in the short term for noncompliant browsers.
- **Use W3C technologies and guidelines** A somewhat self-evident but occasionally troublesome suggestion. Of course you should always try to follow the W3C guidelines, at least in spirit. However, be careful because many W3C guidelines are no more than proposed ideas, and browsers may lack significant or consistent support for a defined specification.
- **Provide context and orientation information** In some sense, this just means to explain things or provide instructions for complex areas. You should design pages so that the meaning of links is clear through the use of ToolTips or scope notes. Further, forms should be designed that explain what is required. In the most basic way, a site should provide a help system.

- **Provide clear navigation mechanisms** Basically, you should provide obvious navigation that is easy to understand and at a consistent location on the screen. Navigational aids such as search engines, site maps, and site indexes should also be provided.
- **Ensure that documents are clear and simple** Yet another fairly obvious suggestion, but powerful nonetheless, is that simplicity will lead to greater accessibility. Given that not everyone will be able to read a language well, and usability is directly related to simplicity and consistency, try to make your documents simple.

Besides manual inspection of a site, it is easy enough to evaluate it for accessibility using a tool such as Bobby (http://www.cast.org/bobby), as shown in Figure 2-7. Bobby


Welcome to Bobby WorldWide: CAST - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www.cast.org/bobby/
CAST Bobby
[Text version]
Site Tools:
Take Notes | Provide Feedback | Change Interface | Get Language Help
Navigator
Bobby
Customization
Pricing Options
About Bobby
Sponsorship
Approved Sites
Documentation
Support
Site Tools My Cast Site Map
Bobby Sponsor
IBM
Internet

Welcome to Bobby WorldWide

About Bobby | Terms of Use | Advanced Options | FAQ | Support | Approved Sites | Online Discussions

URL: http:// Submit

(●) Web Content Accessibility Guidelines 1.0
( ) U.S. Section 508 Guidelines

Enter the URL of the page that you want Bobby to examine and click Submit. This dialog tests only one page at a time. Bobby limits the number of pages it checks in order to keep the server available to all. To test an entire site as a batch, use the downloadable version of Bobby. If you have difficulty reading the report, try the Customization page using the default settings.

CAST offers Web accessibility seminars if you would like a further introduction to the accessibility issues Bobby examines.

Bobby was created by CAST to help Web page authors identify and repair significant barriers to access by individuals with disabilities. For directions on getting Bobby Approval see our Approval page. *Bobby WorldWide* is a new version of Bobby software that:

- Analyzes sites using the U.S. Federal Government's Section 508 guidelines
- Analyzes sites using the WAI Conformance A, AA, and AAA guidelines
- Tests pages that use Secure Socket Layers (SSL)
- Provides better detection of HTML files types in Frame Source

Figure 2-7. *Bobby can be used to check the accessibility of a Web site.*

will analyze a Web page and see if it meets certain basic accessibility criteria, such as the use of **alt** text.

Building a Usable Site

One of the keys to usable Web site development is to focus from the beginning on the users of the application. Remember that the user's goal is not to use computers or to use your Web site. The user's goal is to accomplish some task—purchase a product, find a bill payment center, register a complaint, and so on. You should try to make direct contact with users, and you must listen to them. Do not fall into the trap of thinking that you should just simply ask users what they want and then they will design your site for you. Users are not designers, and they make illogical or unrealistic requests. Because of this, you may be tempted to implement your own idea of a great site instead, without regard for user requests. However, the core idea of user-centered design is to always remember we are designing for users and not ourselves. Recall again the following very important Web design rules:

Rule: You are NOT the user.

Rule: Users are NOT designers.

Although not all user input will be valuable, you should solicit information from your intended audience. You might consider interviewing them or giving a survey. Whatever you do, make sure to let users talk—and listen to them. While this may seem like JAD (Joint Application Design), which will be discussed in Chapter 4, we will not let users control the project; rather, they will be used as a source of ideas and a way to verify the execution of implemented features. From interviews, you should build a profile of stereotypical types of users. While this may seem to be a bad idea, consider that unless you have a very small audience, it is virtually impossible to build a site that will conform perfectly to all the preferences and task requirements that all possible users might have. Even if it were possible, it would be prohibitively expensive.

From your discussions with users, build a prototype site, or just a set of simple diagrams on paper of how pages might look, and test it out with users. Make sure you test your site with users as early as possible in the development cycle so as not to build a site that users can't figure out.

Suggestion: Perform user testing early and often.

There are many ways to verify usability. Tests might include

- Casual observation of users
- Surveys and interviews
- Focus groups

- Lab testing
- Heuristic evaluations by developers or usability experts

The results of the tests can include more quantitative measurements, such as the number of mistakes made during a task, the amount of mouse travel, the time it takes to perform a task, and so on. Tests will certainly also have to include qualitative measures of what feature the users liked or didn't like. Before you don a white coat and rent lab time in a room with a two-way mirror to observe users, consider that formal testing may be overkill for most sites because of the cost and trouble of performing user tests in a formal fashion. Simple observations might do the trick, and opinions tend to be free from many users, though not always well founded. Collect a few users, or even your friends and neighbors, and sit them down at the site, and have them perform a few tasks. What's interesting is that even an informal test will uncover the major problems with a site. However, informal tests only work if you let them. Designers seem far too proud of their sites and tend to act as co-pilots, showing a user the interesting aspects of a site. Talking too much during a test or guiding the user in any way keeps the user from making his or her own decisions and may actually steer the user away from mistakes.

Suggestion: When performing even an informal usability test, avoid talking too much or guiding the user.

Before running off to round up your friends to ask them what they think, first consider that far too often users will tell you what you want to hear or what they *think* they would do in certain situations. Or they simply may not want to admit their misunderstandings. It is better to observe users' behavior than to rely on statements from them. However, if this is not possible, user input is acceptable, particularly if it is coupled with your own ad hoc usability analysis of a site. For instance, see whether the site follows the basic usability criteria that have been described in this chapter. Table 2-4 presents some guidelines you should use for judging a site.

When evaluating a site, the rules of thumb here cover the basic aspects of usability. However, don't assume just because the site meets most of these basic ideas that it is a good site. There are plenty of other ways for a site to fall down. For example, a site might not contain excellent content, its technology may be unreliable, or its graphics may be hideous to look at. Chapter 5 presents a more in-depth evaluation procedure that accounts for many other aspects of Web design. Remember that usability isn't the only part of a positive Web experience.

Guideline	Explanation
Be consistent	Consistency is the key to an easy-to-use interface. If something is consistent, the user only has to learn it once. Within your own site, don't change the position of buttons or the way things act.
Don't violate a user's expectations, and make sure to follow Web and GUI conventions	Consistency can go beyond the contents of a site. Users will have expectations about how things work shaped by visits to other sites. Make sure your site is consistent with what they expect. In short, follow any conventions used in GUI or site design that the user is familiar with.
Support the ways people use Web pages	Users use the Web pages in a few basic ways. They load a page, they unload a page, they print the page, they save the page (either by bookmarking the address or saving the file to a local drive), they read the page, or they interact with the page (such as by filling in forms or manipulating content objects within the page). As with the previous guideline, make sure users can do all the things they expect to be able to do. If users expect to print or bookmark a page and they can't, they may consider the site unusable.
Use surprise properly and sparingly	Occasionally, being inconsistent is useful. If you want to "wake a user up," it might be OK to dramatically change the way a page looks or acts. Just make sure you don't do this often, since users may never become comfortable with the site and may even become frustrated with the ever-changing interface.
Simplify the site and individual pages as much as possible	Simplicity makes it easy for users to understand a site. Try to pare a site or page down to its bare essentials. Look at statistical logs to determine what pages are not needed from a site. On a page level, remove clutter from layouts and try to reduce visual noise.

Table 2-4. *Common Web Usability Guidelines*

Guideline	Explanation
Rely on recognition, not recall	Memorization is difficult. Don't expect the user to memorize the structure of your site or the position of your buttons. Minimize what the user has to remember by exposing available choices. Even something as simple as hiding a menu when it isn't in use increases the cognitive load on a user, since they have to memorize what items are on what menus.
Do not assume users will read instructions	You may not get a chance to hold a training class for every user who visits your site. Generally, users will read help files only when they are in trouble. Make sure that they don't need to be trained. Avoid introducing features in a site that would require training or documentation for proper use.
Prevent or correct user errors	Don't let users make mistakes that are unnecessary. For example, validate form entries and limit users to doing only what they should. Don't provide a choice that is not easily undone by the user. If errors do occur, let the user know about the error and its possible solution.
Provide feedback	Let users know what's happening. Don't be imprecise with feedback. If there is going to be a delay, let them how long it is going to take. If an error has occurred, provide a clear error message.
Support different interaction styles	Try to provide multiple ways of doing the same thing to deal with different approaches to problems. For example, some users may prefer to use a site map rather than a search engine when looking for something. Don't limit users, but do account for a range of interaction styles, from novice users to power users.
Minimize mouse travel and keystrokes	Typing and moving the mouse around the screen is work for the user, so try to minimize it. This means successive button choices should be nearby. Try to minimize the distance from primary navigation to the Back button, which is certainly the most commonly pressed button in a browser. Navigation should probably be toward the top of the screen.

Table 2-4. *Common Web Usability Guidelines* (continued)

Guideline	Explanation
Consider medium of consumption.	Make sure to understand where the user will consume the content—on screen or on paper. If users print pages to consume them, shouldn't the usability test be performed on the paper document as well?
Consider environment of use.	If known, consider where a user will interact with a page. Where users interact with a page will affect how usable they perceive it. For example, relying on sound in a noisy environment isn't a wise idea.
Focus on speed.	Users dread slow-loading sites. Make sure pages are fast-loading by practicing minimal design. This doesn't mean eliminating graphics, only that a page should be no slower than it needs to be to deliver its message.

Table 2-4. *Common Web Usability Guidelines* (continued)

Usability Above All Else

One problem with usability discussions is that it is easy to use usability concerns as a way to squash any other reasonable value. For example, some people have gone so far as to discuss how banner ads contribute to poor site usability because they are animated or increase the download time. However, consider that without the banner ads the site may not be economically viable. Pleasing graphics also are a common target for usability experts. It is interesting to note how boring most usability gurus' sites actually are. While a site without much graphics may be usable, it might not do much to improve the brand identity of the organization running the site; in fact, without graphics, it may undermine brand identity built through other mediums. In some situations, it may be important to let the user endure a slightly longer download in order to see the corporate logo and new advertising look.

Advanced technology also is a common enemy of good site usability. The truth is that while advanced technology may lock out some users, what is provided may be worth it. If we always designed for the lowest common denominator, we'd still have text-only Web pages. Don't let usability completely stifle innovation. Usability is certainly very important, but there are often other considerations in a Web site's design. Always remember that while we design for users, we are ultimately in control of our site.

Suggestion: Do not use usability concerns as a way to avoid or eliminate visual, technological, or economic aspects of a site.

Who’s in Control of the Experience?

While it is true that we must give the people what they want, the masters of sites—meaning those who pay for them—may have desires that are not congruent with the desires of the site’s users. Do not become a slave to the user; remember that, in some sense, we are the masters of our own sites. How we want to treat our visitors is going to influence greatly how they feel about visiting our sites. Do you want to be a dictator, forcing the user to download certain plug-ins or resize a window? Conversely, you could be very democratic and let users pick their own path through your site. You may even allow users to modify content on the site or influence other users with indicators of link popularity. Last, you could aim for a middle ground and maybe act as a benevolent dictator, trying to help users along the way and giving them freedom within certain constraints, but always trying to guide them along.

The issue of control during a site visit is somewhat of an unwritten contract between the site user and the developer. There is give and take in the relationship. While one of the main tenets of user-centered design is to put the user in control, users are imperfect like everyone else; if we give them complete control, they may make serious errors. Developers will want to keep users from making mistakes. However, the role of the benevolent dictator of the online experience is difficult. If you control things too much and users notice that they can’t resize their window or press certain buttons, they may become angry or frustrated. The key is to provide an illusion of control.

Users should be able to do everything they need to do and nothing more. People need to feel like they are in control, but the control should have limits. Good interfaces exhibit this control. Consider, for example, the famous adventure game *Myst*™. In *Myst*, the user can click objects onscreen and move in a direction simply by clicking in the appropriate direction. The interface is very simple and also very restrictive, though game players rarely notice this. In *Myst*, as in many well-designed video games, the progression is very controlled by the game designer, but the illusion of control is always preserved. A great Web site would follow the cue of a video game by trying to guide someone to a conclusion like purchasing a product, but in a manner that the user doesn’t really notice.

The best example of the balance of control in an experience is probably Las Vegas. Casinos create a complete experience of visiting an ancient land, tropical paradise, or foreign country. A gimmick outside the casino like an exploding volcano or pirate battle attracts hordes of visitors. The intent is that some of these visitors will step onto the nearby conveyor belt to be quickly whisked into the casino. Inside the casino, temperature, lighting, and oxygen level are carefully controlled in an attempt to create a pleasant environment. The passage of time becomes difficult to determine because windows are few and tinted, and clocks are nonexistent. Assistance is plentiful from dealers and waitresses who will provide free drinks. If you get hungry, cheap food is nearby at an all-you-can-eat buffet. Want to stay overnight? Rooms are reasonably priced—and if you spend enough, they might even be free. But when you come to your senses as your wallet begins to empty, notice how difficult it is to find the exit! Good

Las Vegas casinos practice the ultimate in experience design, second only (maybe) to Disneyland. The experience is always controlled; the point is to maximize the money the casino takes in. If you step out of line, get irate and loud when you lose, or try to do something to win back control in gambling by card counting, you'll find that you are quickly escorted outside. The experience is fun and you can win, but the control is there and the house always has the edge. It's pure math. If you plan to run a commercial site, learn from Las Vegas.

Suggestion: Practice "Las Vegas" Web design. Provide the user with a pleasant experience, complete with perks and the illusion of unlimited choices, but control the situation strictly at all times.

Summary

Usability is about the aspects of a site that aren't always noticeable but yet seriously influence the ease in which a user is able to accomplish a task using the site. Usable sites should be easy to learn, easy to use, and easy to remember. They should also result in few errors and be satisfying to the user. While some ways to improve usability, such as consistency and simplicity of design, are easy to formulate, sometimes it is difficult to satisfy the needs of every user. One reason is that users have different Web skill levels—novice, intermediate, and advanced (power users)—that will affect site usability. Another is that, while users generally share certain capabilities for accessing a site, such as vision and memory, users are also individuals, with unique characteristics, opinions, and experiences. They will also tend to view your site as a mere island in a big ocean of sites, and it is best to assume that they won't want to learn your special rules.

With so many varieties of users, you probably won't be able to perfectly accommodate every user's unique tastes and requirements. However, if you create an adaptive interface that can be used by the three broad categories of users and make sure to test your site carefully with real users, you stand a good chance of making a site that is usable by most users. Be particularly careful not to lock users out, particularly those who may be disabled or slightly different from your average user.

Finally, a site should always be built to meet the needs of its users within the constraints or the desires of its creators. However, never use the quest for a usable site as a way to avoid difficult problems or as an excuse not to use graphics or technology or introduce new features that a user might want. An overzealous Web professional waving the usability banner can easily stifle innovation. Balance is always the key to great Web design.

This page intentionally left blank.

Chapter 3

The Web Medium

While the human element may be the most critical aspect of Web-based communication, effective Web design is also extremely dependent on correct technical execution. If a site is poorly constructed or error ridden, visitors may lose sight of its message or function. To excel at Web design, practitioners should have a complete understanding of the elements of the Web medium.

The Web medium is composed of three major components: client, server, and network. We will briefly overview each component and its subcomponents here in order to provide designers with a complete vocabulary of modern Web technology—and possibly provoke further study. We will also provide links about the activities of the various standards bodies, particularly the World Wide Web Consortium (W3C), which defines Web technologies, and the IETF, which sets many of the network, related protocols. Later chapters will focus on correct site execution and the effects of Web technologies on design decisions.

Core Web Technologies

As described in Chapter 1, the Web is implemented as a client-server system over a vast public network called the Internet. The three components of any client-server system are the client side, the server side, and the network. A visualization of the basic components that make up the Web is shown in Figure 3-1. We will now survey each of the primary components in turn, starting with the client side, which is primarily defined by the browser.

Web Browsers

The Web browser is the interpreter of our Web sites. It is very important to understand the Web browser being supported and what capabilities it has. The two most common browsers at the time of this book's publication are Microsoft's Internet Explorer (which accounts for the majority of browser users) and Netscape's Communicator (Navigator). While these two browsers account for most users accessing public Web sites, there are numerous other versions of browsers in use.

Note *The exact figures for browser usage at public Web sites are continually changing and are tracked by various statistics sites as well as browser-related sites such as http://www.upsdell.com/BrowserNews/.*

The problem with published browser usage reports is that they don't necessarily reflect your browsing audience. Consider a site that publishes Macintosh software—its browser usage pattern might actually show a fair number of users with OmniWeb, a Macintosh-specific browser that has a notable number of rabid followers. However, most sites probably wouldn't consider OmniWeb something to even think about. Depending on your users, the types of browsers will vary. From statistics showing that surveyed

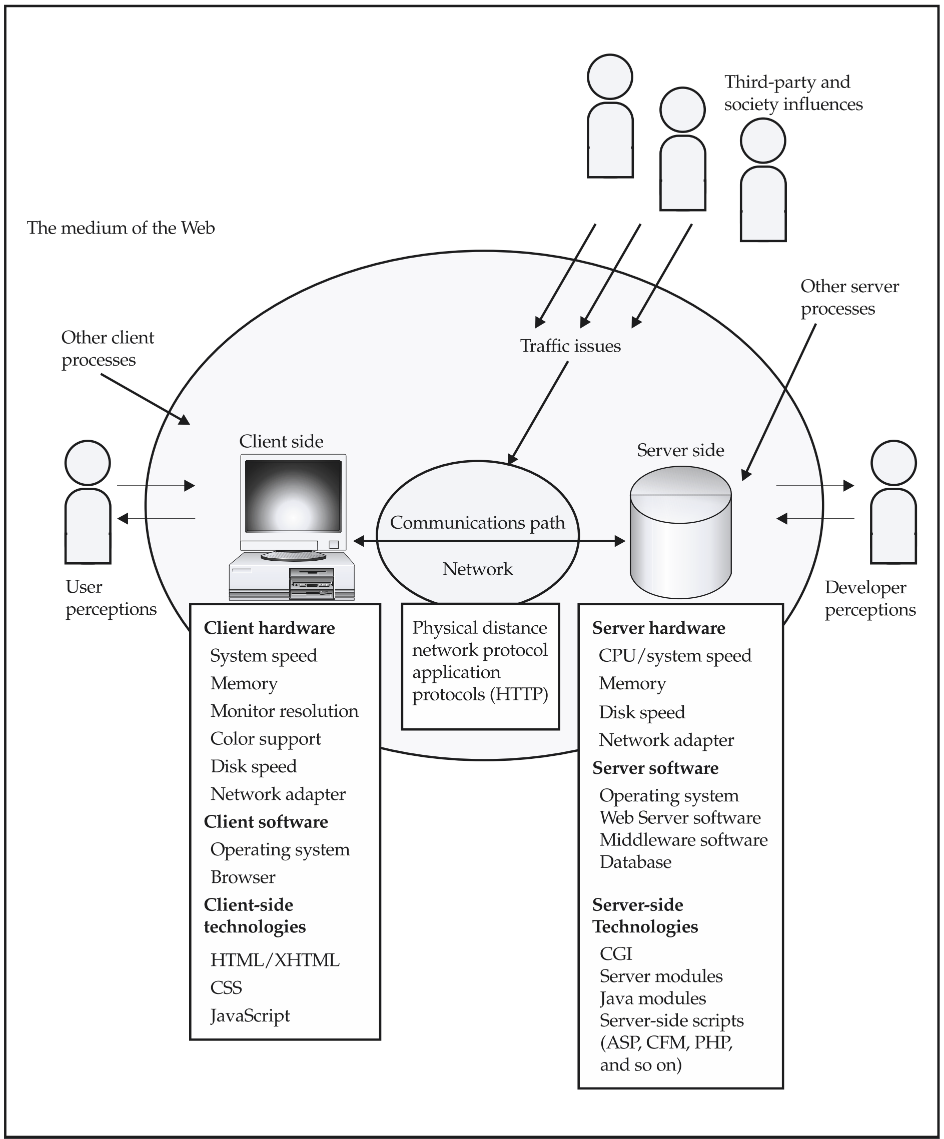


Figure 3-1. *Components of the Web medium*

sites favor a particular browser, it does not necessarily follow that your site will exhibit the same browser usage patterns—though it is pretty likely. Look at your own log files to determine browser usage patterns. If you are building an intranet site, you might not even have to look at your logs to understand what browsers are in use.

Rule: Beware of relying on published browser usage figures; track actual browser usage on *your* site.

Given a mix of browsers made up of the top two vendors with a smattering of other browsers, the question becomes how this information relates to site design and technology use. One possibility is to look at the various browsers and their capabilities, and then design for some common set of features. First, look at the browsers listed in Table 3-1.

Considering the variations among browsers, the common ground isn't terribly advanced. The safest design platform for some still seems to be what Netscape 3.*x* supports, though more and more designers are embracing design for the 4.*x* and 5.*x* generation browsers and using CSS, Flash, and JavaScript more often.

The only problem with moving to the next generation is that the gap between what different generations of browsers support can be rather large. Because of this, sites (and users) significantly favor Internet Explorer over Netscape. (The installed base for IE browsers includes between 85% to 90% of all users at the time of this writing.) With the advent of Netscape's Mozilla-based browsers (Netscape 6 and 7, and Mozilla 1.0), things may get more interesting because these browsers promise more support for standards-based Web page development than Netscape's 4.*x* generation browsers. Even so, there will not be an overnight adoption of new, non-IE browsers around the Web. As the installed base increases, the longer it will take for consumers to embrace new technologies. Therefore, public sites should consider developing for at least one, if not two, generations prior to the current release of a browser. Even more than six years after the release of the 2.*x* generation browsers, some public sites still support that generation of browsers perfectly.

Tip: Consider developing for at least the last two, if not three, versions of a browser to account for slow upgrades.

It is easy to be overwhelmed with potential browser considerations, even if dealing just with the major browsers' most recent versions. At the time of this writing, there were more than 20 major versions of the 4.*x* generation alone and more than 400 other different potential Netscape variations—primarily older versions or beta releases—floating around the Web, all with different capabilities and bugs. Of course, Netscape isn't the only browser vendor, and there are slight upgrades made to Internet Explorer as well. The only point to make here is that browsers are moving targets. Every release has new features and different bugs. Just because someone is using a 4.*x* generation browser doesn't guarantee a site will work the same under the same version on another platform or under an interim release. Sorry, but Netscape 4 or Internet Explorer 4 on Windows won't work the same on Macintosh and NT. Even different interim releases like 4.03 and 4.5 may have significant differences in page rendering and bugs. Add in

Browser	Version	HTML Version Support	JavaScript	CSS	Programming	Comments
Internet Explorer	3	HTML 3.2 + extras	JavaScript 1.0	Somewhat	Helper apps, ActiveX controls, Netscape plug-in compatibility, Java, VBScript	Some corporate and slow upgraders still use this version. Good possibility to target this browser for a fall-back version of a site.
Internet Explorer	4	HTML 4 + extras	JavaScript 1.1 + extras. Note that the browser supports more advanced JavaScript features, but its **language** attribute support indicates 1.1 as the maximum supported JavaScript.	Partial CSS1	Helper apps, ActiveX controls, Netscape plug-in compatibility, Java, VBScript	IE 4's main advances were in CSS support and improved JavaScript. IE 4 was the first browser to support pages that could be significantly manipulated after page-load using JavaScript and relying on the Document Object Model (DOM).
Internet Explorer	5	HTML 4 + extras	JavaScript 1.2 + extras. Note that the browser supports more advanced JavaScript features, but its **language** attribute support indicates 1.2 as the maximum supported JavaScript.	Most of CSS1 + some extensions	Helper apps, ActiveX controls, Netscape plug-in compatibility, Java, VBScript	IE 5 mostly refines the features provided in HTML 4, though it does begin the use of client-side XML.

Table 3-1. *Common Browser Versions and Characteristics*

Browser	Version	HTML Version Support	JavaScript	CSS	Programming	Comments
Internet Explorer	5.5	HTML 4 + extras	JavaScript 1.2 + extras. The same conformance issue as previous browsers hold; IE 5.5 supports more advanced JavaScript but may not report it properly using some scripting techniques.	Most of CSS1 + some extensions	Helper apps, ActiveX controls, Netscape plug-in compatibility, Java, VBScript	IE 5.5 continues to refine the basic ideas presented in IE 4 and 5 with enhancements to style sheet support and XML.
Internet Explorer	6.0	HTML 4 + extras, XHTML 1.0	JavaScript 1.3 + extras. Note that the browser supports more advanced JavaScript features but its **language** attribute support indicates 1.3 as the maximum supported JavaScript.	CSS1 + extensions	Helper apps, ActiveX controls, Netscape plug-in compatibility, and VBScript. Note that IE 6 does not initially ship with Java, though many users add it.	IE 6 continues to refine the basic ideas presented in IE 4 and 5.*x* with enhancements to standards support and XML. DOM Level 1 compliance is close to full.
Netscape	1.*x*	HTML 2 + extras	No	No	Helper apps	No frames (good example of a worst-case graphical browser). Generally not considered in design decisions.
Netscape	2.*x*	HTML 2 + extras	JavaScript 1.0	No	Helper apps, plug-ins, Java	No background color on table cells, Java implementation buggy, JavaScript limited to simple form validation. Rarely considered in design decisions.

Table 3-1. *Common Browser Versions and Characteristics* (continued)

Browser	Version	HTML Version Support	JavaScript	CSS	Programming	Comments
Netscape	3.*x*	HTML 3.2	JavaScript 1.1	No	Helper apps, plug-ins, Java	Rollover buttons become possible, Java more stable. Occasionally considered as a fall-back browser in design decisions.
Netscape	4.*x*	HTML 4 + extras	JavaScript 1.2, 1.3	Some CSS1 + CSS-P	Helper apps, plug-ins, Java	Limited DHTML support, primarily object movement and visibility; CSS support buggy, suppressed JavaScript error messages, automatic installation of plug-ins introduced. Still considered by some designers because of slow Netscape upgrades.
Mozilla/ Netscape	Netscape 6.*x*/ Mozilla 1.*x*	HTML 4.01, XHTML 1.0	JavaScript 1.5	Full CSS1, partial CSS2	Helper apps, plug-ins, Java	Great standards support for XHTML, CSS, XML, PNG, and other W3C approved standards. Not widely used at time of this edition's publication.

Table 3-1. *Common Browser Versions and Characteristics* (continued)

the continual use of half-done beta browsers, and you have a recipe for disaster. Pages often won't render correctly, and errors will ensue. Users unfortunately won't always place blame correctly. A small layout problem may be interpreted as the designer screwing up, not the browser vendor releasing a poorly tested product.

Rule: Users often don't blame browsers for simple errors—they blame sites.

So what's a developer to do? First, make sure you know what's going on. Keep up with the latest news in browsers at sites like http://www.upsdell.com/BrowserNews/. In particular, watch out for beta and interim releases. They are often the most dangerous, and users will not consider a 6.1 and 6.2 to be significantly different.

Tip: Be careful of features in beta and interim releases of browsers.

The next thing to consider is exactly what browsers you need to be aware of. This requires that you know the browsers used by the site's audience, so look to your log files. In general, public sites should be as browser agnostic as possible, while private sites like intranets may be designed specifically for a single browser. Designers should

Browser	URL	Comments
Internet Explorer	http://www.microsoft.com/ie	Consider having the last three versions of this popular browser. Note that this may require having multiple systems or boot options to run numerous versions of IE.
Netscape	http://browsers.netscape.com/browsers	With so many versions available, consider using the last version of each major release: 6.2, 6.1, 6.0, 4.7, 4.6, 4.5, 4.0*x*, 3.*x*, and 2.*x*
Mozilla	http://www.mozilla.org	The browser behind Netscape's project to build a 6.*x* generation browser should always be followed as a preview to what's coming soon and to test Web standards.

Table 3-2. *Useful Browsers for Testing Purposes*

Browser	URL	Comments
Opera	http://www.opera.com	This fast, standards-aware browser is becoming very popular and may be a strong third choice for some users.
America Online	http://webmaster.info.aol.com/	Not a Web browser per se, but the use of Web browsers under AOL and associated AOL TV is often very troublesome. Developers should look at public sites under AOL very carefully.
Lynx	http://lynx.browser.org	It is useful to test with Lynx, a text-only browser, to understand how a page renders without any graphics.
Amaya	http://www.w3.org/Amaya/	Not a realistic browser for users, but the W3C's test browser often implements interesting standards-related features before commercial browsers. Useful for experimenting with specifications. Avoid for realistic testing.

Table 3-2. *Useful Browsers for Testing Purposes* (continued)

be aware of the browser families listed in Table 3-2. Users interested in development for non-PC platforms may also find Palm (http://www.palmos.com/dev/), television (http://www.developers.aoltv.com/ and http://developer.msntv.com/), and cell phone simulators (http://developer.openwave.com/) very useful tools for testing sites.

Tip: Beyond the leading browsers, consider testing with standards-oriented browsers as well as text-only or alternative-environment-access browsers.

Given the number of browsers available and the significant difficulties involved in testing dozens of different configurations just to ensure a site renders under common viewing environments, some authors decide to write for a particular browser version or indicate that a particular vendor's browser is the preferred viewing platform. Many sites that do this exhibit a browser badge on the site. If a particular browser is required, do not blatantly advertise it on the home page as many sites do. It simply announces that you practice exclusionary development.

Tip: Do not advertise favored browsers blatantly on a home page.

Markup Languages

The foundation of any Web page is markup. Markup technologies such as HTML, XHTML, and XML define the structure and possible meaning of page content. Despite the common belief that markup languages define the look of Web pages, and the equally common use of HTML in this manner, page appearance should really be accomplished using other technologies, particularly style sheets.

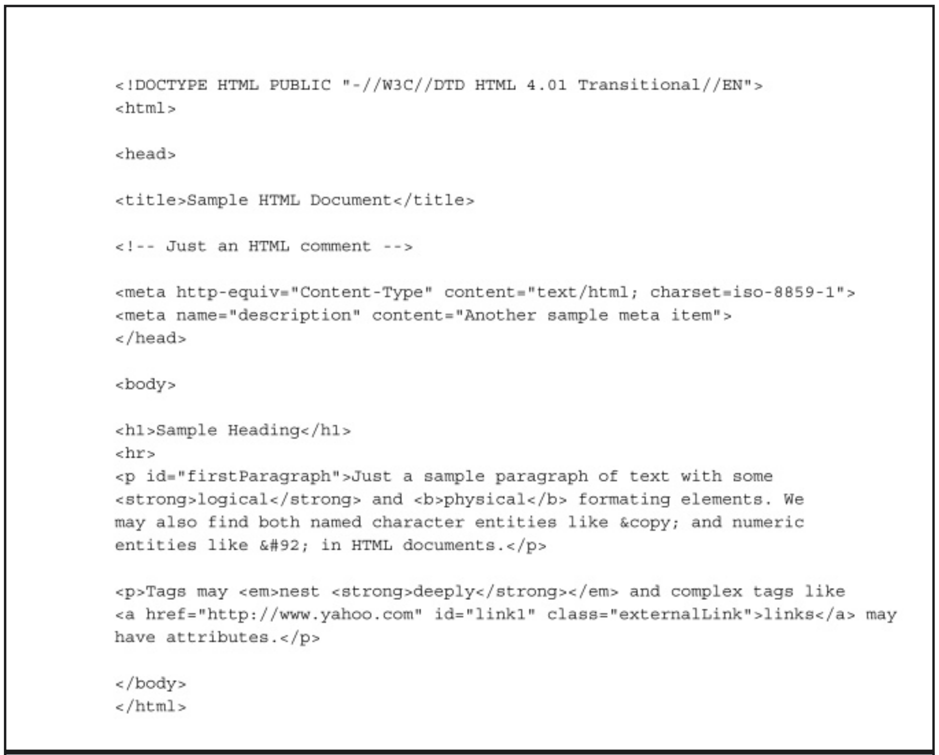
HTML

HTML (HyperText Markup Language) is the primary markup technology used in Web pages. Traditional HTML is defined by a SGML (Standardized General Markup Language) DTD (Document Type Definition—see the upcoming section "XML") and comes in three primary versions (HTML 2, HTML 3.2, and HTML 4). HTML 4 comes in three varieties: transitional, strict, and frameset, with most document authors using the transitional variant. HTML 4.01 is the most current and final version of HTML. An example of an HTML document showing common structures is presented in Figure 3-2.

While the various tags and rules of HTML are fairly well defined, most browser vendors provide extensions to the language beyond the W3C definition. Further, the browsers themselves do little to enforce the markup language rules, leading to sloppy usage of the technology. Also, while HTML should be used primarily for structuring a document, many developers use it to format the document for display as well. HTML's formatting duties should eventually be completely supplanted by Cascading Style Sheets (CSS). However, even with adequate style sheet support in browsers, many developers continue to use HTML tables and even proprietary HTML tags in their page design. There are no plans for further development of HTML by the W3C and browser vendors, and developers are encouraged to embrace XHTML.

The HTML 4.0 specification is available at the following URL:

- http://www.w3.org/TR/html401/



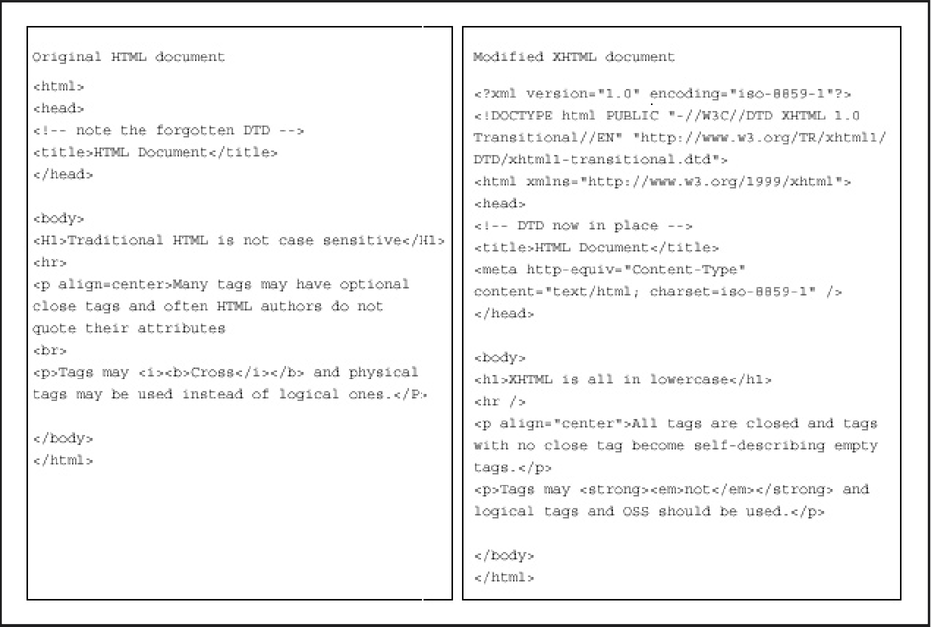
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>

<head>

<title>Sample HTML Document</title>

<!-- Just an HTML comment -->

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="description" content="Another sample meta item">
</head>

<body>

<h1>Sample Heading</h1>
<hr>
<p id="firstParagraph">Just a sample paragraph of text with some
<strong>logical</strong> and <b>physical</b> formating elements. We
may also find both named character entities like &copy; and numeric
entities like &#92; in HTML documents.</p>

<p>Tags may <em>nest <strong>deeply</strong></em> and complex tags like
<a href="http://www.yahoo.com" id="link1" class="externalLink">links</a> may
have attributes.</p>

</body>
</html>
```

Figure 3-2. *Sample Document with common HTML structures*

XHTML

XHTML is a reformulation of HTML using XML (extensible Markup Language) rather than SGML. XHTML solves two primary problems with HTML. First, XHTML continues to force designers to separate the look of the document from its structure, by putting more emphasis on the use of style sheets. Second, XHTML brings much stricter enforcement of markup rules to Web pages. For example, XHTML documents must contain only lowercase tags, always have quotes on attributes, and basically follow all the rules as defined in the specification. Figure 3-3 shows an example document in HTML and its equivalent in XHTML.

A rigorous discussion of HTML and XHTML that covers all the requirements of XHTML can be found in Appendix C as well as in the companion book, *HTML: The Complete Reference* (www.htmlref.com).



```
Original HTML document
<html>
<head>
<!-- note the forgotten DTD -->
<title>HTML Document</title>
</head>

<body>
<H1>Traditional HTML is not case sensitive</H1>
<hr>
<p align=center>Many tags may have optional
close tags and often HTML authors do not
quote their attributes
<br>
<p>Tags may <i><b>Cross</i></b> and physical
tags may be used instead of logical ones.</P>

</body>
</html>
```

```
Modified XHTML document
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!-- DTD now in place -->
<title>HTML Document</title>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
</head>

<body>
<h1>XHTML is all in lowercase</h1>
<hr />
<p align="center">All tags are closed and tags
with no close tag become self-describing empty
tags.</p>
<p>Tags may <strong><em>not</em></strong> and
logical tags and CSS should be used.</p>

</body>
</html>
```

Figure 3-3. *Moving from HTML to XHTML requires syntactical strictness*

XHTML's syntactical strictness is both its biggest benefit and biggest weakness. Well-formed pages may be easier to manipulate and exchange by a program but are harder to create for a human. Uptake of XHTML has been slow because of this strictness. XHTML's extra rigor makes it less accessible than HTML, which is much more forgiving to beginners. So, until more tools that generate correct XHTML become available, the language will probably continue its slow uptake in the Web community at large.

The following URLs provide important information about XHTML:

- XHTML 1.0 Specification: http://www.w3.org/TR/xhtml1/
- XHTML Basic Specification: http://www.w3.org/TR/xhtml-basic/
- XHTML 1.1 Module XHTML: http://www.w3.org/TR/xhtml11/

XML

Extensible Markup Language (XML) is being touted by many as a revolutionary markup technology that will change the face of the Web. Yet, despite the hype, few understand exactly what XML actually is. In short, XML is a form of SGML modified for the Web;

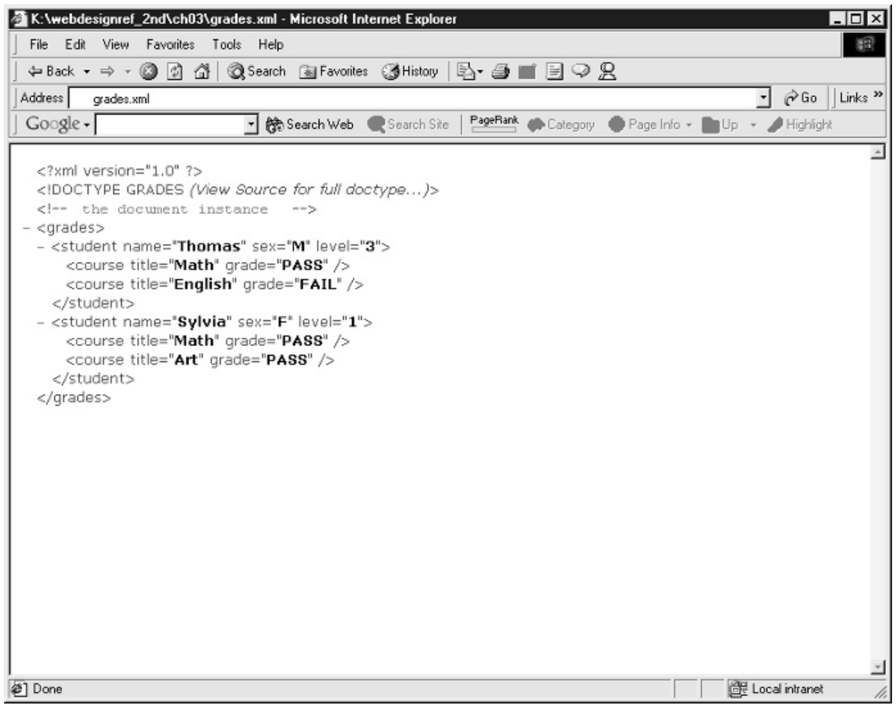
thus, it allows developers to define their own markup language. So, if you want to invent YML (Your Markup Language) with XML, you can. To do this we would define the rules of our invented language by writing a *document type definition*, or *DTD*. A DTD defines how a language can be used by indicating what elements can contain what other elements, the values of attributes, and so on. A simple DTD to define a grading language for elementary school children is defined here:

```
<!--Grades DTD-->
<!ELEMENT  grades  (student+)>
<!ELEMENT  student (course+)>
<!ATTLIST  student  name  CDATA  #REQUIRED
           sex  (M|F)  #REQUIRED
           level  (1|2|3|4|5|6) #REQUIRED>

<!ELEMENT   course EMPTY>
<!ATTLIST  course title  CDATA  #REQUIRED
           grade  (PASS|FAIL) #REQUIRED>
```

This DTD file named grades.dtd would be referenced by an XML file such as the one shown here:

```
<?xml version="1.0"?>
<!DOCTYPE GRADES SYSTEM "grades.dtd">
<!-- the document instance -->
<grades>
<student name="Thomas" sex="M" level="3">
   <course title="Math" grade="PASS" />
   <course title="English"  grade="FAIL" />
</student>

<student name="Sylvia" sex="F"  level="1">
   <course title="Math" grade="PASS" />
   <course title="Art" grade="PASS" />
</student>
</grades>
```

The example would not only be syntactically checked, but we could check the validity of the document against the DTD, a process known as *validation*. Yet, regardless of correctness, without a defined presentation you will not see much of a result, as shown in Figure 3-4. Presentation will eventually be handled by applying style rules to the XML document using one of the technologies discussed in the next section.


K:\webdesignref_2nd\ch03\grades.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address grades.xml
Go Links
Google Search Web Search Site PageRank Category Page Info Up Highlight
Done
Local intranet

```
  <?xml version="1.0" ?>
  <!DOCTYPE GRADES (View Source for full doctype...)>
  <!-- the document instance -->
- <grades>
  - <student name="Thomas" sex="M" level="3">
      <course title="Math" grade="PASS" />
      <course title="English" grade="FAIL" />
    </student>
  - <student name="Sylvia" sex="F" level="1">
      <course title="Math" grade="PASS" />
      <course title="Art" grade="PASS" />
    </student>
  </grades>
```

Figure 3-4. *Rendering of XML example in Internet Explorer 5*

Many readers may now be wondering about the value of developers defining their own individual markup languages. Why not just use XHTML or HTML? Wouldn't inventing new languages be the equivalent of creating a markup Tower of Babel on the Internet? Maybe, or it just may enable a whole new range of possibilities for markup. So far, the negative impact of inventing too many custom XML-based languages has been limited, and most Web developers are content using a commonly defined language like XHTML, WML (Wireless Markup Language), SVG (Scalable Vector Graphics), and numerous other XML-based languages. The precision and self-description properties of XML documents should enable a new class of Web technologies called *Web Services* that really could change the Web by allowing sites and programs to talk with each other more easily.

The XML Specification can be found online at http://www.w3.org/TR/REC-xml.

Style Sheet Technologies

Markup languages like HTML do not excel at presentation. This is not a shortcoming of the technology, but simply that HTML was not designed for this task. In reality, the look of the page should be controlled by the design elements provided by CSS (Cascading Style Sheets). In some cases, particularly when using an XML language, markup transformation may also be required to create the appropriate presentation format, so XSL (eXtensible Style Language) will be used as well.

CSS

CSS (Cascading Style Sheets) is used to specify the look of a Web page. This technology has been present at least partially in browsers as old as Internet Explorer 3.0, but it has long been overlooked in favor of HTML-based layout for a variety of reasons, including lack of consistent browser and tool support, as well as simple developer ignorance. With the rise of the 6.*x* generations of browsers, CSS is finally becoming a viable prospect for page layout.

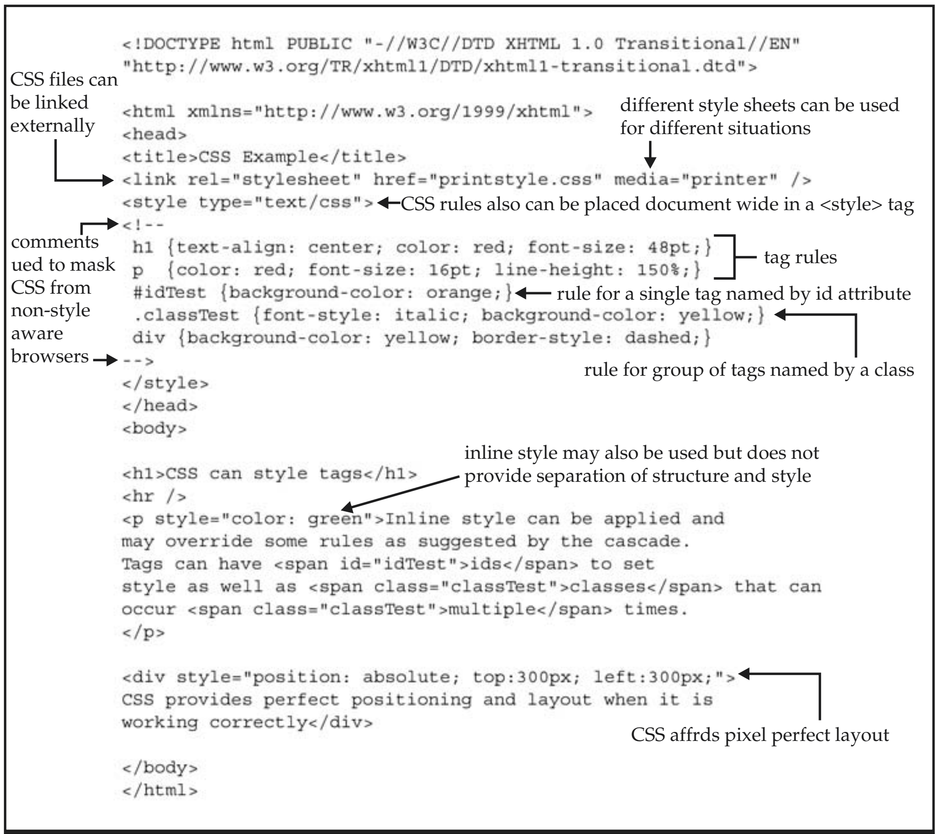
CSS-based style sheets specify rules that define the presentation of a type of a type (for example, **<h1>**)—a group or, more correctly, class of tags—or a single tag as indicated by its **id** attribute. Style sheet rules can be used to define a variety of visual aspects of page objects, including color, size, and position. The various style rules can be combined depending on tag usage—thus the "cascading" moniker for the technology. An example of CSS in use is shown in Figure 3-5.

These URLs provide more information about CSS:

- CSS1 Specification: http://www.w3.org/TR/REC-CSS1/
- CSS2 Specification: http://www.w3.org/TR/REC-CSS2/

XSL

XSL is another style sheet technology used on the Web. It is primarily used to style XML languages. This is usually accomplished through XSL Transformation (XSLT), which is often used to convert XML markup into other markup, often XHTML or HTML plus CSS. It is possible to also use XSL Formatting Objects to style content, but, so far, this does not seem to be a commonly employed aspect of XSL. Thus, when developers



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>CSS Example</title>
<link rel="stylesheet" href="printstyle.css" media="printer" />
<style type="text/css">
<!--
 h1 {text-align: center; color: red; font-size: 48pt;}
 p  {color: red; font-size: 16pt; line-height: 150%;}
 #idTest {background-color: orange;}
 .classTest {font-style: italic; background-color: yellow;}
 div {background-color: yellow; border-style: dashed;}
-->
</style>
</head>
<body>

<h1>CSS can style tags</h1>
<hr />
<p style="color: green">Inline style can be applied and
may override some rules as suggested by the cascade.
Tags can have <span id="idTest">ids</span> to set
style as well as <span class="classTest">classes</span> that can
occur <span class="classTest">multiple</span> times.
</p>

<div style="position: absolute; top:300px; left:300px;">
CSS provides perfect positioning and layout when it is
working correctly</div>

</body>
</html>
```

CSS files can be linked externally
different style sheets can be used for different situations
CSS rules also can be placed document wide in a <style> tag
comments ued to mask CSS from non-style aware browsers
tag rules
rule for a single tag named by id attribute
rule for group of tags named by a class
inline style may also be used but does not provide separation of structure and style
CSS affrds pixel perfect layout

Figure 3-5. *An example of CSS*

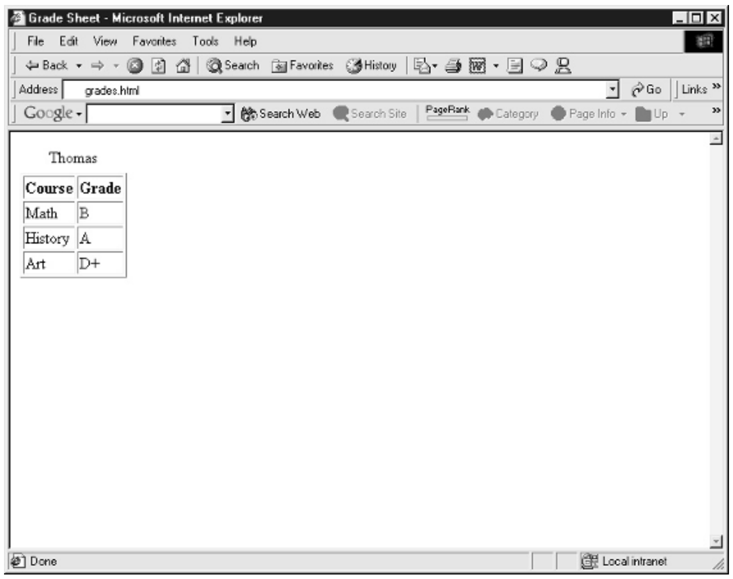
speak of XSL, they often are speaking of XSLT. An example of XSL Transformation is shown in Figure 3-6.

The relationship is set on the second line in the grades.xml file. The grades.xsl file specifies the transformations that would result in the HTML output as shown in Figure 3-7.

Note *Generally, the XSL transformation occurs on the server side, but XSL may become more prevalent on the client side as browsers continue to advance.*

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="grades.xsl"?>
<grades>

<student>Thomas</student>

 <course>
  <title>Math</title>
  <grade>B</grade>
 </course>

 <course>
  <title>History</title>
  <grade>A</grade>
 </course>


 <course>
  <title>Art</title>
  <grade>D+</grade>
 </course>

</grades>
```

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/
TR/WD-xsl">
<xsl:template match="/">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
<head>
<title>Grade Sheet</title>
</head>
<body>
<table border="1">
<caption><xsl:value-of select="grades/student"/>
</caption>
<tr>
  <th>Course</th>
  <th>Grade</th>
</tr>

<xsl:for-each select="grades/course">
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="grade"/></td>
  </tr>
</xsl:for-each>

</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```


Grade Sheet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address grades.html Go Links
Search Web Search Site PageRank Category Page Info Up
Done
Local intranet

Thomas

Course	Grade
Math	B
History	A
Art	D+

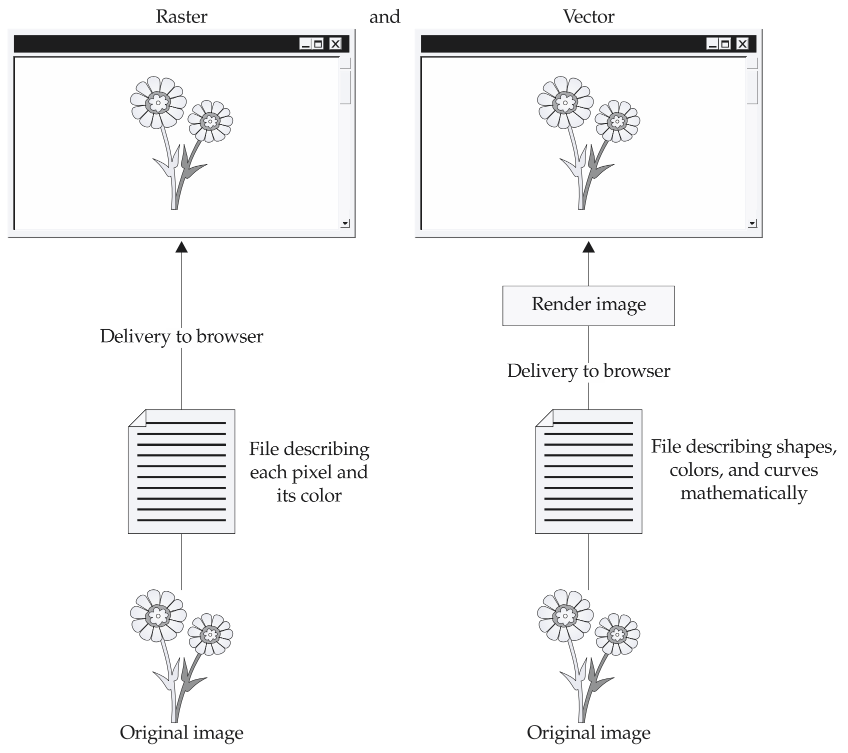
Figure 3-6. *XSLT in action*

Information about XSL can be found at these URLs:

- XSL Transformations 1.0 Specification: http://www.w3.org/TR/xslt
- XSL Activity at W3C: http://www.w3.org/Style/XSL/

Images

Most Web browsers support either directly or through extension a variety of image formats, such as GIF, JPEG, Flash, and PNG. The image formats can be separated into two general categories: *bitmap* (or *raster*) images and *vector* images. Raster images describe each individual pixel and its color, while vector images describe an image generally as a collection of mathematical directions used to draw—or more precisely, *render*—the image. Regardless of storage format, all images become bitmaps onscreen. The fundamental difference between the two general image formats is shown here:


Raster
and
Vector
Render image
Delivery to browser
Delivery to browser
File describing each pixel and its color
File describing shapes, colors, and curves mathematically
Original image
Original image

Some designers speak of the value of one general format over the other, but, in reality, both have their problems. Vector images tend to be compact in description and can be scaled mathematically, but they suffer in potential rendering time and realism. Bitmap images can be very detailed but do not scale up well and tend to be very large in terms of file size. We will examine the specific types of the images in the following sections. A complete discussion of their usage is presented in Chapter 14.

GIF

GIF (Graphics Interchange Format) is a bitmap format that does not provide a great degree of compression or color support, being limited to 8-bit or 256 simultaneous colors. However, the GIF format is relatively versatile and supports transparency, animation, and interlacing. It is commonly used in Web pages for logos, graphical navigation elements, and photos that do not require high-quality reproduction.

Information about the GIF Specification can be found at this URL:

- http://www.w3.org/Graphics/GIF/spec-gif89a.txt

JPEG

JPEG (Joint Photographic Experts Group) images support up to 24-bit color and are well suited for reproduction of photographs. Despite being a raster format, JPEG images allow designers to balance file size with image quality and support an impressive lossy compression algorithm that can significantly shrink image size with little discernable quality loss to the casual viewer. JPEG images do support progressive loading, but are not quite as versatile as GIF images because they lack transparency and animation features.

Information about JPEGs can be found at these URLs:

- JPEG Activity at the W3C: http://www.w3.org/Graphics/JPEG/
- JPEG Specification: http://www.jpeg.org/

Note *The JPEG 2000 standard aims to eliminate many of the problems with JPEG and provide an even greater degree of quality and compression than standard JPEG files. However, so far, JPEG 2000 is not available in Web browsers.*

PNG

PNG (Portable Network Graphics) images provide an advanced image format designed to replace GIF as the dominant form of graphics on the Web. PNG images provide three primary advantages over GIF: alpha transparency, which provides variable degrees of transparency (versus GIF, which has a single degree of transparency); gamma correction to help improve image brightness across systems; and improved interlacing and compression. While PNG provides numerous benefits, many of its advanced features are not properly implemented in the latest browsers, so the rush to embrace the format has yet to materialize.

Information about PNG can be found at these URLs:

- PNG Activity at the W3C: http://www.w3.org/Graphics/PNG/
- PNG Resources and Specifications: http://www.libpng.org/pub/png/

Flash

Macromedia's Flash is a vector image format that supports still images, animations, and complex interactivity using a built-in scripting language similar to JavaScript, called ActionScript. The format, defined in the form of an SWF file, is arguably the most popular multimedia format on the Web. It is used for implementation of navigation systems, animations, and presentations, as well as full-blown Web sites. The biggest complaint made about the format is that it is proprietary; thus, Macromedia has opened the format to the public, though it is not blessed by the W3C (which backs a rival standard called SVG). It could be further said that Flash, which was first popularized as an alternative to Macromedia's complex and sometimes clunky CD-ROM development environment Director, has become amost exactly what it sought to augment.

Information about Flash can be found at these URLs:

- Macromedia's Flash Homepage: http://www.flash.com
- SWF File Format Page: http://www.openswf.org

SVG

SVG (Scalable Vector Graphics) is an XML language for describing simple two-dimensional images. Because the language is XML based, scripting interaction is straightforward using standard JavaScript in conjunction with the Document Object Model. While the SVG format is an open standard, it has been slow to be adopted by the Web development community and will be unlikely to overtake Flash in the near term.

Information about SVG can be found at these URLs:

- SVG Activity at the W3C: http://www.w3.org/Graphics/SVG
- SVG 1.0 Specification: http://www.w3.org/TR/SVG/

VML

VML (Vector Markup Language) is yet another vector image used in Web pages. It is relatively unnoticed by most Web developers, despite the fact that it has been natively supported in Microsoft Internet Explorer since the 5.0 version. It was briefly introduced to the W3C for standardization, but SVG is being pushed over VML, and Flash is currently the popular vector format for the masses. However, Microsoft-oriented developers should be well aware of this format, since it is found in pages exported from Microsoft products.

Information about VML can be found at these URLs:

- W3C VML Note: http://www.w3.org/TR/NOTE-VML
- Microsoft VML Info:
  http://msdn.microsoft.com/library/default.asp?url=/workshop/author/vml/

Other Image Formats

The previously discussed image formats are the primary standard for well-supported image formats on the Web. However, other images are supported in some browsers, and, in theory, the **<img>** tag does not discriminate among the type, of images included in a Web page. The most important other format is probably BMP, which is supported by Microsoft's Internet Explorer. A variant called Wireless BMP (WBMP) is also noteworthy and is supported in some wireless browsers. Many browsers, particularly older browsers or those with a UNIX release, support Xbitmaps. Using plug-ins or helper applications, everything from PostScript files to TIFFs can be viewed in a browser.

Animation

A little animation can spice up a Web page a great deal. Animation on the Web is used for many things: active logos, animated icons, demonstrations, and short cartoons. There are a variety of animation technologies available to Web designers. Some of the most common animation approaches include animated GIFs, Flash and Shockwave, and JavaScript animations (also called DHTML). Other animation possibilities also exist: Java-based animations and older animation techniques such as "server push" are still possible. However, the field has narrowed significantly, and very few older or propriety animation formats are actually worth exploring. Table 3-3 details the animation choices commonly used and provides some facts about each.

Sound

Audio technologies on the Internet cover a lot of ground, from traditional download-and-play systems in a variety of formats such as WAV and MP3 to *streaming audio,* which attempts to play data as it is downloaded over a connection. Surprisingly, the most advanced technologies, and the most popular, may not be the best solution for Web sites. For example, MP3 files, while of high quality, tend to take too long to download, and streaming technologies might not provide reliable playback in all situations because of the unpredictable delivery conditions on the Internet. Fortunately, much has improved since the simple days of adding a WAV or MIDI file for background music, but there is still a long way to go before sounds will become commonplace, primarily because of the large size of audio files.

Audio files can be compressed to reduce the amount of data being sent. The software on the serving side compresses the data, which is decompressed and played back on

Animation Technology	Comments
Animated GIFs	Animated GIFs (GIF89a) are the simplest form of animation and are supported natively by most browsers. Looping and minimal timing information can be set in an animated GIF, but complex animation is beyond this format's capabilities.
JavaScript/DHTML	JavaScript can be used to move objects around the screen. This type of use of JavaScript is often described as dynamic HTML, or DHTML. Regardless of the name, this form of animation tends to be choppy and is not suggested for anything beyond simple button rollover and scrolling text effects.
Flash	Macromedia Flash, introduced earlier in the chapter, is the leading format for sophisticated Web, based animations. Flash files are very compact, and most Web users have Flash preinstalled on their system. Flash supports a growing programming facility based upon JavaScript.
Shockwave	Shockwave files are compressed Macromedia Director files. Their main benefit over Flash is simply that they support complex scripting. However, with the growing features of Flash, Shockwave files are falling quickly out of favor.

Table 3-3. *Common Web Animation Choices*

the receiving end. The compression/decompression software is known together as a *codec*. Just like image formats, audio compression methods are either lossy or lossless. Typically, audio codecs are lossy because of size considerations. Common audio delivery approaches for Web pages are shown in Table 3-4.

Video

The holy grail of Internet multimedia is certainly high-quality, 30-frames-per-second, real-time video. The main challenge to delivering video over the Internet is its extreme size. Digital video is measured by the number of frames per second of video and by the size and resolution of these frames. A 640 × 480 image with 24 bits color and a frame

File Format	Description
WAV	Waveform (or simply *wave*) files is the most common sound format on Windows platforms. WAVs may also be played on Macs and other systems with player software.
MPEG (MP3)	*Motion Pictures Experts Group* format is a standard format that has significant compression capabilities. MPEG Level 3 or MP3 files are very commonly used for distribution of music on the Web. However, due to their size, MPEG files can be unwieldy for direct Web page playback unless streamed over a fast connection.
RealAudio (.rm)	*RealAudio* (http://www.real.com) is the predominant streaming technology currently in use on the Web. It requires a proprietary player, but basic versions of the player are available free.
MIDI	*Musical Instrument Digital Interface* format is not a digitized audio format. It represents notes and other information so that music can be synthesized. MIDI is well supported and files are very small, but it is useful for only certain applications due to its sound quality on PC hardware.
Windows Media Audio (WMA)	Windows Media Technologies (http://www.microsoft.com/windows/windowsmedia) offers a suite of utilities for creating, serving up, and viewing streamed multimedia, including high-quality audio. This is a serious competitor to the Real platform.
SWF	While it is not a music format per se, many sites opt to embed sound within Flash files. Flash files typically import either WAV or MP3 files.

Table 3-4. *Common Web Audio Choices*

rate of 30 frames per second takes up a staggering 27MB per second—and that's without sound. Add CD-quality audio (705,600 bits of data for each second of data; for stereo, double that amount to 1.4 Mbps) and the file size increases proportionately. Granted, these are uncompressed frames and audio, but the point is that a lot of compression as well as bandwidth is needed for high-quality, large-size video.

As with audio, numerous formats are supported for Web-based video, including AVI, QuickTime, MPEG, RealVideo, and ASF. Table 3-5 presents a brief overview of the various Web video formats.

Even with improvements in network and compression technology, audio and video services have a long way to go on the Web if they are to approach the quality and reliability that users are familiar with from radio and television. Until that time, developers should always proceed with caution with real time media technologies. Further, just because audio and video can be delivered over the Web doesn't mean that it should be. Always pick the best media format for the message to be delivered and remember that if you have nothing to say, whether it is in Flash or not isn't going to help. We now switch gears and turn our attention to the programming aspects of the Web medium.

Video Format	Description
AVI	*Audio Video Interleave*. The Video for Windows file format for digital video and audio is very common and easy to specify. AVI files tend to be too large for streaming directly but are often used for small download-and-play clips.
MOV (QuickTime)	MOV is the extension that indicates the use of Apple's QuickTime format (http://www.apple.com/quicktime/). A very common digital video format, it continues its popularity on the Internet.
Windows Media Video (WMV)	The Windows Media platform (http://www.microsoft.com/windows/windowsmedia) also supports streaming video, and, because of the ubiquity of the Windows Media player, this format has become one of the most popular video platforms on the Web.
Real Platform (RM)	The only major challenger to the Windows Media platform, the Real platform delivers surprisingly reliable video at various quality levels depending on end-user bandwidth availability.
Flash (SWF)	Like audio, some developers prefer to avoid the headache of multiple technologies in a page and embed video in Flash or even convert the individual video frames to Flash frames. While not always the best solution for straight streaming, for interactive video clips, Flash is hard to beat.

Table 3-5. *Common Web Video Formats*

Programming Technologies

Understanding the basic idea of adding programming to a site isn't hard, but it's easy to get overwhelmed by the number of technologies to choose from, particularly if you assume that each is very different. The reality is that Web programming technologies can be placed into two basic groups: client side and server-side. Client side technologies are those that are run on the client, generally within the context of the browser, though some technologies like Java applets or ActiveX controls may actually appear to run, or may truly run, beyond the browser, and Helper applications do so implicitly. Of course, programs can and do run instead on the server and thus are appropriately termed server-side programming. Table 3-6 presents the general programming choices available to Web developers; Figure 3-7 shows the relationship of all programming technologies.

The challenge of Web-based programming is making sure to choose the right technology for the job. More often than not, designers are quick to pick a favorite technology, whether it is JavaScript, ColdFusion, or ASP and use it in all situations. The reality is that each technology has its pros and cons. In general, client-side and server-side programming technologies have characteristics that make them complimentary rather than adversarial. For example, when adding a form to a Web site to collect data to save in a database, it is obvious that it would make sense to check the form on the client side to make sure that the user entered the correct information, since it would not force a network round-trip to the server just to check the input data. Client-side programming would make the form validation more responsive and frustrate the user less. On the other hand, putting the data in the database would be best handled by a server-side technology, given that the database would be located on the server side of the equation. Each general type of programming has its place, and a mixture is often the best solution.

Client Side	Server Side
Helper applications	CGI scripts and programs
Browser API programs	Server API programs
—Netscape plug-ins	—Apache modules
—ActiveX Controls	—ISAPI extensions and filters
—Java applets	—Java servlets
Scripting languages	Server-side scripting
—JavaScript	—Active Server Pages (ASP/ASP.NET)
—VBScript	—ColdFusion
	—PHP

Table 3-6. *Web Client-Side and Server-Side Programming Options*

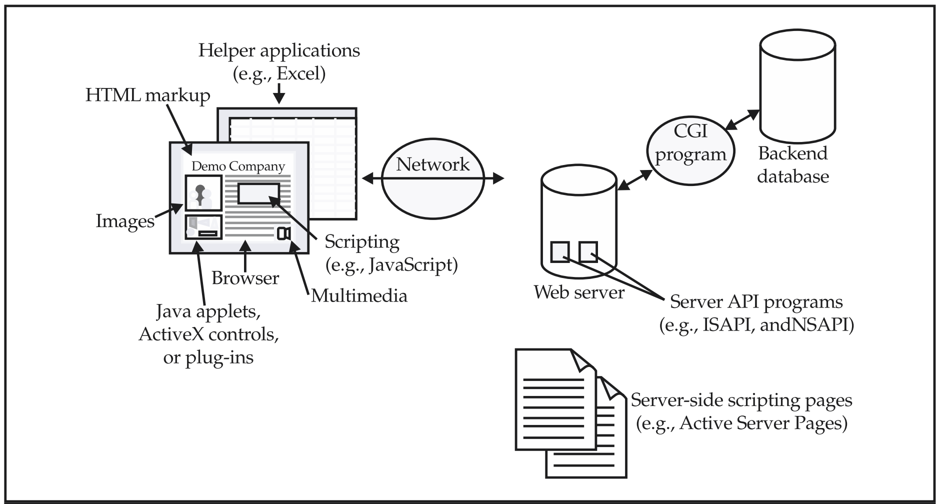


Figure 3-7. *Web Programming Technologies in context*

Rule: Consider using both client-side and server-side technologies in a site, rather than one or the other.

Client-side Programming

The first group of programming facilities we discuss are client-side technologies. Client-side programming technologies run the gamut from simple helper applications—launched upon download of media types like Zip files or of Word documents—to scripts built in browser-based scripting languages, such as JavaScript.

Helpers

One approach to client-side programming comes in the form of programmed solutions, like helper applications. In the early days of the Web, around the time of Mosaic or Netscape 1.*x*, browsers had limited functionality and support for media beyond HTML. If new media types or binary forms were encountered, they had to be passed to an external program called a "helper application." Helper applications generally run outside the browser window. An example of a helper application would be a compression or archive tool like WinZip, which would be launched automatically when a compressed file was downloaded from the Web. Helpers are often problematic because they are not well integrated with the browser and lack methods to communicate back to the Web browser. Because the helper was not integrated within the Web browser, external media types and binaries could not be easily embedded within the Web page. Last, helper applications generally had to be downloaded and installed by the user, which kept many people from using them.

The idea of a helper application is rather simple: it is a program that the browser calls upon for help. Any program can be a helper application for a Web browser, assuming that a MIME type can be associated with the helper. When an object is delivered on the Web, HTTP header information is added to the object, indicating its type. This information is in the form of a MIME type. For example, every Acrobat file should have a content-type of *application/pdf* associated with it. When a browser receives a file with such a MIME type, it will look in its preferences to determine how to handle the file. These options may include saving the file to disk, deleting the file, or handing the file off to another program, such as a helper or browser plug-in. With MIME types and helpers, a developer can put Microsoft Word files on their Web site; users may be able to download them and read them automatically, assuming they have the appropriate helper application. Figure 3-8 overviews the basic way helper applications operate.

Oddly, helper applications are not used as much as they could be. Consider, for example, the use of HTML on an intranet. Within an organization, data may often be created in Microsoft Word or Excel format. While it is possible to easily translate such information into HTML, why would one want to? HTML is relatively expensive to create and, often difficult to update, and may limit the quality of the document's presentation. The main reason that documents are put in HTML is that they can ubiquitously read, meaning we don't have to rely on users having a particular application to read our document, other than a Web browser. However, in an intranet, this probably isn't an issue. In fact, it might be easier to create helper mappings on every system within a corporation rather than to reformat documents in HTML.

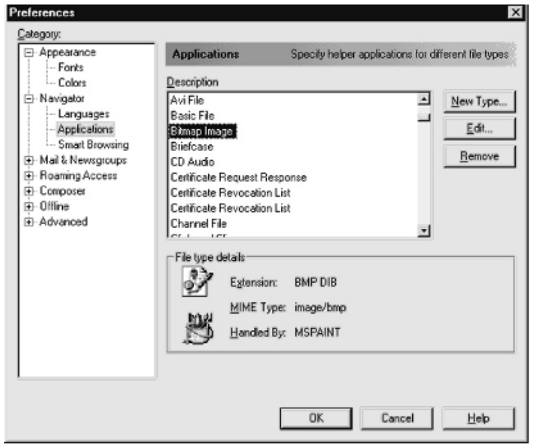
Suggestion: Rely on helper applications when translation to a native Web form is impractical.

Netscape Plug-Ins

Plug-ins were introduced by Netscape in Navigator 2 and have limited support in other browsers, like Opera or Internet Explorer. Internet Explorer favors ActiveX controls, which are described in the next section. Using plug-ins addresses the communication and integration issues that plagued helper applications. Recall that helper applications are not integrated into the design of a Web page, but rather appear in a separate window and may not be able to communicate well with the browser. However, plug-ins are components that run within the context of the browser itself and, thus, can easily be integrated into the design of a page and can communicate with the browser through technologies like JavaScript (which will be introduced in a moment).

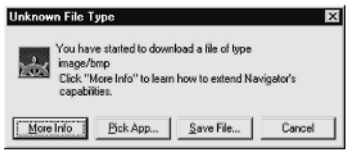
The plug-in approach of extending a browser's feature set has its drawbacks. Users must locate and download plug-ins, install them, and even restart their browsers. Many users find this rather complicated. Netscape 4 offers some installation relief with

1. Browser checks lookup table mapping MIME to action.



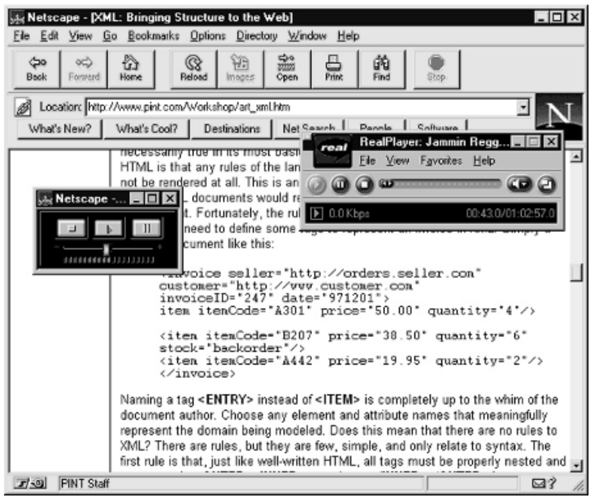
File type details
Extension: BMP DIB
MIME Type: image/bmp
Handled By: MSPAINT

2. If no cation, browser prompts user.



You have started to download a file of type image/bmp
Click "More Info" to learn how to extend Navigator's capabilities.

3. Pass to helper application if set up to do so.


Netscape - [XML: Bringing Structure to the Web]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://www.pint.com/Workshop/art_xml.htm
What's New? What's Cool? Destinations Net Search People Software
RealPlayer: Jammin Regg...
File View Favorites Help
0.0 Kbps 00:43.0/01:02:57.0
Netscape -...
PINT Staff

HTML is that any rules of the lan
not be rendered at all. This is an
documents would re
t. Fortunately, the rul
need to define some
cument like this:

```
 voice seller="http://orders.seller.com"
customer="http://www.customer.com"
invoiceID="247" date="971201">
item itemCode="A301" price="50.00" quantity="4"/>

<item itemCode="B207" price="38.50" quantity="6"
stock="backorder"/>
<item itemCode="A442" price="19.95" quantity="2"/>
</invoice>
```

Naming a tag **<ENTRY>** instead of **<ITEM>** is completely up to the whim of the document author. Choose any element and attribute names that meaningfully represent the domain being modeled. Does this mean that there are no rules to XML? There are rules, but they are few, simple, and only relate to syntax. The first rule is that, just like well-written HTML, all tags must be properly nested and

Figure 3-8. *Overview of helper use*

self-installing (somewhat) plug-ins and other features, but plug-ins remain troublesome. To further combat this problem, many of the most commonly requested plug-ins, such as Macromedia's Flash, are included as a standard feature with Netscape browsers. The standard plug-ins are primarily geared towards media handling and include Macromedia Flash and Shockwave, Adobe Acrobat, and Real player (audio and video). If plug-ins are used, make sure to focus on the popular ones first, given the installation hassle you'll put the user through.

Suggestion: Focus on using only the more popular plug-in technologies unless automatic installation can be performed.

Note *Even if installation were not such a problem, plug-ins are not available on every machine. An executable program, or binary, must be created for each particular operating system; thus, most plug-ins work on Windows systems, though a few of the more popular ones have versions that work on Macintosh and UNIX systems as well.*

The main benefit of plug-ins is that they can be well integrated into Web pages. They may be included by using the **<embed>** or **<object>** tags, though **<embed>** is nearly always favored. For example, to embed a short Flash movie called welcome.swf that can be viewed by a Flash player plug-in, you would use the following HTML fragment:

```
<embed src="welcome.swf" quality="high"
       type="application/x-shockwave-flash" scale="exactfit"
       width="406" height="59" bgcolor="#FFFF00">
</embed>
```

The **<embed>** element displays the plug-in (in this case, a Flash animation) as part of the HTML document. Of course, always remember that the main downside of plug-ins is the barrier to entry they create because of installation and system requirements. If installation can be improved, designers will be able to rely on the technologies provided more and more.

ActiveX

ActiveX (http://www.microsoft.com/activex), which is the Internet portion of the Component Object Model (COM), is Microsoft's component technology for creating small components, or controls, within a Web page. ActiveX distributes these controls via the Internet, adding new functionality to Internet Explorer. Microsoft maintains that ActiveX controls are more similar to generalized components than to plug-ins because ActiveX controls can reside beyond the browser, even within container programs such as Microsoft Office. ActiveX controls are similar to Netscape plug-ins in that they are persistent and machine-specific. Although this makes resource use a problem, installation is not an issue: the components download and install automatically.

Security is a big concern for ActiveX controls. Because these small pieces of code potentially have full access to a user's system, they could cause serious damage. This capability, combined with automatic installation, creates a serious problem with ActiveX. End users may be quick to click a button to install new functionality, only to have it do something malicious, like erase an important system file. The potentially unrestricted functionality of ActiveX controls creates a gaping security hole. To address this problem, Microsoft provides authentication information to indicate who wrote a control, in the form of code signed by a certificate, as shown by the various dialogs in Figure 3-9.

Certificates only provide some indication that the control creator is reputable; they do nothing to prevent a control from actually doing something malicious—that's up to the user to prevent. Safe Web browsing should be practiced by accepting controls only from reputable sources.

Adding an ActiveX control to a Web page requires the use of the **<object>** tag. For example, this markup is used to add a Flash file to a page.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
        codebase="http://download.macromedia.com/pub/shockwave/
        cabs/flash/swflash.cab#version=5,0,0,0"
        width="406" height="59">
     <param name="movie" value="welcome.swf" />
     <param name="quality" value="high" />
     <param name="scale" value="exactfit" />
     <strong>Sorry, no ActiveX in this browser!</strong>
</object>
```

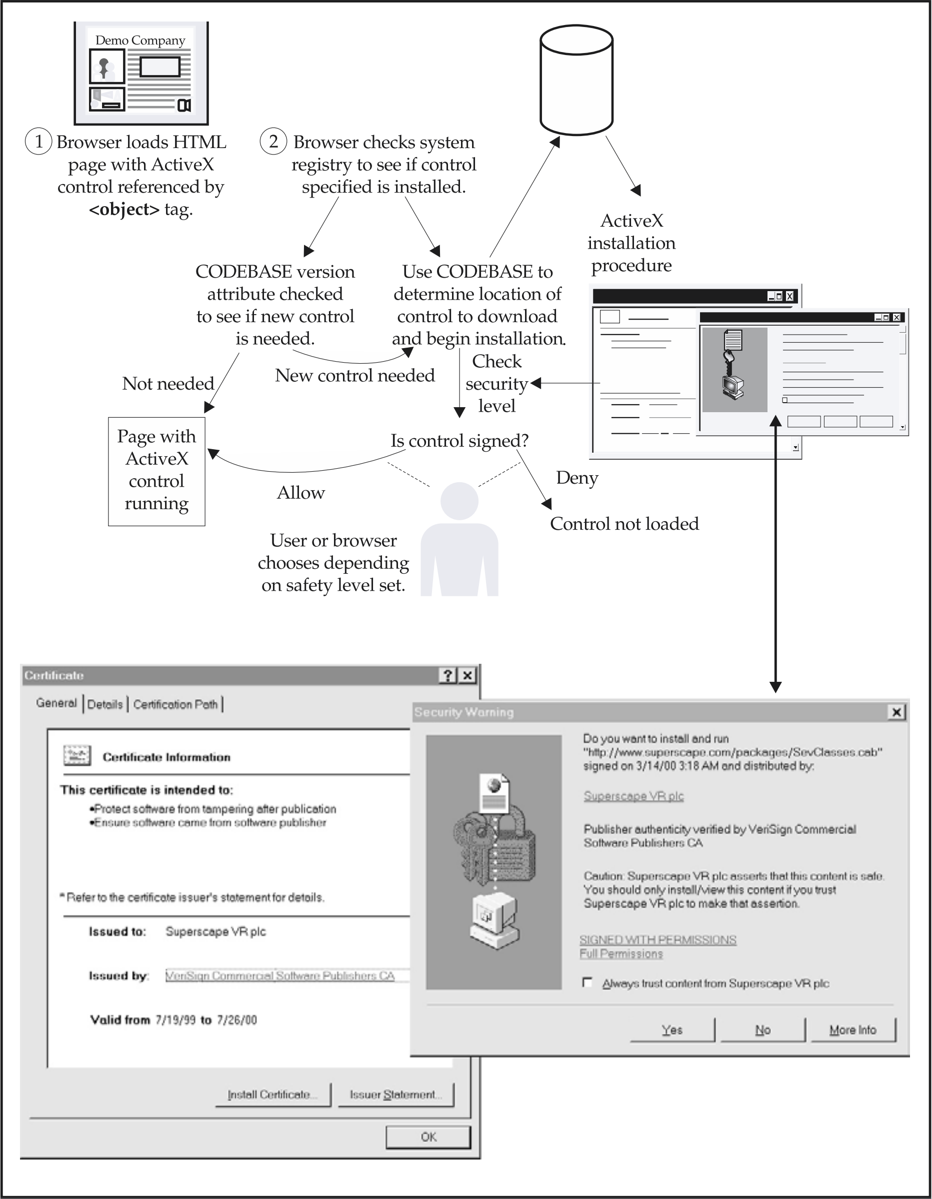
What appears in a browser with no ActiveX? Just a short message indicating the user doesn't have ActiveX. The reality is that the page should allow alternative technologies, such as plug-ins using the **<embed>** tag or even images, before giving a failure message.

Suggestion: If ActiveX controls are used on a public site, make sure to provide alternatives for Netscape or other browsers.

Java

The main downside of component technologies like Netscape plug-ins and Microsoft ActiveX controls is that they are fairly operating system specific. Not every user runs on Windows or even Macintosh, so how do you deal with such a heterogeneous world? One solution is to create a common environment and port it to all systems—this is the intent of Java.

Sun Microsystems' Java technology (http://www.javasoft.com) is an attractive, revolutionary approach to cross-platform, Internet-based development. Java promises a platform-neutral development language, somewhat similar in syntax to C++, that allows programs to be written once and deployed on any machine, browser, or operating


Demo Company
1 Browser loads HTML page with ActiveX control referenced by **<object>** tag.
2 Browser checks system registry to see if control specified is installed.
ActiveX installation procedure
CODEBASE version attribute checked to see if new control is needed.
Use CODEBASE to determine location of control to download and begin installation.
Not needed
New control needed
Check security level
Page with ActiveX control running
Is control signed?
Allow
Deny
Control not loaded
User or browser chooses depending on safety level set.
Certificate
General | Details | Certification Path
Install Certificate...
Issuer Statement...
OK
Security Warning
Yes
No
More Info

Certificate Information

This certificate is intended to:
- •Protect software from tampering after publication
- •Ensure software came from software publisher

* Refer to the certificate issuer's statement for details.

Issued to: Superscape VR plc

Issued by: VeriSign Commercial Software Publishers CA

Valid from 7/19/99 to 7/26/00

Do you want to install and run "http://www.superscape.com/packages/SevClasses.cab" signed on 3/14/00 3:18 AM and distributed by:

Superscape VR plc

Publisher authenticity verified by VeriSign Commercial Software Publishers CA

Caution: Superscape VR plc asserts that this content is safe. You should only install/view this content if you trust Superscape VR plc to make that assertion.

SIGNED WITH PERMISSIONS
Full Permissions

☐ Always trust content from Superscape VR plc

Figure 3-9. *ActiveX signed-code certificate*

system that supports the Java virtual machine (JVM). Web pages use small Java programs, called *applets*, that are downloaded and run directly within a browser to provide new functionality.

Applets are written in the Java language and compiled to a machine-independent byte code in the form of a .class file, which is downloaded automatically to the Java-capable browser and run within the browser environment. But even with a fast processor, the end system may appear to run the byte code slowly compared to a natively compiled application because the byte code must be interpreted by the JVM. This leads to the common perception that Java is slow. The reality is that Java isn't necessarily slow, but its interpretation can be. Even with recent Just-In-Time (JIT) compilers in newer browsers, Java often doesn't deliver performance equal to natively compiled applications.

Rule: Consider end-user system performance carefully when using Java.

Even if compilation weren't an issue, current Java applets generally aren't persistent; they may have to be downloaded again and again. Java-enabled browsers act like thin-client applications because they add code only when they need it. In this sense, the browser doesn't become bloated with added features, but expands and contracts upon use.

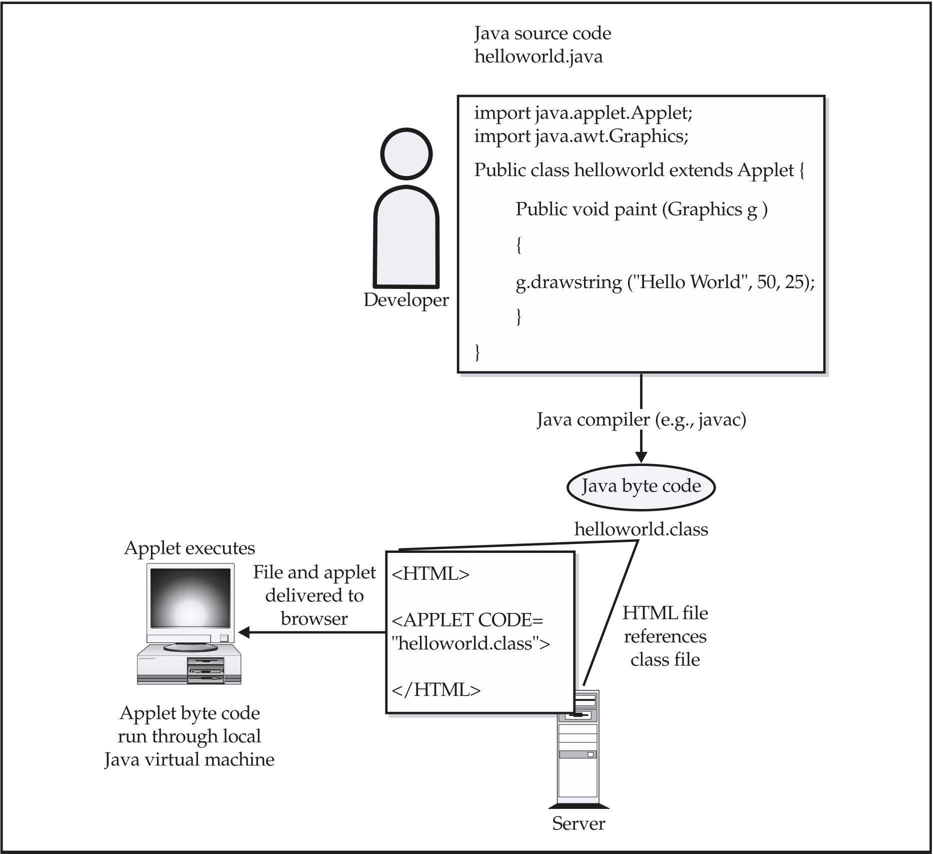
Adding a Java applet to a Web page is relatively easy and can be done using the **<applet>** or **<object>** tag, though **<applet>** is preferred for backward compatibility. If, for example, we had a .class file called helloworld, we might reference it with the following markup:

```
<applet code="helloworld.class"
        height="50"
        width="175">
<h1>Hello World for you non-Java-aware browsers</h1>
</applet>
```

In the preceding code, between **<applet>** and **</applet>** is an alternative rendering for browsers that do not support Java or that have Java support disabled.

The basic idea of how Java is utilized is shown in Figure 3-10.

Security in Java has been a serious concern from the outset. Because programs are downloaded and run automatically, a malicious program could be downloaded and run without the user being able to stop it. Under the first implementation of the technology, Java applets had little access to resources outside the browser's environment. Within Web pages, applets can't write to local disks or perform other potentially harmful functions. This framework has been referred to as the *Java sandbox*. Developers who want to provide Java functions outside of the sandbox must write Java applications that run as separate applications from browsers. Other Internet programming technologies (Netscape plug-ins and ActiveX) provide less safety from damaging programs.


Java source code
helloworld.java
Developer
Java compiler (e.g., javac)
Java byte code
helloworld.class
Applet executes
File and applet delivered to browser
HTML file references class file
Applet byte code run through local Java virtual machine
Server

```
import java.applet.Applet;
import java.awt.Graphics;
Public class helloworld extends Applet {
        Public void paint (Graphics g )
        {
        g.drawstring ("Hello World", 50, 25);
        }
}
```

```
<HTML>

<APPLET CODE=
"helloworld.class">

</HTML>
```

Figure 3-10. *Overview of Java use*

The reality of Java, as far as a Web designer is concerned, is that it really isn't useful on public sites. There are so many different Java Virtual Machines in browsers that the idea of "write once, run everywhere" has been turned into "write once, debug everywhere." The major benefit of Java applets just isn't there. Designers should need no proof other than the fact that major sites that relied on Java applets have in most cases long since removed them. However, within intranets or on the server side in the form of Java servlets, we have seen Java achieve significant success.

JavaScript

JavaScript, which is of no relation to Java other than in name, is the premiere client-side scripting language used in Web browsers. Originally developed by Netscape for Navigator 2.0, the language has grown significantly over the years and is supported by all major browsers in one form or another. For example, Microsoft, supports Jscript, which is their take on the JavaScript language. Standardization of the language came in the form of ECMAScript, but the name JavaScript continues to be used by most developers.

JavaScript is a loosely typed scripting language that has simple uses for tasks like form data validation or minor page embellishments, such as rollover buttons. The inclusion of JavaScript in an HTML page is primarily handled by the **<script>** tag. For example, in this short fragment,

```
<h1>About to leave HTML</h1>
<script type="text/javascript">
<!--
  alert("Hello from JavaScript!");
//-->
</script>
<h1>Welcome back to HTML</h1>
```

we see a statement printed in an HTML document, then an alert dialog is created by JavaScript, and finally another HTML statement in executed. The interaction between HTML and JavaScript is significant, and mastery of markup is required to reap the most benefits from this technology. JavaScript will be presented in this book in small, hopefully palatable, doses to improve page usability. Some techniques for correct JavaScript use will also be presented. For an in-depth discussion, readers should see the links provided or the companion book, *JavaScript: The Complete Reference*.

Information about ECMAScript and JavaScript can be found at these URLs:

- ECMAScript Spec: http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM
- Netscape JavaScript Information: http://developer.netscape.com/javascript/
- Microsoft Scripting Information: http://msdn.microsoft.com/scripting

Document Object Model

With the rise of the standardized *document object model*, or *DOM*, JavaScript is poised to become nearly as important as HTML or CSS for Web developers, because it will provide the ability to manipulate any aspect of an HTML document. In the past, page manipulations were possible using browser and document objects defined by each browser vendor. Browser differences made all but the simplest scripts difficult to implement. The W3C DOM specification promises to help ease cross-browser scripting because it specifies a language-neutral interface that allows programs and scripts to dynamically access and update the content, markup, and style of Web documents.

Since the DOM is used via JavaScript to manipulate HTML documents, this usage is often referred to as *dynamic HTML*, or *DHTML*. However, the term really is deceptive and its usage is not encouraged. The DOM comes in two primary variants at the moment: DOM Level 1, which provides access and manipulation facilities for basic markup elements and bindings to manipulate HTML tags, and DOM Level 2, which extends the interface to allow manipulation of CSS properties and provides a richer event interface.

Online information about the DOM can be found at these URLs:

- DOM Level 1 Spec: http://www.w3.org/TR/REC-DOM-Level-1/
- DOM Level 2 Core Spec: http://www.w3.org/TR/DOM-Level-2-Core/
- DOM Level 2 Events Spec: http://www.w3.org/TR/DOM-Level-2-Events/
- DOM Level 2 Style Spec: http://www.w3.org/TR/DOM-Level-2-Style/

Server-Side Technologies

The Web server handles the server side of the Web communications medium, responding to the various HTTP requests made to it. Servers may directly return various file objects, such as HTML documents, images, multimedia files, scripts, or style sheets, or they may run executable programs, which return a similar result. In this sense, the Web server acts both as a file server and as an application server. We will survey the basic components of the server side here before addressing the network components of the medium.

Web Servers

Like the Web browser, the Web server frames the environment of each Web transaction. The term “Web server” is usually understood to mean both the hardware and software. The major issue with hardware is whether the Web server is capable of handling the memory, disk, and network input/output requirements resulting from site traffic. The interplay of operating systems, such as UNIX or Windows 2000, and Web server software also is closely related to performance, as is security.

From Apache to Zeus, all Web server software platforms handle basic HTTP transactions, but all tend to offer more than basic file serving facilities. Most Web server platforms provide basic security and authentication services, logging, and programming facilities. An in-depth discussion of the popular servers and their facilities is presented in Chapter 17; here, we will focus only on the programming aspects of sites.

CGI

The oldest of the server-side programming technologies, CGI (Common Gateway Interface) programs can be written in nearly any programming language, though commonly Perl is associated with CGI applications. CGI is not a language or program, but in fact just a way to program—unlike other server-side programming environments, which define both language and style. CGI defines the basic input and output methods for server-side programs launched by a Web server, as illustrated in Figure 3-12. While assumed by some to be slow and insecure, CGI is adequate for many Web development projects when correctly understood and used.

Online information about CGI can be found at these URLs:

- CGI Overview and Documentation: http://hoohoo.ncsa.uiuc.edu/cgi/overview.html
- CGI Resource Index: http://cgi.resourceindex.com/

Server-Side Scripting

Server-side scripting technologies, such as Microsoft's Active Server Pages (ASP) or Macromedia's ColdFusion, allow dynamic pages to be created easily. All server-side scripting languages, including the popular ASP, ColdFusion, JSP, and PHP languages, work fairly similarly. The idea is that script templates that contain a combination of HTML and scripting language are executed server side to build a resulting Web page. Usually, some form of server engine intercepts page requests, and when files with certain extensions—such as .asp, .cfm, .jsp, .php, or .shtml—are encountered, the script elements in the page are replaced with the resulting markup output. The process is illustrated in Figure 3-13.

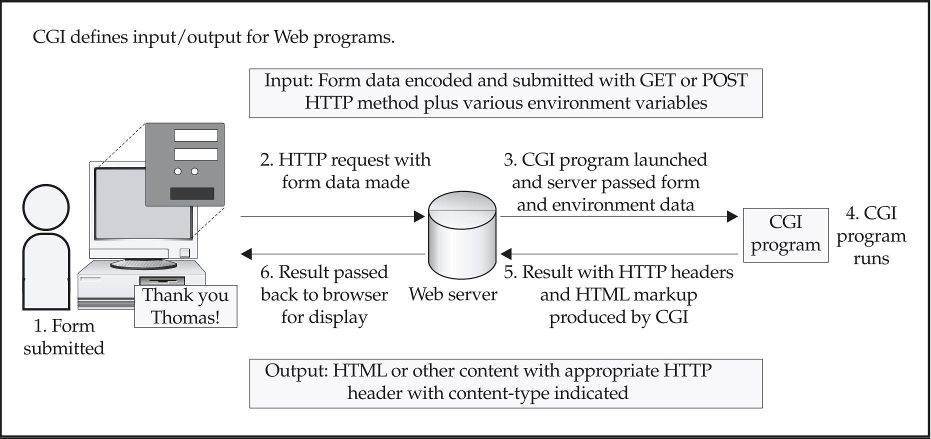
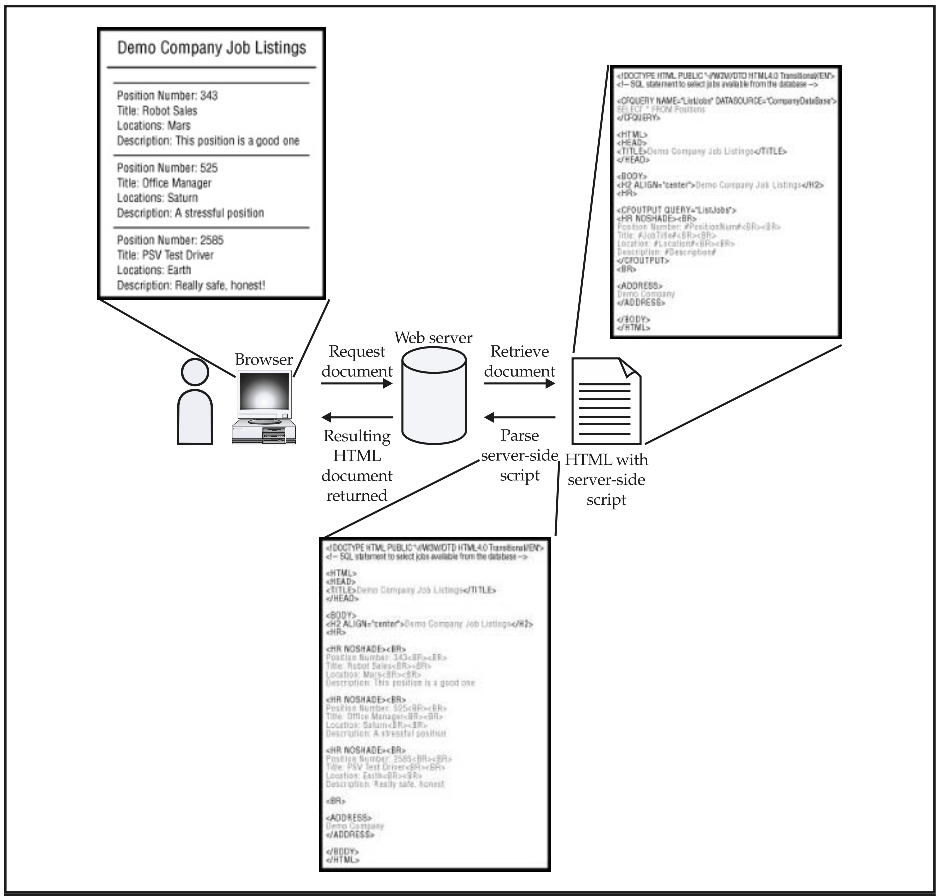

CGI defines input/output for Web programs.
Input: Form data encoded and submitted with GET or POST HTTP method plus various environment variables
2. HTTP request with form data made
3. CGI program launched and server passed form and environment data
CGI program
4. CGI program runs
6. Result passed back to browser for display
Web server
5. Result with HTTP headers and HTML markup produced by CGI
Thank you Thomas!
1. Form submitted
Output: HTML or other content with appropriate HTTP header with content-type indicated

Figure 3-11. *Overview of CGI*



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.0 Transitional//EN">
<!-- SQL statement to select jobs available from the database -->

<CFQUERY NAME="ListJobs" DATASOURCE="CompanyDataBase">
SELECT * FROM Positions
</CFQUERY>

<HTML>
<HEAD>
<TITLE>Demo Company Job Listings</TITLE>
</HEAD>

<BODY>
<H2 ALIGN="center">Demo Company Job Listings</H2>
<HR>

<CFOUTPUT QUERY="ListJobs">
<HR NOSHADE><BR>
Position Number: #PositionNum#<BR><BR>
Title: #JobTitle#<BR><BR>
Location: #Location#<BR><BR>
Description: #Description#
</CFOUTPUT>
<BR>

<ADDRESS>
Demo Company
</ADDRESS>

</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.0 Transitional//EN">
<!-- SQL statement to select jobs available from the database -->

<HTML>
<HEAD>
<TITLE>Demo Company Job Listings</TITLE>
</HEAD>

<BODY>
<H2 ALIGN="center">Demo Company Job Listings</H2>
<HR>

<HR NOSHADE><BR>
Position Number: 343<BR><BR>
Title: Robot Sales<BR><BR>
Location: Mars<BR><BR>
Description: This position is a good one

<HR NOSHADE><BR>
Position Number: 525<BR><BR>
Title: Office Manager<BR><BR>
Location: Saturn<BR><BR>
Description: A stressful position

<HR NOSHADE><BR>
Position Number: 2585<BR><BR>
Title: PSV Test Driver<BR><BR>
Location: Earth<BR><BR>
Description: Really safe, honest

<BR>

<ADDRESS>
Demo Company
</ADDRESS>

</BODY>
</HTML>
```

Figure 3-12. *Overview of server-side scripting*

Server-side scripting languages are often used to build dynamic pages from databases, personalize content for users, or generate reusable components in pages. The syntax for each language is different, and many developers are somewhat religious about the merits of one language over the next, but the fact of the matter is that none of them scales well for extremely high-volume sites. Such sites usually require server API programs, which are discussed next.

Online information about server-side scripting can be found at these URLs:

- ASP Information: http://msdn.microsoft.com/asp
- ColdFusion Information: http://www.macromedia.com/software/coldfusion/
- PHP Information: http://www.php.net/
- JSP Information: http://java.sun.com/products/jsp

Server APIs

Server API (Application Programming Interfaces) programs are special server-side programs built to interact closely with the Web server. A simple way to think of server API programs is as plug-ins to a Web server. Common APIs include ISAPI for Microsoft's IIS server, NSAPI for the Netscape/IPlanet/Sun server, Apache Modules for Apache, and Java servlets for Java-enabled Web servers. The benefit of server API programs is that their close interaction with the Web server generally translates into high performance. The downside, of course, is the complexity of writing such a program and the possibility that an errant server module may actually crash the entire server.

Information about server APIs can be found at these URLs:

- Apache Module Information: http://modules.apache.org/
- ISAPI Filters/Extension Information: http://msdn.microsoft.com
- Java Servlet Information: http://java.sun.com/products/servlet

Network and Related Protocols

The underlying protocols of the Web include the TCP/IP suite of networking protocols. Not a single protocol but a group of protocols, TCP/IP is what makes all services on the Internet possible. Individually, IP (Internet Protocol) provides the basic addressing and routing information necessary to deliver data across the Internet. However, TCP (Transport Control Protocol) provides the facilities that make communications reliable, such as correction and retransmission. Together, in conjunction with the Domain Name Service (or DNS), which is the process of translating fully qualified domain names like www.webdesignref.com into their underlying IP addresses (66.45.42.235), we have the ability to build higher-level services, such as e-mail or Web sites, on the Internet. Knowledge of lower-level protocols may seem pointless to many Web designers, but it is particularly helpful to understand networking details when designing extremely scalable Web sites. However, regardless of site aims, the next protocol discussed should be understood by every Web designer.

HTTP

HTTP (Hypertext Transport Protocol) is the application-level protocol that handles the discussion between a user-agent, generally a Web browser, and a Web server. The protocol is simple and defines eight basic commands (GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE, and CONNECT) that can be made by a user-agent to request or manipulate data. Responses may contain both numeric and textual codes (for example, 404 Not Found) and associated data.

The simplicity of the HTTP protocol is both a blessing and a curse. It is simple to implement, but its lack of state management and its performance problems plague Web developers. The HTTP 1.1 specification as defined in RFC 2616 addressed many of the performance problems, but state management still has to be resolved using cookies, hidden data variables, or extended URLs. An overview of HTTP can be found in Chapter 17, while Appendix G details its request and response format.

Information about HTTP can be found at these URLs:

- W3C HTTP Activity: http://www.w3.org/Protocols/
- HTTP 1.1 Specification: ftp://ftp.isi.edu/in-notes/rfc2616.txt

MIME

MIME (Multipurpose Internet Mail Extensions), the unsung hero of Web protocols, is used by browsers to determine what kind of data they have received from a server. Specifically, an HTTP header called Content-type contains a MIME value, which is looked up by a browser to understand what type of data it is receiving and what to do with it. Servers append MIME types to HTTP headers either by generating them from a program or by mapping a file extension (for example, .html) to an appropriate MIME type (for example, text/html). MIME allows Web sites to deliver any type of data, not just the common Web formats like HTML.

Information about MIME can be found at this URL:

- MIME Specification: http://www.ietf.org/rfc/rfc2045.txt

Addressing: URL/URI/URNs/URCs

To request and link to Web pages, it is necessary to use an addressing scheme. Web users are familiar with *URLs (Uniform Resource Locator)*, like http://www.webdesignref.com/, which specify protocol and location. In specifications, *URI (Uniform Resource Identifier)* is the more commonly accepted term for short names or address strings that refer to a resource on the Web. Yet, whatever the name, URI or URLs do not provide all that may be required on the Web in the future, since they specify only location. *Uniform Resource Names (URNs)* and *Uniform Resource Characteristics (URCs)* may eventually be implemented to provide non-location-dependent addressing and extra information about resources, respectively. However, resource characteristics are more commonly specified using a form of meta data, as described next.

Online information about addressing can be found at this URL:

- W3C Addressing Activity: http://www.w3.org/Addressing/

Meta Data

Meta data is defined as data about data. Web developers may be familiar with putting meta data in a Web page using the **<meta>** tag. Often, this is used to specify keywords and descriptions for search engines. For example,

```
<meta name="keywords" content="robots,androids, bots">
<meta name="description" content="Demo Company makes the best
      robots in the Solar System!">
```

Meta data is also used in Web pages to control page characteristics, particularly those related to HTTP headers. For example,

```
<meta http-equiv="Expires" content="Wed, 15 May 2002 08:21:57 GMT" />
```

would set an expiration date for a Web page using the HTTP expires header.

The key to meta data is having a consistent and descriptive enough vocabulary for describing data. The Resource Description Framework (RDF) provides a standard way for using XML to represent meta data in the form of statements about properties and relationships of items on the Web. However, RDF itself is just a framework and needs a vocabulary. A popular vocabulary called Dublin Core initially has started to gain some traction. However, at the time of this edition's writing, the use of meta data vocabulary beyond the simple **<meta>** tag for keywords and descriptions is not common practice on the Web, though it is prevalent in many large sites and very common in large intranets.

Online information about meta data can be found at these URLs:

- W3C RDF Information: http://www.w3.org/RDF/
- Dublin Core Metadata Initiative: http://dublincore.org/

Web Services

Finally, the latest wrinkle in the Web medium is the rise of Web Services. The basic concept of Web Services is that Web sites may interact directly with each other, exchanging information or even running programs remotely. Web Services allow for complex distributed applications to be built using the pieces of various Web sites. For example, imagine running a small travel site and offering flight, hotel, and car rental booking services directly from your site through a large travel partner's Web site without the user being aware. Web Services would provide the facilities for your site to talk to others and seamlessly make such a service possible.

The key to Web Services is the use of standardized message formats, typically specified in XML. A protocol called SOAP (Simple Object Access Protocol) appears to be the leading candidate for Web Services. However, others do exist, and Web Services are not prevalent enough yet to assume victory for SOAP. Beyond messaging protocols, Web Services also require a facility for service providers to describe their offered services, and for users to discover the services they require. So far, service description is being handled by a protocol called WSDL (Web Service Description Language), while service discovery is handled by UDDI (Universal Description, Discovery, and Integration). As mentioned, these protocols may not necessarily become standard; but regardless of what protocol is adopted, Web Services will provide for a much richer Web experience, which is coming to be known as the *semantic Web*.

Information about Web Services can be found at these URLs:

- W3C Web Services Activity: http://www.w3.org/2002/ws/
- W3C Semantic Web Activity: http://www.w3.org/2001/sw/

Note *A good portion of the activity in the Web Services space revolves around Microsoft's .NET technology, which also provides SOAP as well as a sophisticated Web programming environment. However, what .NET actually means to Web Services and what it includes are still very fluid. The best source of information on the Microsoft variant of Web Services can be found at http://www.microsoft.com/net/.*

Summary

Understanding the various aspects of the Web medium is mandatory for aspiring Web designers. Even if the focus is only on front-end interface creation, designers should have at least passing knowledge of the various components of the Web sites, ranging from addressing systems to XML-based Web Services. While it might be said that architects often make lousy carpenters, it can also be assumed that they generally have some sense of the properties of the building materials their projects use, and so should Web architects. Some of these "building materials," such as Web browsers, HTML/XHTML, CSS, JavaScript, and media formats, should already be very familiar, while others, like XML and networking protocols, may seem of little use to visual designers. However, with the transition away from simple print-oriented Web design to more interactive software-focused Web sites, designers would be well advised to become more proficient in programming and networking technologies. The next chapter explores just how Web sites are built and provides a useful overview of the processes that can be employed to guide complex Web projects.

This page intentionally left blank.

Chapter 4

The Web Design Process

Building a great Web site can be challenging. With so many different components, ranging from visual design to database integration, there is plenty of room for things to go wrong. In order to minimize the risk of a Web project failing, we need a process to guide us. Unfortunately, some Web designers utilize what might be called the "NIKE" method of Web development—they *just do it*, often with little forethought or planning. Building a site this way is not methodical. The site's goals tend to be loosely defined, the process more intuitive than procedural, and the end result highly unpredictable. Sites developed this way are like plants. They grow organically—sometimes into a beautiful flower, but more often into a tangled mess. Complex Web sites require careful planning. A process or methodology should always be employed to help guide our Web design and development efforts.

The Need for Process

Today, Web development finds itself in a crisis similar to the "software crisis" of the late 1960s. A few years ago most Web sites were little more than digital brochures, or "brochureware." Creating such a site didn't require a great deal of planning—often, it was sufficient simply to develop an interface and then to populate the site with content. Since then, sites have become much larger and more complex. With the introduction of interactivity and e-commerce, sites have clearly moved away from brochureware to become full-fledged software applications. Despite this, many developers have yet to adopt a robust site-building methodology, but continue to rely on ad hoc methods.

Note *The "software crisis" refers to a time in the software development field when increasing hardware capabilities allowed for significantly more complex programs to be built. It was challenging to build and maintain such new programs because little methodology had been used in the past, resulting in numerous project failures. Methodology such as structured or top-down design was introduced to combat this crisis.*

Evidence of the crisis in Web development practices is everywhere. Unlike the in-house client/server software projects of the past, the dirty laundry of many failed Web projects is often aired for all to see. The number of pages that seem to be forever "under construction" or "coming soon" suggests that many Web sites are poorly planned. Some sites have been in a state of construction for years, judging by their content or date of last modification. These online ghost towns are cluttered with old content, old-style HTML, dated technologies, broken links, and malfunctioning scripts. Don't discount some of these problems as mere typos or slight oversights. A broken link is a catastrophic failure, like a software program with menus that just don't go anywhere!

The reason why sites exhibit problems certainly vary. Some sites may deteriorate simply because their builders got bored or moved on. Other sites may fall apart because the site wasn't considered useful, or funding was withdrawn. Still other sites probably just couldn't be completed because the sites overwhelmed the developers—they may not have understood the tools they were working with, or were not versed well enough in the medium's restrictions. The almost countless dead sites on the Web suggest that Web development projects are risky and often fail.

Ad Hoc Web Process

Often the process to build a Web site is to simply implement the site, perform a brief visual test in a browser, and then release it to the world. This is similar to the "by the seat of your pants" code-and-test process used in small software projects. The numerous problems in Web sites built using informal methods show the problem with this overly simplistic approach. Today's process for the Web is so fast that the process almost boils down to two steps: implement and then release. Visual Web design tools encourage this design-on-the-fly approach. Some tools encourage the developer to immediately mock up an interface and later use wizards to add functionality, while others can create huge amounts of code but have an interface added later on. There is no doubt that a speedy approach to development, given the time demands of the Web, is important. Releasing a shoddy, poorly thought-out site, however, may backfire when users become frustrated with the site's problems.

In the software industry, most professionals tend to agree that such informal or "design as you go along" methods are only good for small projects, generally with only one programmer, and where future maintenance is not expected to be great. Often, programs built with such little planning exhibit convoluted programming logic—often called "spaghetti code," which is very difficult to maintain because nobody besides the initial developer can untangle the mess. Even the initial developer may forget the meaning of the code over time.

Web sites exhibit similar patterns. Small Web sites that have short expected life spans are often built by one person using little methodology. Inspection of the site's underlying HTML, JavaScript, and navigation structure will frequently show that "spaghetti code" is being served, complete with a side dish of "markup salad."

Planning can help offset some of the problems that may be encountered during a Web development project. Unfortunately, in the ad hoc Web process, planning is often limited to a few brief meetings, a brief but incomplete collection of potential content, and maybe a hastily conceived flow diagram. The amount of time spent planning is generally negligible next to the amount of time spent during implementation. Of course, it is always possible to plan too much and suffer from a form of "analysis paralysis," which keeps a site from ever getting built, but this is relatively uncommon. Always have the amount of planning be proportional to the complexity of the project. The key to dealing with project management challenges is to create a formal process by which to plan, implement, test, and deploy a site in a structured manner.

Basic Web Process Model

To help reduce the difficulty in constructing sites, we should adopt a *process model* that describes the various phases involved in Web site development. Each step can then be carefully performed by the developer, using guidelines and documentation along the way telling the developer how to do things and ensuring that each step is carried out properly. An ideal process model for the Web would help the developer address the complexity of the site, minimize the risk of project failure, deal with the near certainty of change during the project, and deliver the site quickly with adequate feedback for management during the process. Of course, the ideal process model would also have to be easy to learn and execute. This is a pretty tall order, and it is unlikely that any single process model is always going to fit all the particular requirements of every project.

The most basic process model used in Web site development should be familiar to most people, as it is deductive. The basic model starts with the big picture and narrows down to the specific steps necessary to complete the site. In software engineering, this model is often called the *waterfall model*—or sometimes the *software lifecycle model*, because it describes the phases in the lifetime of software. The stages in the waterfall model proceed one after another until conclusion. The model starts first with a planning stage, then a design phase, then implementation and testing, and ends with a maintenance phase. The phases may appear to be distinct steps, and the progress from one stage to another may not always be obvious. Further, progress isn't always toward a conclusion; on occasion, previous steps may be revisited if the project encounters unforeseen changes. The actual number of steps and their names varies from person to person, but a general idea of the waterfall model is shown in Figure 4-1.

Note *While this model of Web development is probably the most common, many Web designers seem to think they invented a special form of it; then they publish it on their Web site as their patent-pending design process. There really isn't anything new here, whether there are five steps or seven steps or whether the names are complex sounding or simple. Always remember that what matters is that the model helps the site's production and improves the final result.*

The good thing about the pure waterfall approach is that it makes developers plan everything up front. That is also its biggest weakness. There is often a great deal of uncertainty as to what is required to accomplish a Web project, particularly if the developer has not had a great deal of Web development experience. Another problem with this process model is that each step is supposed to be distinct. The reality is that in Web development, as in software, steps tend to overlap, influence previous and future steps, and occasionally need to be repeated. Unfortunately, the waterfall approach can be fairly rigid and may require the developer to stop the project and redo many steps if too many changes occur. In short, the process doesn't deal well with change. Even so,

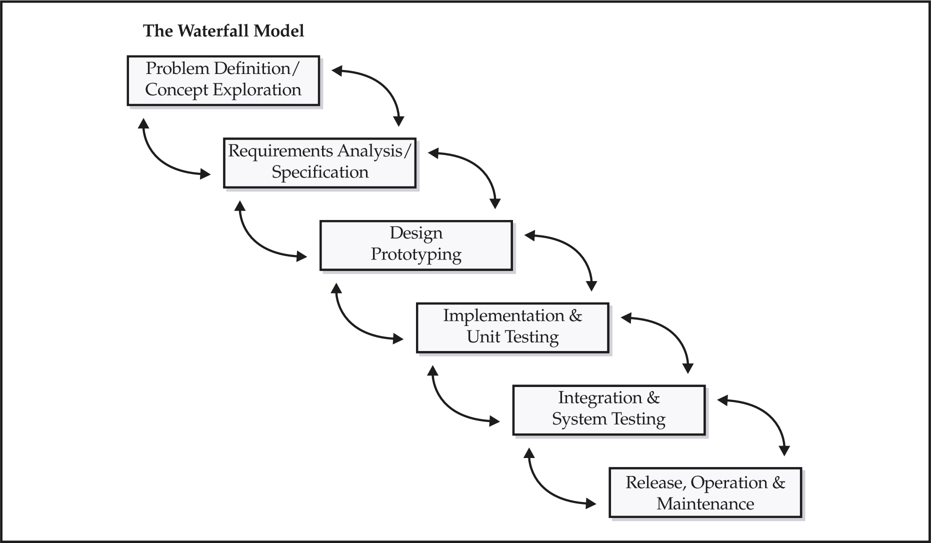

The Waterfall Model
Problem Definition/ Concept Exploration
Requirements Analysis/ Specification
Design Prototyping
Implementation & Unit Testing
Integration & System Testing
Release, Operation & Maintenance

Figure 4-1. *The waterfall model*

the waterfall model for site design continues to be very popular because it is both easy to understand and easy to follow. Further, the distinct steps in the process appeal to management, as they can be easily monitored and serve as project milestones.

Modified Waterfall

One important aspect of the waterfall model is that it forces developers to plan up front. However, because of all the steps required in the process, many developers tend to rush through the early stages and end up repeating them again later on or building a site based upon flawed ideas. The process is so rigid that it doesn't support much exploration, and it may cause unnecessary risk. One possible improvement is to spend more time in the first few stages of the waterfall and iterate a few times, exploring the goals and requirements of the site before entering into the design and implementation phase. Because of the cyclical nature of this process, it has been dubbed the "modified waterfall with whirlpool" (similar to the small whirlpools that are often found near a waterfall in nature). When approaching a project with a high degree of uncertainty, the modified waterfall with whirlpool approach, as illustrated in Figure 4-2, is a good idea.

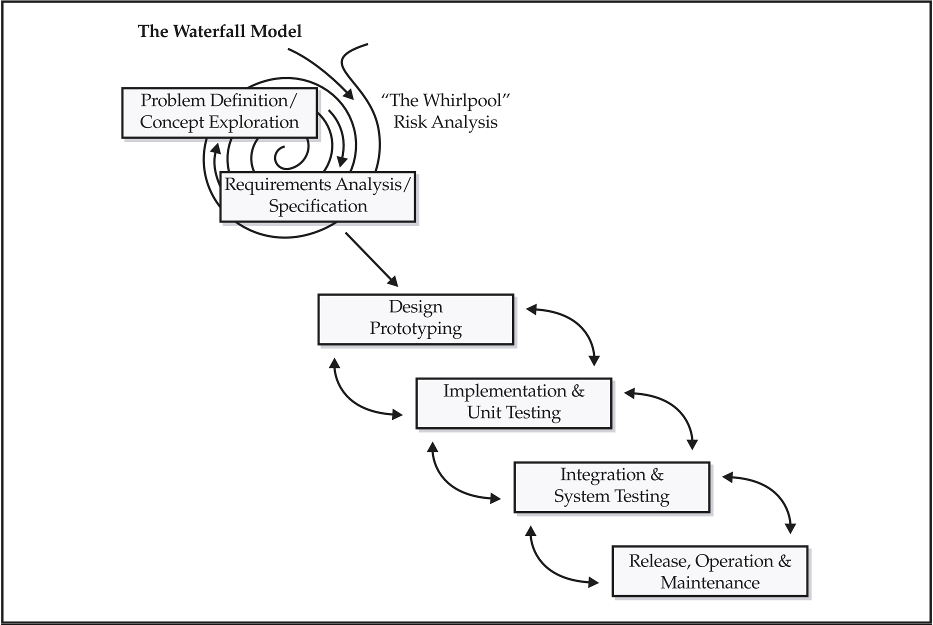

The Waterfall Model
"The Whirlpool" Risk Analysis
Problem Definition/ Concept Exploration
Requirements Analysis/ Specification
Design Prototyping
Implementation & Unit Testing
Integration & System Testing
Release, Operation & Maintenance

Figure 4-2. *Modified waterfall with risk analysis whirlpool*

Joint Application Development

The last software development process model that makes sense for Web site development is called *joint application design*, or JAD. It is also called *evolutionary prototyping* because it involves evolving a prototype site to its final form in a series of steps. Rather than creating a mock site to test a theory, a prototype is built and shown to the client or potentially the end user. The concerned party then provides direct feedback that is used to guide the next version of the prototype, and so on until the final form is developed. The basic concept of JAD is shown in Figure 4-3.

Many aspects of the JAD process model seem appropriate for Web development, particularly when it is difficult to determine the specifics of a project. The process is very incremental, as compared to the large release approach of the waterfall model, so it also appears to be faster. However, JAD can have some serious drawbacks. First, letting users see an unfinished site could harm the relationship between the users and developer. Even when users want to actively participate in guiding the project, we must always remember that users are not designers. This guiding Web design principle should always be remembered, as users may steer development off course with

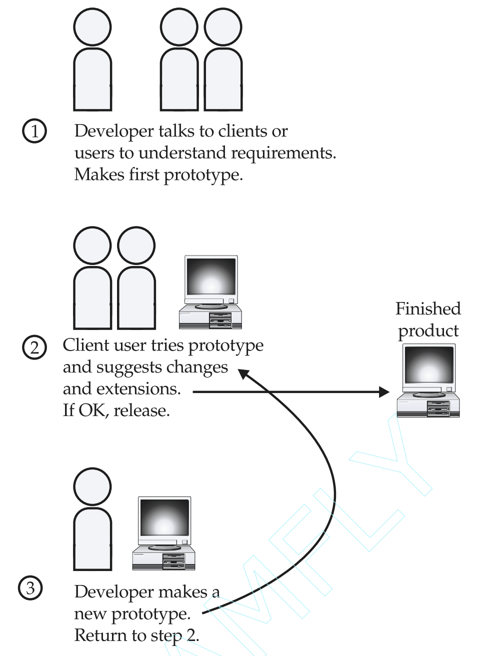

1 Developer talks to clients or users to understand requirements. Makes first prototype.
2 Client user tries prototype and suggests changes and extensions. If OK, release.
Finished product
3 Developer makes a new prototype. Return to step 2.

Figure 4-3. *Joint application design in action*

unrealistic demands. Budgeting a project run in a JAD style is also difficult, since the number of revisions can't be predicted. If users are fickle, costs can quickly spiral out of control. Remember that the core concept behind JAD is to build the wrong site numerous times until the correct site falls out. Despite its drawbacks, JAD has its place in Web development, particularly in maintenance projects. However, for initial project development, JAD is best left to experienced developers—particularly those who are capable of communicating with users well.

A few possible candidates for guiding a Web project have been discussed. Numerous others exist and might serve a developer equally well. Remember that the act of building a site is to clearly identify a problem to solve or a goal to reach and then attempt to arrive at an outcome in a consistent and enlightened manner. Site development should be approached critically and deliberately rather than casually or passively. A critical approach doesn't necessarily rule out chance or sudden inspiration, and it does offer the opportunity to direct it. Designers should not look at the use of Web site engineering concepts as limiting factors, but rather as something that can guide design.

Approaching a Web Site Project

In theory, Web site engineering process models make sense, but do they work in practice? The answer is a resounding *Yes*. However, site development rarely works in a consistent manner, because of the newness of the field, the significant time constraints, and the ever-changing nature of Web projects. Developers should always proceed with caution. To guide development, a process model should be adopted at the start of the project. If the site is brand new or the addition is very complex, the waterfall model or the modified waterfall with whirlpool model should be adopted. If the project is an extension maintenance project, is relatively simple or has many unknown factors, joint application design may make sense. Regardless of the project, the first step is always the same: set the overall goal for the project.

Goals and Problems

Many Web site projects ultimately fail because they lack clear goals. In the first few years of Web design, many corporate sites were built purely to show that the firm had a site. Somehow, without a site the firm would not be progressive or a market leader; competitors with sites were considered a threat. Many times, the resulting site provided of little benefit because it wasn't really designed to provide anything other than a presence for the company. As familiarity with the Web has grown, the reasons for having Web sites have become clearer. Today, site goals have become important and are usually clearly articulated up front. However, don't assume that logic rules the Web—a great number of site development projects continue to be driven by pure fancy and are often more reactive to perceived threats than intended to solve real problems.

Coming up with a goal for a Web site isn't difficult; the problem is refining it. Be wary of vague goals like "provide better customer service" or "make more money by opening up an online market." These may serve as a good sound bite or mission statement for a project, but details are required. Good goal statements might include something like:

- Build a customer support site that will improve customer satisfaction by providing 24/7 access to common questions and result in a 25 percent decrease in telephone support.
- Create an online automobile parts store that will sell at least $10,000/month of products directly to the consumer.
- Develop a Japanese food restaurant site that will inform potential customers of critical information such as hours, menu, atmosphere, and prices, as well as encourage them to order by phone or visit the location.

Notice that two of the three goal statements have measurable goals. This is very important, as it provides a way to easily determine success or failure, as well as assign a realistic budget to the project. The third goal statement does not provide an obviously measurable goal. This can be dangerous because it is difficult to convince others that the site is successful or to even place a value on the site. In the case of the restaurant site, a goal for number of viewers of the site or a way to measure customer visits using a coupon would help. Consider a revised goal statement like this:

- Develop a Japanese food restaurant site that will inform at least 300 potential customers per month of critical information such as hours, menu, atmosphere, and prices as well as encourage them to order by phone or visit the location.

The simple addition of a particular number of visitors makes the goal statement work. By stating a number of desired visitors, the restaurant owner could compare the cost of placing advertisements in print or on the radio versus the cost of running the site to provide the same effective inquiry rate.

Brainstorming

In general, coming up with a goal statement is fairly straightforward. The largest problem is keeping the statement concise and realistic. In many Web projects there is a desire to include everything in the site. Remember, the site can't be everything to everyone; there must be a specific audience and set of tasks in mind. To determine goals, a brainstorming session is often required. The purpose of a brainstorming session is simply to bring out as many potential ideas about the site as possible. A white board and Post-it notes are useful during a brainstorming session to quickly write down or modify any possible ideas for the site.

Oftentimes, brainstorming sessions get off track because participants jump ahead or bring too much philosophy about site design to the table. In such cases, it is best to focus the group by talking about site issues they should all agree on. Attempt to find a common design philosophy by having people discuss what they don't want to see in the site. Getting meeting participants to agree they don't want the site to be slow, difficult to use, and so on is usually easy. Once you obtain a common goal in the group, even if it is just that they all believe that the site shouldn't be slow, future exploration and statements of what the site should do seem to go smoother.

Note *When conducting a project to redo a site, be careful not to run brainstorm meetings by berating the existing site, unless no participant in the project has any ownership stake in the site. A surefire way to derail a site overhaul project is to get the original designers on the defensive because of criticism of their work. Remember, people have to build sites, so building a positive team is very important.*

Narrowing the Goal

During the brainstorming session, all ideas are great. The point of the session is to develop what might be called the *wish list*. A wish list is a document that describes all possible ideas for inclusion in a site regardless of price, feasibility, or applicability. It is important not to stifle any ideas during brainstorming, lest this take away the creative aspect of site development. However, eventually the wish list will have to be narrowed down to what is reasonable and appropriate for the site. This can be a significant challenge with a site that may have many possible goals. Consider a corporate site that contains product information, investor information, press releases, job postings, and technical support sections. Each person with ownership stakes in a particular section will think his or her section is most important. Everyone literally wants a big link to his or her section to be on the home page. Getting compromise with so many stakeholders can be challenging!

One possibility for narrowing the goal is to use small sheets of papers or a deck of 3 × 5 cards. Have each one of the ideas written on a card and put them in a large pile. Now go around the room and have each person pull out one card at a time to include in the site on the basis of importance. Of course, make sure to limit the number of cards pulled from the pile. By performing a procedure like this, it's more likely that all of the most important ideas will surface. Unfortunately, this exercise may fail—particularly if the participants place a great deal of ownership in their respective areas.

Audience

The best way to narrow a goal is to make sure that the audience is always considered. What a brainstorming group wants and what a user wants don't always correspond. The first thing to do is to accurately describe the site's audience and their reason for visiting the site. However, don't look for a generic Joe Enduser with a modem who happened upon your site by chance. It is unlikely such a user could be identified for most sites, and most users will probably have a particular goal in mind. First, think about what kind of people your end users are. Consider asking some basic questions about the site's users, such as these:

- Where are they located?
- How old are they?
- What is their gender?
- What language do they speak?
- How technically and Web proficient are they?
- Are the users disabled (sight, movement, and so on) in any manner?
- What kind of connection would they have to the Internet?
- What kind of computer would they use?
- What kind of browser would they probably use?

Next, consider what the users are doing at the site:

- How did they get to the site?
- What do they want to accomplish at the site?

- When will they visit the site?
- How long will they stay during a particular visit?
- From what page(s) will they leave the site?
- When will they return to the site, if ever?
- How often do they return?

While you might be able to describe the user from these questions, you should quickly determine that your site would probably not have one single type of user with a single goal. For most sites, there are many types of users, each with different characteristics and goals.

Stats Logs

If the site has been running for some time, you have a gold mine of information about your audience—your stats logs. Far too often designers don't really look at logs for anything other than basic trends such as number of page views. However, from looking at logs you should be able to determine useful information, such as the types of browsers commonly accessing the site, the general pattern of when and how visitors use the site, the current delivery and server requirements, and a variety of other valuable ideas. Of course, stats logs won't tell you much about user satisfaction and specific details of site usage.

User Profiling

The best way to understand users is to actually talk to them. If at all possible, you should interview users directly to resolve any questions you may have about their wants and characteristics. A survey may also be appropriate, but live interviews provide the possibility to explore ideas beyond predetermined questions. Unfortunately, interviewing or even surveying users can be very time consuming and will not account for every single type of user characteristic or desire. From user interviews and surveys or even from just thinking about generic users, you should attempt to create stereotypical but detailed profiles of common users.

Consider developing at least three named users. For most sites, the three stereotypical users should correspond roughly to an inexperienced user, a user who has Web experience but doesn't visit your site often, and a power user who understands the Web and may visit the site frequently. Most sites will have these classes of users, with the intermediate infrequent visitor most often being the largest group. Make sure to assign percentages to each of the generic groups so that you give each the appropriate weight. Now name each person. You may want to name each after a particular real user you interviewed, or use generic names like Bob Beginner, Irene Intermediate, and Paul Poweruser.

Now work up very specific profiles for each stereotypical user using the questions from the previous section. Try to make sure that the answers correspond roughly to the average answers for each group. So, if there were a few intermediate users interviewed who had fast connections, but most have slow connections, assume the more common case. Chapter 2 discussed the concept of general user characteristics versus individual traits in more detail.

Once your profiles for each generic site visitor are complete, you should begin to create visit scenarios. What exactly would Bob Beginner do when he visits your site? What are the tasks he wishes to perform? What is his goal? Scenario planning should help you focus on what each user will actually want to do. From this exercise, you may find that your goal statements are not in line with what the users are probably interested in doing. If so, you are still in the risk analysis whirlpool. Return to the initial step and modify the goal statement based on your new information.

Site Requirements

Based on the goals of the site and what the audience is like, the site's requirements should begin to present themselves. These requirements should be roughly broken up along visual, technical, content, and delivery requirements. To determine requirements, you might ask questions like these:

- What kind of content will be required?
- What kind of look should the site have?
- What types of programs will have to be built?
- How many servers will be required to service the site's visitors?
- What kind of restrictions will users place on the site with respect to bandwidth, screen-size, the browser, and so on?

Requirements will begin to show site costs and potential implementation problems. The requirements will suggest how many developers are required and show what content is lacking. If the requirements seem excessive in view of the potential gain, it is time to revisit the goal stage or question if the audience was accurately defined. The first three steps of the process may be repeated numerous times until a site plan or specification is thrown out of the whirlpool.

The Site Plan

Once a goal, audience, and site requirements have been discussed and documented, a formal site plan should be drawn up. The site plan should contain the following sections:

- **Short goal statement** This section would contain a brief discussion to explain the overall purpose of the site and its basic success measurements.
- **Detailed goal discussion** This section would discuss the site's goals in detail and provide measurable goals to verify the benefit of the site.

- **Audience discussion** This section would profile the users who would visit the site. The section would describe both audience characteristics and the tasks the audience would want to accomplish at the site.
- **Usage discussion** This section discusses the various task/visit scenarios for the site's users. Start first with how the user will arrive at the site and then follow the visit to its conclusion. This section may also include a discussion of usage measurements, such as number of downloads, page accesses per visit, form being filled out, and so on as they relate to the detailed goal discussion.
- **Content requirements** The content requirements section should provide a laundry list of all text, images, and other media required in the site. A matrix showing the required content, form, existence, and potential owner or creator is useful, as it shows how much content may be outstanding. A simple matrix is shown in Table 4-1.

Content Name	Description	Content Type	Content Format	Exists?	Owner
Butler Robot Press Release	Press release for new Butler 7 series robot that ran in *Robots Today*.	Text	Microsoft Word	Yes	Jennifer Tuggle
Software Agreement Form	Brief description of legal liability of using trial robot personality software	Text	Paper	Yes	John P. Lawyer
Handheld Super-computer Screen Shot	Picture of the new Demo Company Cray-9000 handheld palm size computer	Image	GIF	No	Pascal Wirth

Table 4-1. *Content Matrix*

Content Name	Description	Content Type	Content Format	Exists?	Owner
Welcome from President Message	Brief introduction letter from President to welcome user to site	Text	Microsoft Word	No	President's Executive Assistant

Table 4-2. *Content Matrix* (continued)

- **Technical requirements** This section should provide an overview of the types of technology the site will employ, such as HTML, JavaScript, CGI, Java, plug-ins, and so on. It should cover any technical constraints such as performance requirements, security requirements, multi-device or multi-platform considerations, and any other technical requirements that are related to the visitor's capabilities.
- **Visual requirements** The visual requirements section should outline basic considerations for interface design. The section should indicate in broad strokes how the site should relate to any existing marketing materials and provide an indication of user constraints for graphics and multimedia, such as screen size, color depth, bandwidth, and so on. The section may outline some specifics, such as organizational logo usage limitations, fonts required, or color use; however, many of the details of the site's visuals will be determined later in the development process.
- **Delivery requirements** This section should indicate the delivery requirements, particularly any hosting considerations. A basic discussion of how many users will visit the site, how many pages will be consumed on a typical day, and the size of a typical page should be included in this section. Even if these are just guesses, it is possible to provide a brief analysis of the server and bandwidth required to deliver the site.
- **Miscellaneous requirements** There may be other requirements that need to be detailed in the site plan, such as language requirements, legal issues, industry standards, and other similar considerations. They may not necessarily require their own separate discussion, but instead may be addressed throughout the other sections of the document.
- **Site structure diagram** This section should provide a site structure or flow diagram detailing the various sections within a site. Appropriate labels for

sections and general ideas for each section should be developed based on the various user scenarios explored in earlier project phases. Organization of the various sections of the site is important and may have to be refined over time. Often a site diagram will look something like the one shown in Figure 4-4.

- **Staffing** This section should detail the resources required to execute the site. Measurements can be in simple man-hours and should relate to each of the four staffing areas: content, technology, visual design, and management.
- **Time line** The time line should show how the project would proceed using the staffing estimates from the preceding section combined with the typical waterfall process outlined earlier in the chapter.
- **Budget** A budget is primarily determined from the staffing requirements and the delivery requirements. However, marketing costs or other issues such as content licensing could be addressed in the budget.

The actual organization and content of the site plan is up to the developer. Remember, the purpose of the plan is to communicate the site's goals to the various people working on the project and help guide the project towards a positive conclusion. Don't skip writing the plan even though it may seem daunting, as without such a document you can only develop a project in an evolutionary or JAD fashion. Furthermore, it will be nearly impossible to obtain any realistic bids from outside vendors on a Web site without a specification.

A finished plan doesn't allow you to immediately proceed to implementation. Once the specification is developed, it should be questioned one last time. The completed

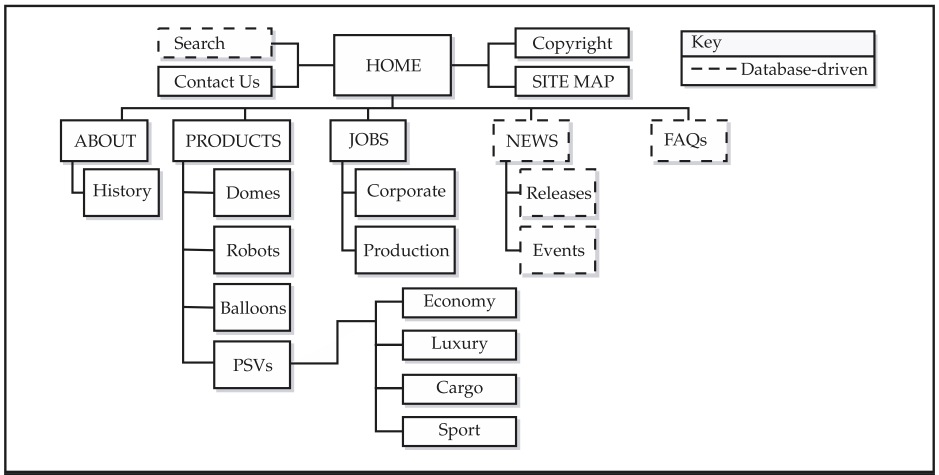


Figure 4-4. *Typical site diagram*

specification may reveal unrealistic estimates that will throw you back in the whirlpool of questioning initial goals or audience. If it survives, it may be time to actually continue the process and fall over the waterfall into the design and prototyping stage.

Design Phase Dissected

The design or prototyping stage is the most fun for most Web designers, as it starts to bring form to the project. During this phase, both technical and visual prototypes should be developed. However, before prototypes are built, consider collecting as much content as possible. The content itself will influence the site and help guide its form. If the content is written in a very serious tone but the visuals are fun and carefree, the site will seem very strange to the user. Seeing the content up front would allow the designer to integrate the design and content. Also, consider that content collection can be one of the slowest aspects of site design. Many participants in a Web project are quick to attend brainstorming meetings but are difficult to find once their content contributions are required. Lack of content is by far the biggest problem in Web projects. Deal with this potential problem early.

Suggestion: Always collect content as soon as possible.

Block Composites

Design should proceed top-down. Think first about how the user will enter the site and conclude with about how they will leave. In most cases, this means designing the home page first, followed by subsection pages, and finally form or content pages.

Rule: Visual design should proceed in a top-down fashion from home page to subsection pages and finally to content pages.

First consider creating page mockups on paper in a block form, as shown in Figure 4-5.

Block comps (or more commonly *wireframes*) allow designers to focus on the types of objects in the page and their organization without worrying too much about precise placement and detail of the layout itself. The block sectioning approach will also help the designer to consider making templates for pages, which will make it easier to implement them later on. Make sure to create your block comps within the constraints of a Web browser window. The influence of the browser's borders can be a significant factor. Once the home page block comp has been built, flesh out the other types of pages in the site in a similar fashion. Once a complete scenario has been detailed in this abstract sense, make sure that the path through the blocked screen is logical. If it is, move on to the next phase.

Screen and Paper Comps

The next phase of design is the paper or screen prototyping phase. In this phase, the designer can either sketch or create a digital composite that shows a much more

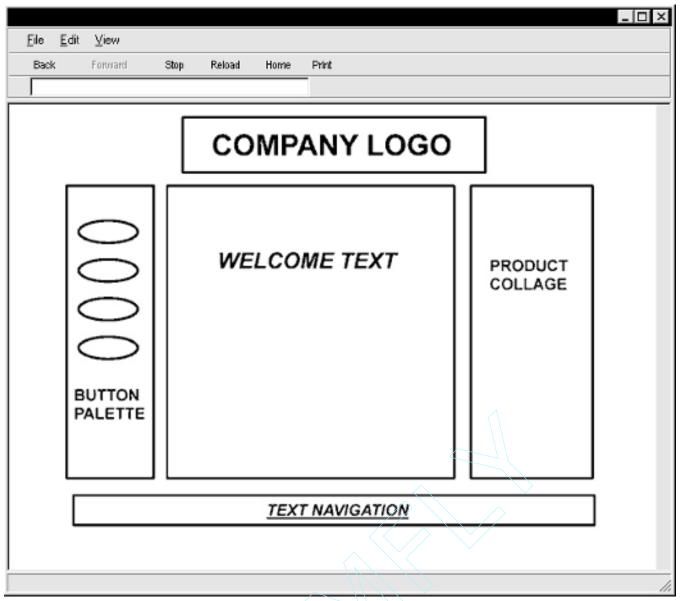


Figure 4-5. *Home page wireframe or block composite*

detailed visual example of a typical page in the site. Make sure that, whether you do the composite on paper or screen, a browser window is assumed and that screen dimensions are considered. A piece of paper with a browser window outline as used in the block comp stage can be used for sketches.

Suggestion: Always consider the bordering effect of the browser window when developing visual composites.

Sketch the various buttons, headings, and features within the page. Make sure to provide some indication of text in the page—either a form of "greeked" text or real content, if possible.

Note *Many designers appear to use only temporary "lorem ipsum" or greeking text within screen composites. This approach does bring focus to the designed page elements, but if real content is available—use it! This more closely simulates what the final result will be like.*

The comping stage provides the most room for creativity, but designers are warned to be creative within the constraints of what is possible on the Web and what visual requirements were presented in the design specification. Thinking about file size, color support, and browser capabilities may seem limiting, but doing so usually prevents the designer from coming up with a page that looks visually stunning but is nearly impossible to implement or download in a reasonable amount of time. In particular, resist the urge to become so artistic as to reinvent an organization's look in a Web site. Remember, the site plan will have spelled out visual requirements, including marketing constraints. The difficult balance between form, function, purpose, and content, as discussed in Chapter 1, should become readily apparent as designers grapple with satisfying their creative urges within the constraints of Web technology, user capabilities, and site requirements. A typical paper comp is shown in Figure 4-6.

In the case of a digital prototype, create a single image that shows the entire intended screen, including all buttons, images, and text. Save the image as a GIF or JPEG and load it into the Web browser to test how it would look within a typical environment. At this stage, resist the urge to fully implement your page design with HTML. You may end up having to scrap the design, and it would be wasteful to fully implement at this stage.

Once your paper or digital prototype is complete, it should be tested with users. Ask a few users to indicate which sections on the screen are clickable and what buttons

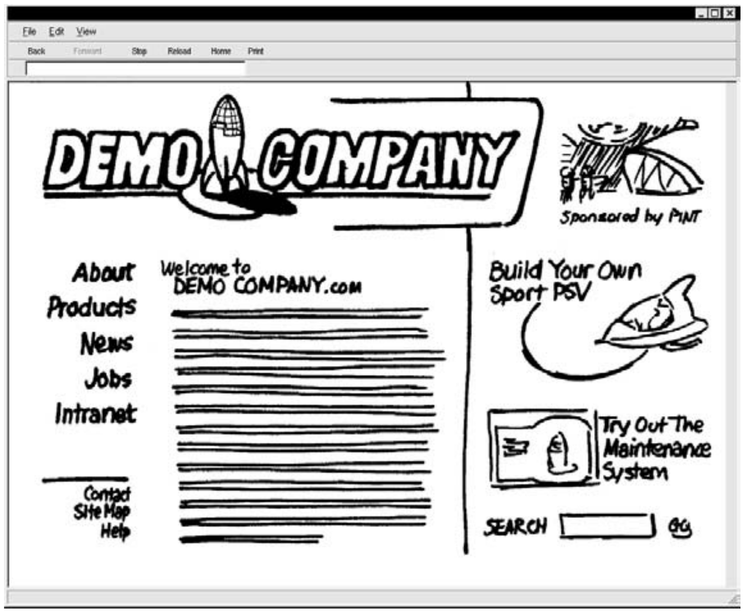


Figure 4-6. *Paper comp for Demo Company site home page*

they would select in order to accomplish a particular task. Make sure to show the prototype to more than one user, as individual taste may be a significant factor in prototype acceptance. If the user has too many negative comments about the page, consider starting over. During prototyping, you can't get too attached to your children, so to speak. If you do, the site will no longer be user focused, but developer focused. Remember the following design rule:

Rule: Don't marry your design prototypes. Listen to your users and refine your designs.

Once you come up with an acceptable home page design, continue the process with subpages and content pages. A typical subpage composite is shown in Figure 4-7.

In highly interactive sites, you may have to develop prototype pages for each step within a particular task, such as purchasing or download. Prototype pages for such steps may have to be more fully fleshed out and include form field labels and other details to be truly useful. A sample paper composite for a more interactive page is shown in Figure 4-8.

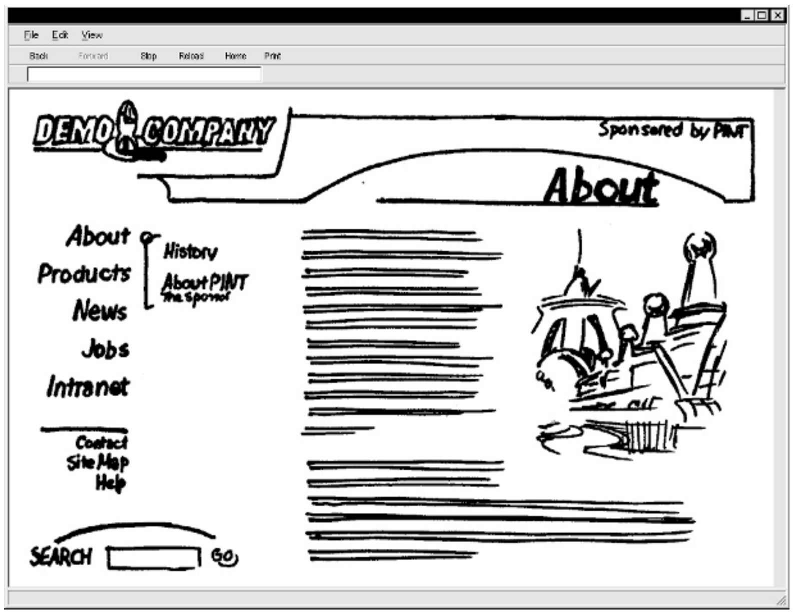


Figure 4-7. *Subpage paper composite for Demo Company*

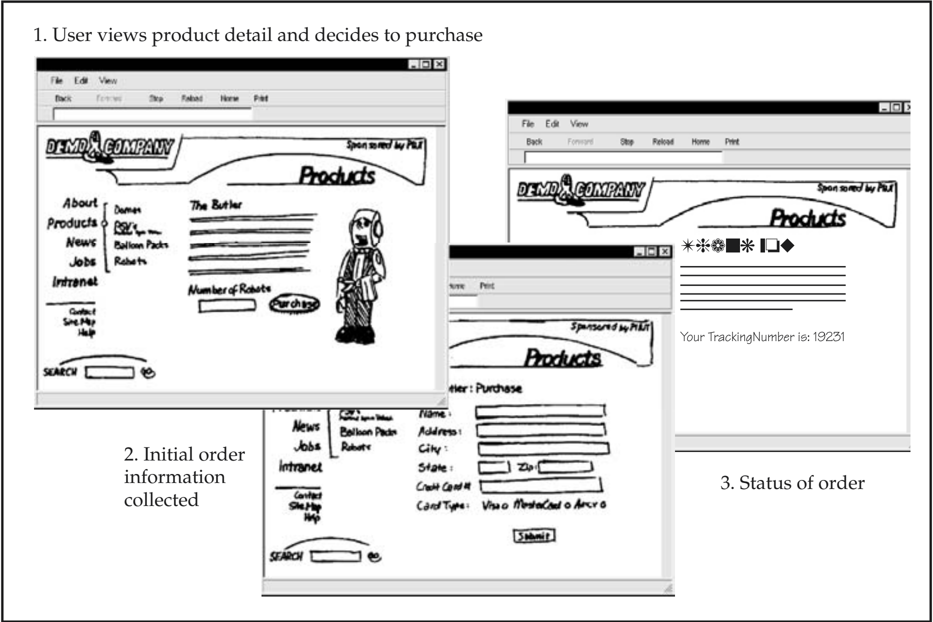


Figure 4-8. *E-commerce paper composite*

While not all sites will require technical prototypes, developers of highly interactive sites should consider not only interface prototypes but also working proof of concept prototypes, showing how technological aspects work, such as database query, personalization, e-commerce, and so on. Unfortunately, what tends to happen is that technical prototypes are not built until a nearly complete interface is put in place, which may result in a heavy amount of rework.

The Mock Site

After all design prototypes have been finalized, it is time to create what might be called the mock, or alpha, site. Implementation of the mock site starts first by cutting a digital comp into its pieces, assembling the pages using HTML, and, potentially, cascading style sheets. Try assembling the site with templates so that the entire site can be quickly assembled. However, do not put the content in place during this phase. Use greeking text on most pages unless real text is required for testing scenarios. Once the mock site is assembled, the site should be fully navigable—but with no content and only canned or basic interactivity.

It is important not to go through too much trouble implementing technical features that may change. For example, in an e-commerce site, you may want to make only one or two products purchasable. In that situation, it is a good idea to have a few users try the mock site. Observe if the site is easy to navigate and responsive. Have users attempt to complete real tasks with faked results in place. If the users have difficultly performing the tasks, you may have to consider scrapping the design and returning to a previous step in the development process. Generally, this won't happen unless the site was overdesigned or little user feedback was considered until that point.

Beta Site Implementation

Once the mock site is acceptable, it is time to actually implement the real site. Real content should be placed in pages, and back-end components and interactive elements should be integrated with the final visual design. Implementation and technology considerations are too numerous to discuss here and are presented individually in Chapters 11–17. While implementation would seem to be the most time-consuming aspect of a project, in reality, if all the components have been collected and prototypes built previous to this stage, the actual site implementation might occur relatively rapidly.

Testing

For most developers, testing is probably the least favorite aspect of the Web development process. After all the hard work of specification, design, and implementation, most people are ready to just launch the site. Resist the urge. Testing is key to a positive user takeaway value. Don't force your users to test your site after its release. If they encounter bugs with what is considered a production site, they won't be forgiving. Always remember the following design rule:

Rule: Sites always have bugs, so test your site well.

Unfortunately, testing on the Web is generally relegated to a quick look at the site using a few browsers and maybe checking the links in the site. Bugs will exist in Web sites, no matter what. Unfortunately, most developers consider that if the site looks right, it is right. Remember from Chapter 1 that site design doesn't just include visual design: you must test all the other aspects of site design as well, as expressed in the design rule presented here:

Rule: Testing should address all aspects of a site, including content, visuals, function, and purpose.

The next chapter will discuss evaluation and testing of sites in detail, particularly when looking at a completed site, but the basic aspects of Web testing are overviewed here.

Visual Acceptance Testing

Visual acceptance testing ensures the site looks the way it was intended to look. View each of the pages in the site and make sure that they are consistent in layout, color, and style. Look at the site under different browsers, resolutions, and viewing environments equivalent to those of a real user. Browse the site very quickly and see if the layouts jump slightly. Consider looking at the pages while squinting to notice abstract irregularities in layout. Visual acceptance testing may also require each page to be printed. Remember not to focus on print testing pages that are designed for online consumption.

Functionality Testing

Functionality testing and visual testing do overlap in the sense that the most basic function of a page is to simply render onscreen. However, most sites contain at least basic functions such as navigation. Make sure to check every link in a site and rectify any broken links. Broken links should be considered catastrophic functional errors. Make sure to test all interactive elements such as forms, shopping carts, search engines, and so on. Use both realistic test situations and extreme cases. Try to break your forms by providing obviously bad data such as typing in a search query that would not return a result or one that would return a very large number of matching pages. Remember: users won’t think and act as you do, so prepare for the unexpected.

Content Proofing

The content details of a site are very important. Make sure content is all in place and that grammar and word usage is consistent. Check details like product names, copyright dates, and trademarks—and always remember to check the spelling! Clients and users may often regard an entire site as being poor just on the basis of one small typo; the importance of this cannot be stressed enough. The best way to perform this test is to print each page and read literally every single word for accuracy.

System and Browser Compatibility Testing

Though system and browser restrictions should have been respected during development, you should verify this during testing. Make sure to browse the site with the same types of systems and browsers the site’s users will have. Unfortunately, it often seems that designers check compatibility on systems far more powerful than the typical user’s. The project plan should have detailed browser requirements, so make sure the site works under the specified browsers.

Delivery Testing

Check to make sure the site is delivered adequately. Try browsing the site under real user conditions. If the site was designed for modem users, set up a dial-up account to test delivery speed. To simulate site traffic, consider using testing software to create virtual users clicking on the site. This will simulate how the site will react under real conditions. Make sure that you test the site on the actual production server to be used or a system equivalent to it. Be careful not to underestimate delivery influences. The whole project may be derailed if this was not adequately thought about during specification. For further information on delivery conditions, see Chapter 17.

User Acceptance Testing

User acceptance testing should be performed after the site appears to work correctly. In software, this form of testing is often called *beta* testing. Let the users actually try the working site and comment on it one last time. Do not perform this type of testing until the more obvious bugs have been rectified.

Rule: User testing is the most important form of testing.

User testing is the most important form of testing because it most closely simulates real use. If problems are uncovered during this phase of testing, you may not be able to correct them right away. If the problems are not dramatic, you may still release the site and correct the problems later. However, if any significant issues are uncovered, it is wise to delay release until they can be corrected.

Release and Beyond

Once the site is ready to be released, don’t relax—you are not done. In fact, your work has just begun. It is now time to observe the site in action. Does the site meet user expectations? Were the site development goals satisfied? Are any small corrections required? The bottom line is that the site must live on. New features will be required. Upgrades to deal with technology changes are inevitable. Visual changes to meet marketing demands are very likely. The initial development signifies the start of a continual development process most call *maintenance*. Once over the waterfall, it is time to climb back to the top, as stated in the following design rule:

Rule: Site development is an ongoing process—plan, design, develop, release, repeat.

Welcome to the Real World

While the site development process appears to be a very straightforward cycle, it doesn’t always go so smoothly. There are just too many variables to account for in the real world. For example, consider the effects of building a site for another person such as boss or client. If someone else is paying for a site to be built, you may still need to indulge their desire, whether or not the requests make sense or satisfy user wants. Because of this possibility, make sure you attempt to persuade others that decisions should always be made with the user in mind. Try showing the benefits of design theories rather than preaching rules. Be prepared to show examples of your ideas that are fully fleshed out. However, accept that they often may be shot down.

Most Web projects tend to have political problems. Don't expect everyone to agree. Departments in a company will wrestle for control, often with battle lines being drawn between the marketing department and the technology groups. To stir up even more trouble, there may be numerous self-proclaimed Web experts nearby ready to give advice. Don't be surprised when someone's brother's friend turns out to be a Web "expert" who claims you can build the whole site with the latest Web development tool in one hour. The only way to combat political problems is to be patient and attempt to educate. Not everyone will understand the purpose of the site; without a clear specification in place, developers may find themselves in a precarious position open to attack from all sides.

Another challenge in building Web sites is dealing with the degree of change in a project. Quite often new stakeholders arrive during the middle of the project, new technologies are adopted during development, features are added or removed at a moment's notice, visuals are changed to conform to new branding, and even the focus of the project changes just before launch. The process model that we adopt will likely help us bring order to a project, but it won't solve every problem, particularly when the scope changes too much. If there is too much change, a project will get off track and you'll have to revisit aspects of development you had thought were finished.

Finally, always remember that the purpose of following a process model like the one discussed in this chapter is to minimize the problems that occur during a Web project. However, no process model will account for every real-world problem, particularly those involving people. Experience is the only teacher for dealing with many problems. Developers lacking experience in Web projects are always encouraged to roll with the punches and consider all obstacles as learning experiences.

Summary

Building a modern Web site can be challenging, so site builders should adopt a methodology or process model. This process model should help guide the development process, as well as minimize risk, manage complexity, and generally improve the end result. Software engineering process models such as the modified waterfall can be applied easily to most Web projects. However, when project management experience is lacking or there are no clear goal statements, a prototype-driven or joint application process should be employed. It will be difficult to plan for what is unknown, and, if the process can't be hammered down, it is probably best to try something quickly, fail, and learn from it.

While iterative prototype-based development would seem to easily fit with the organic nature of many sites, it can produce needless risk and result in building the wrong site numerous times before building the right one. Planning during the early stages of a site's development minimizes risk and should improve the end result. A design document that usually includes site goals, audience and task analysis, content requirements, site structure, technical requirements, and management considerations should always be developed. The design document guides the production of the Web site. During the design phase of site production, use block diagrams, paper mock-ups, storyboards, and even mock sites to reduce the likelihood of having to redesign the site later on. If a plan is well thought out and the design phase prototypes are built, implementation should proceed rapidly and require little rework. However, once it's finished, be careful not to rush the site online—adequate testing is required. Maintenance and continued vigilance will be required, or your finely crafted site will begin to degrade.

This page intentionally left blank.

Chapter 5

Evaluating Web Sites

Often, developers are faced with upgrading an existing Web site rather than starting from scratch. Being able to fully evaluate the execution of a Web site is an important skill that all developers should strive to master. Site evaluation is also a great way to learn from others. Looking at sites that are well executed may inspire designers, while evaluating those that are broken may show them how to avoid errors. Yet site evaluations are not always easy to conduct. Often, developers focus on what they are familiar with or focus only on surface aspects of sites, such as visual design. As in building a site, an evaluation of a site must focus not only on visuals but also on technology, content, purpose, and delivery. Even when keeping all aspects of Web design in mind, a developer looking at a site may not understand either the initial design considerations or the decisions made that result in what is being evaluated. In this sense, evaluators may have to act as archeologists and try to uncover deeper meaning from basic site characteristics.

The primary method for site evaluation we present in this chapter is often termed *expert evaluation*. The goal is to study a site as informed developers and try to find common execution and usability problems. However, the problem with this type of site evaluation is that developers may not think like users and may assume that things are usable when they are not. Expert evaluation is simply no substitute for real user interviews and testing. Yet don't quickly dismiss expert analysis in favor of usability studies. User testing does little to uncover execution flaws, so we should make sure that sites pass the execution part of our evaluation first before wasting valuable user testing time. Further, many common usability problems are easily observable and user testing simply verifies what a skilled developer may already know to be true through experience. Given these considerations, we will proceed with an overview of expert evaluation first, followed by a discussion of conducting user testing.

The Goals of Expert Evaluation

There are two goals when conducting expert evaluations of Web sites. The first is to uncover obvious execution flaws with sites, such as poor HTML markup, error prone JavaScript, broken links, and other problems (which should be caught during quality assurance but often are not). The second goal is to find obvious usability problems with a site before conducting user testing.

While the use of quality assurance tools and practical knowledge of the various aspects of the Web medium will help us find execution gaffes, usability problems can be more difficult to ferret out. We need to be mindful of how users think when conducting this part of testing. We need to be particularly careful when making assumptions about purpose, audience, creation method, and so on. If these assumptions are incorrect, the associated conclusions could be equally incorrect.

Conducting an Evaluation

When starting an evaluation, it is important to stop and record some basic information. For example, note the URL of the site you are to evaluate, the date, the time, the person conducting the evaluation, and the reason for the evaluation. When you begin the evaluation, you should block out some time to do the evaluation continuously; otherwise, your impressions could be adversely affected. Consider recording your end time to get an idea of how long it took to reach your conclusions. In general, the evaluation will be broken into the following steps:

1. First impression
2. Home page pretesting
3. Sub-page pretesting
4. Navigation pretesting
5. Task analysis
6. Execution Analysis
7. Final Impression

When we have finished with the evaluation, any required supplementary materials should be prepared, and an evaluation summary developed. Appendix B provides a sample form for conducting a site evaluation. Reading the following sections will help you understand the motivation for the various tests and how to conduct them.

First Impression

The first thing to do before you start the detailed evaluation is to stop and write down your first reaction to the site's home page. Just load the home page and look at it for at most five to ten seconds, and write down whatever comes to mind. Ideally, you will not be too familiar with the site, so the first impression will not be tainted. (Be sure to clear your browser's temporary files and cookies to make certain that your results are not skewed by the site already being cached.) If you are very familiar with the site, you might want to get a few other people, show them the site, and ask what they think on a scale from 1 to 5 (where 1 is a negative feeling and 5 is positive). The point here is to gauge a user's initial feeling for a site—remember, people aren't always rational. Unfortunately, a first impression is only just that if it is truly the first time you are looking at a site. Don't discount this part of the test. Even though a first impression may be an emotional reaction heavily influenced by visuals or environment considerations, record it and try to understand what causes your feeling. If users coming to a site have a very positive or negative first impression, it could certainly affect their desire to go further.

Home Page Pretests

The first few pretests conducted will give you a basic sense of the usability of the home page. Some of the pretests will require you to make some logical assumptions that you will later verify to show usability of the site, so don't start using the site yet or you'll spoil this part of the evaluation. Just keep the home page onscreen and your hands off the mouse and keyboard.

Identity Pretest

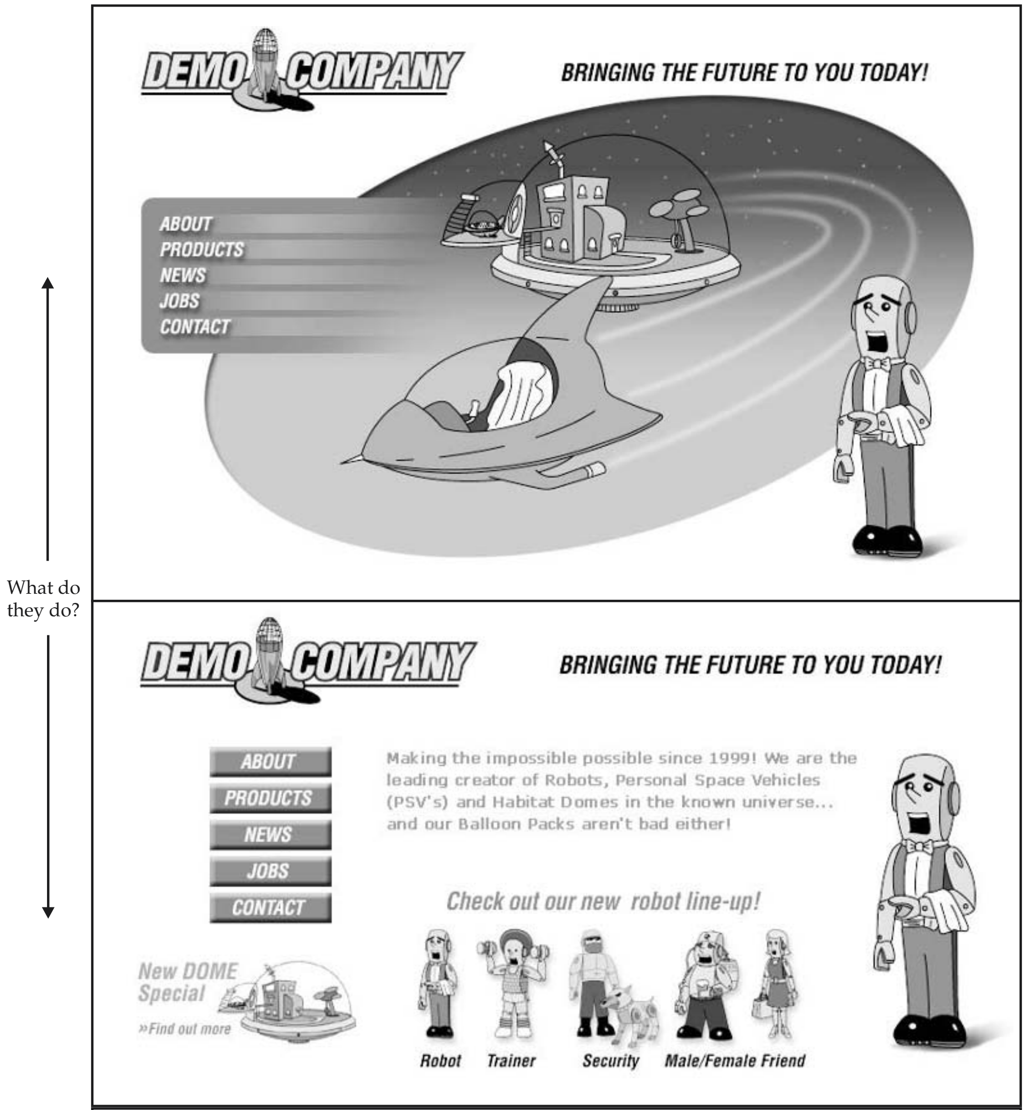
The first pretest to be conducted could be called the identity test. To conduct this test, look at the home page for between 30 seconds to a minute, and see if you can figure out the organization's name, the topic of the home page, and any sense of what the site is about. It would seem obvious that a site should clearly communicate its goals and purpose right away, but often that just isn't the case. Consider the two home pages in Figure 5-1—which passes the home page identity test for you?

Now ask yourself what users are supposed to accomplish at the site. More important—who is the site actually built for? For some sites—particularly those that you may not have much involvement in—performing a site evaluation may be much like an archaeologist looking at an ancient civilization's ruins. The purpose, use, and users of a particular aspect of a Web site will be almost as difficult to discern by a site evaluator as the significance of a few stones from a larger structure by an archaeologist.

Navigation Pretests

The next and probably the most telling is the navigation pretest. In this test, before you use the site, look at the home page and attempt to guess which areas of the screen are clickable. You may consider printing the page and circling the hot spots, conducting what is called a *paper test*. However, given that many pages may not be designed for printing or will remove navigation features in print, it is best just to do a screen test and run your finger, not the mouse, around the screen trying to determine if something is clickable or not. Once you have evaluated the whole page, go back and check your intuition. You will probably find that some clickable areas of the page do not obviously look like they are for purposes of navigation, while other things that *look* clickable actually aren't. Common reasons for failure include inconsistent color usage such as using blue text for labels and logos, removing underlines on links, and trying to make images and supporting materials link together. Note the number of believed links and actual links, determine an accuracy ratio, and record any notable problems for your final report.

The second navigation pretest requires determining the purpose of each clickable zone on the page. Once the links have been identified, record each and write a brief statement about what will happen when the link is pressed. Once finished, check your record by visiting each link and noting whether your guess was correct or not. Surprisingly, this test fails quite often because of poor labels. Often, failed link labels use a metaphor, jargon, or acronym, so make sure that your wording is plain and simple.



Making the impossible possible since 1999! We are the leading creator of Robots, Personal Space Vehicles (PSV's) and Habitat Domes in the known universe... and our Balloon Packs aren't bad either!

Figure 5-1. *Looks can be deceiving*

Once finished with these basic tests, you might want to scan link labels for style and consistency. Make sure that the labels are of similar length, wording, and style, both textually and visually. Observe the rules of the page for what is clickable and what is not, and note any inconsistency in visual style in clickable regions, regardless of whether any such region passed the initial clickable pretest.

Sub-Page Pretests

The primary sub-pages of the site—namely, those that are directly accessible from the home page—should be tested using the same pretests described in the previous two sections. However, for the identity pretest, focus more on the purpose of the page than on the organization. The navigation pretests should proceed normally. While this may seem like a lot of work for an average size site, it should proceed rather quickly if the sub-pages follow a consistent design and navigation pattern. If they do vary greatly, you are probably facing a site that has a high degree of design and navigation inconsistency and deserves significant analysis.

Site Navigation Testing

Once the first layers of the site have been examined, it is time to perform simple tests to probe the quality of the global site navigation. Good sites will provide consistent, well-executed navigation and should provide alternative navigation schemes, such as site maps, indexes, and search engines. First, look to make sure that placement of navigation is consistent from page to page. Subtle shifting may occur, so try browsing the site extremely fast and notice whether the menu items bounce or jump position slightly from page to page. Even this minor variation can break the perceived stability of a site. Next, look to see how robust the navigation is and whether multiple forms of site navigation are supported. Numerous navigation execution questions should be asked during this phase. Is the current location clearly indicated with labels or link path indicators? Does the site have text links at the bottoms of pages? Is alternative text used for graphical navigation buttons? Does the site require excessive scrolling? Are back-to-top links used on longer pages? Does the site have a map or index? The questionnaire in Appendix B presents many of the questions you should be asking during the navigation analysis phase.

One form of navigation that deserves special attention, if present, is the search facility. Very often, search is poorly implemented in a site, despite the fact that more and more users are coming to rely on it. Chapter 9 presents a thorough discussion of how search should be implemented in a site; but for now, focus on how the search is accessed, how it deals with errors, and how both positive and negative results are presented. Search facilities should be clearly marked and easily accessible from every page. A well-implemented search should correct errors or at least clearly indicate them when they occur. Once a positive query is returned, the results should be easy to navigate and refine. All these issues are covered in the sample evaluation; but if you evaluate sites on your own, make sure to enter nonsense queries and “extreme positive” queries, like the organization name, in the search field, to see how the extreme cases are handled.

Task Analysis

The testing so far has concentrated on general navigation of a site, but the goal of navigation is to help a user accomplish some task. Generally, on the Web, users are doing one of three general tasks:

1. Reading
2. Looking for something
3. Performing some interaction

The third task covers user activities like interacting with menus, filling out forms, or other mechanical tasks. Our testing should make sure that the site supports all three of these general task groups. Once we have verified that, we should consider the specific tasks unique to a particular site.

Testing Readability

When thinking about reading Web content, you have to consider both when and how the user will read the content. A user may read content immediately, may print it to read offline, or may bookmark it to read or print at a later date. Web content should be readable both onscreen and on paper.

Testing printing is easy: just print each page in the site. Be careful, though; some pages may purposely not be designed for printing. Also, you may have special print buttons or Adobe Acrobat files for printing. If this is the case, make sure to note the approach and whether it is effective.

Testing the screen readability of content is a little more difficult. Of course, reading content is the best test, but it tends to take a long time. You will almost certainly find, as you perform this test, that content is too long or complex to be easily read onscreen. Even when content is written for screen use, page layout and contrast may make it difficult to read. One way to test page layouts and contrast is to perform what the author dubs the "fuzzy eye" test. In this test, squint and look at the page. If you can still discern the general sense of the page structure easily, the layout and contrast is probably adequate; if you cannot, the items may be too close together or contrast may not be strong enough.

Testing Findability

Of course, information is only useful if site visitors can find it. In order to test the findability of information in a site, you first need to have at least some familiarity with the content in the site before attempting to find an item likely to be there. The simplest findability test would be to look for something required in just about any site—for example, contact information. Once a generic item has been determined, try to find the information from an arbitrary point in the site. You may find that even this test requires numerous clicks once beyond the home page. You can also try the same test using the site's secondary navigation facilities, such as the site map and search facility.

The other findability tests are similar to the simple one just described, but they require that you find a particular item that is very specific to the site. For example, if products are sold, try to find the price of a particular product, the cheapest product, and the most expensive product. If the organization is a corporation, try to find information about the management team or, if it is publicly traded, its current stock price or last reported revenue figure. There are many possible information tasks, and you may want to record not only whether the task was successful or not, but also the time it takes or the number of clicks required to find something—as well as your feelings about the ease of use and adequacy of results.

Testing Interactivity

The final task-related test concerns the various interactive features of the site. This testing is primarily related to filling out forms for performing tasks such as ordering products, making contacts, creating memberships, and so on. Each primary feature of the site should be tested in three ways: correct usage, extreme negative, and extreme positive. Correct usage means following the steps—filling out a form and so on—to buy a product in the basic, obviously correct manner. You may find that it is difficult to figure out what to do during this test. If so, make sure to note down frustrations. Extreme negative and extreme positive tests make mistakes on purpose during interactive tasks. In extreme negative testing, obviously false or blank answers are provided to see if the site handles these properly. Extreme positive testing goes in the opposite direction and tests for out-of-range values and things that would be obviously beyond the capacity of the site. Well-designed sites should limit errors, so, ideally, interactive tests will cause frustration rather than raise execution issues. Unfortunately, given the state of Web development procedures (as discussed in Chapter 4), many execution errors may exist in tested sites. We will discuss a few things to look for in the next section.

Execution Analysis

Execution testing focuses on trying to make sure the site is built correctly. Execution includes issues with content, visuals, technology, and delivery. For example, with content, you might look to see if site content is up-to-date or if there are spelling and grammar errors in pages. Technical execution would focus on whether the site follows standards for HTML, CSS, XML, and other technologies. Visual execution would be concerned with image quality and file size. Delivery would be focused on speed and server capacity. The next few sections detail a few of the things to keep in mind as you evaluate each area, with the Appendix B checklist providing a set of specific questions to try to answer.

Content Execution

The quality of a site is heavily influenced by the freshness and quality of the content presented. A site's content should be appropriate in quantity—not too much that it is difficult to find appropriate information easily, but not so little that the user is left wanting more. The content should also be up-to-date and accurate. Execution issues, such as spelling, grammar, and tone, should also be well considered. Last, the details of the site should be very carefully examined. Truly, with Web sites, the devil is in the details. Copyright dates, trademarks, product names, and very small formatting errors are often glaring to the user and may ruin an otherwise excellent experience.

A good way to evaluate content is to do a careful screen and paper walk-through. Printing pages and going over each one very carefully is probably the best way to find typos and consistency issues. However, many Web maintenance tools and even page editors can be used to spell-check pages. When looking for details, it is tough to spot everything; fortunately, some Web site maintenance tools can be used to evaluate consistency of terminology through the use of custom rules that look for the inclusion of certain key phrases.

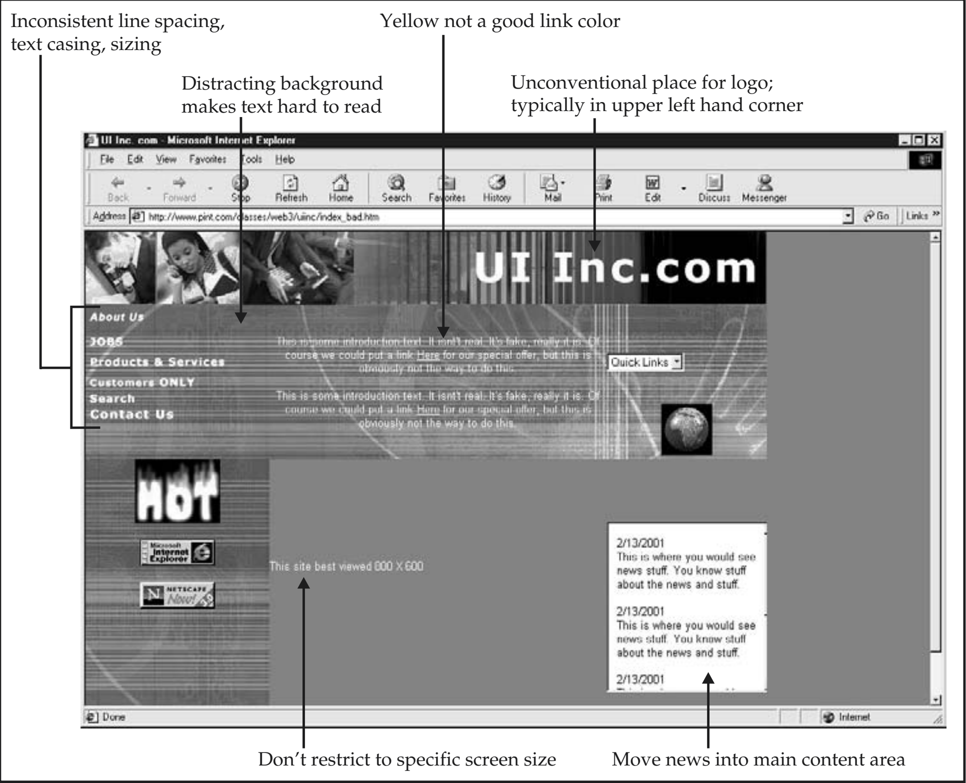
Visual Execution

Evaluating the look and feel of a site can be difficult because doing so is, to a great degree, a matter of personal taste. However, execution of images and layout should be evaluated regardless of your personal take on a site's aesthetics. Images may not be used properly or optimized correctly. There may be color problems in the site, font sizing issues, and page layout problems. In many cases, the page layout may not even fit the screen resolution or print correctly. Pay particular attention to tests of the site under less than ideal conditions, such as lower resolution. In many cases, a site layout will completely fall apart when images are turned off or font sizes modified. When doing the visual portion of a site evaluation, it is important to print out a screen capture of the evaluated page, as it may change over time. Screen printouts can be marked up to draw attention to problem areas as well as interesting features. Figure 5-2 shows an example of a marked-up page with visual and navigation execution notes.

Technical Execution

Web design relies heavily on technology, ranging from simple markup languages to complex programming approaches. When evaluating a site you have full access to, it is possible not only to look at client-side technologies, such as HTML, but also to examine server-side technologies, such as CGI programs or databases. Unfortunately, when examining sites externally, you may be limited to looking only at technology easily viewed at the browser or the effect of technology executed on a server. For some evaluators, it may be appropriate to call in a professional programmer to evaluate the quality of examined code, as glaring errors may escape those who know CGI or JavaScript only just enough to use provided scripts. We overview a few of the more common technologies here for evaluation and leave the rest for Appendix B.

HTML/XHTML Because HTML serves as the bedrock of a Web site, particular attention should be paid to the accuracy and quality of HTML. With the rise of XHTML, use of the doctype indicator and strict compliance are becoming particularly critical. Compliance with the various HTML or XHTML standards should be examined by validating key pages in the site. Online validators, such as http://validator.w3.org, can be used, but readers may


Inconsistent line spacing, text casing, sizing
Yellow not a good link color
Distracting background makes text hard to read
Unconventional place for logo; typically in upper left hand corner
UI Inc. com - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Messenger
Address http://www.pint.com/classes/web3/uiinc/index_bad.htm
Go Links
UI Inc.com
About Us
JOBS
Products & Services
Customers ONLY
Search
Contact Us
Quick Links
HOT
Done
Internet
Don't restrict to specific screen size
Move news into main content area

This is some introduction text. It isn't real. It's fake, really it is. Of course we could put a link Here for our special offer, but this is obviously not the way to do this.

This is some introduction text. It isn't real. It's fake, really it is. Of course we could put a link Here for our special offer, but this is obviously not the way to do this.

This site best viewed 800 X 600

2/13/2001
This is where you would see news stuff. You know stuff about the news and stuff.

2/13/2001
This is where you would see news stuff. You know stuff about the news and stuff.

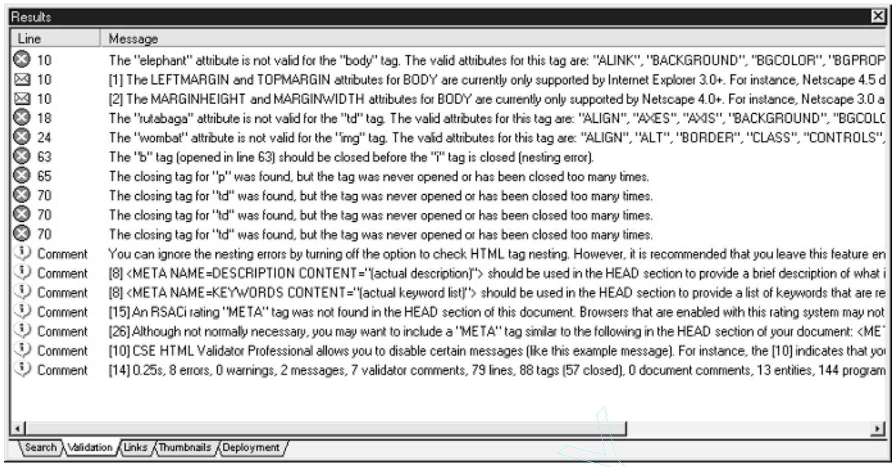
2/13/2001

Figure 5-2. *Printed page marked up*

find stand alone validation tools like CSE Validator (http://www.htmlvalidator.com) to be superior. Figure 5-3 shows this validator in action.

Proprietary tag usage or trick HTML should be carefully noted. Inspection **<meta>** tags, comments, and other small signs such as consistent page formats should be noted to help determine how HTML was created—such as with a tool or by hand. In some cases, telltale signs like indentation patterns of markup may indicate creation by a particular HTML editor; but if it is possible to directly query the developer, ask which tools were used and what standards were followed if any.

CSS Cascading Style Sheets are rapidly becoming an important technology for presenting Web pages. CSS use provides a major benefit in allowing separation of document structure from presentation. However, unless the site uses external style sheets, this benefit is reduced. Document-wide style sheets or inline styles are adequate,

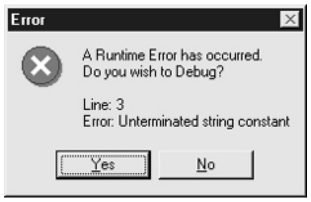


```
Results
Line        Message
10          The "elephant" attribute is not valid for the "body" tag. The valid attributes for this tag are: "ALINK", "BACKGROUND", "BGCOLOR", "BGPROP
10          [1] The LEFTMARGIN and TOPMARGIN attributes for BODY are currently only supported by Internet Explorer 3.0+. For instance, Netscape 4.5 d
10          [2] The MARGINHEIGHT and MARGINWIDTH attributes for BODY are currently only supported by Netscape 4.0+. For instance, Netscape 3.0 a
18          The "rutabaga" attribute is not valid for the "td" tag. The valid attributes for this tag are: "ALIGN", "AXES", "AXIS", "BACKGROUND", "BGCOLC
24          The "wombat" attribute is not valid for the "img" tag. The valid attributes for this tag are: "ALIGN", "ALT", "BORDER", "CLASS", "CONTROLS",
63          The "b" tag (opened in line 63) should be closed before the "i" tag is closed (nesting error).
65          The closing tag for "p" was found, but the tag was never opened or has been closed too many times.
70          The closing tag for "td" was found, but the tag was never opened or has been closed too many times.
70          The closing tag for "td" was found, but the tag was never opened or has been closed too many times.
70          The closing tag for "td" was found, but the tag was never opened or has been closed too many times.
Comment     You can ignore the nesting errors by turning off the option to check HTML tag nesting. However, it is recommended that you leave this feature en
Comment     [8] <META NAME=DESCRIPTION CONTENT="(actual description)"> should be used in the HEAD section to provide a brief description of what i
Comment     [8] <META NAME=KEYWORDS CONTENT="(actual keyword list)"> should be used in the HEAD section to provide a list of keywords that are re
Comment     [15] An RSACi rating "META" tag was not found in the HEAD section of this document. Browsers that are enabled with this rating system may not
Comment     [26] Although not normally necessary, you may want to include a "META" tag similar to the following in the HEAD section of your document: <ME
Comment     [10] CSE HTML Validator Professional allows you to disable certain messages (like this example message). For instance, the [10] indicates that yo
Comment     [14] 0.25s, 8 errors, 0 warnings, 2 messages, 7 validator comments, 79 lines, 88 tags (57 closed), 0 document comments, 13 entities, 144 program
```

Figure 5-3. *Example of HTML validation*

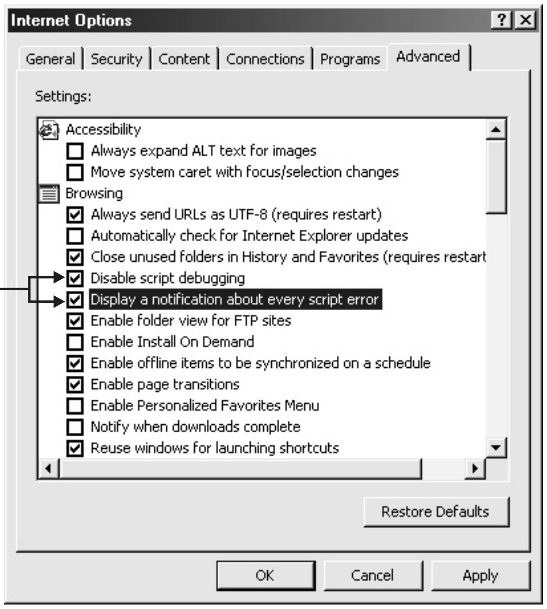
but their use should be considered less than ideal. Regardless of the method of including style rules, extreme care must be taken with CSS because of all the browser bugs and rendering differences. Compliance with the CSS1 and CSS2 standards may not be as important as making sure the various CSS properties work under common browsers. However, a CSS checker, such as http://jigsaw.w3.org/css-validator/, should be used. Close attention should also be paid to the types of rules used and whether or not there is any problem with browsers that do not support CSS. Testing with an older browser or with the CSS facilities turned off should be performed.

JavaScript JavaScript is a very important part of many Web pages, but far too often it is not used in a reliable manner. Well-executed JavaScript-laden pages will employ the **<noscript>** tag to address scripting being turned off, and may even restrict usage without script enabled. Scripts also should be able to address browser incompatibilities and should not throw error messages like the one shown here:



```
Error
A Runtime Error has occurred.
Do you wish to Debug?

Line: 3
Error: Unterminated string constant
```

Fortunately for Web site visitors, most browsers are shipped with the default to turn off JavaScript error notifications, since otherwise you would probably see a great number of them. Set your browser's preferences to show errors, as shown here in Internet Explorer.



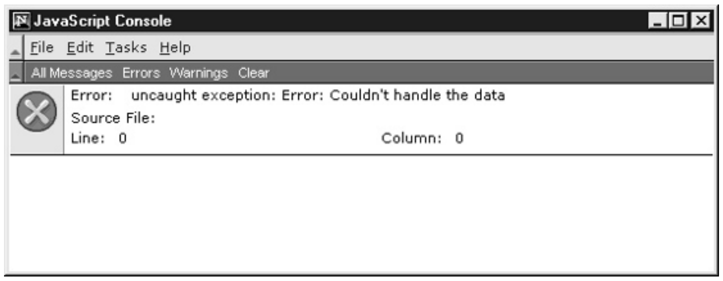
Settings:

Accessibility
- [ ] Always expand ALT text for images
- [ ] Move system caret with focus/selection changes

Browsing
- [x] Always send URLs as UTF-8 (requires restart)
- [ ] Automatically check for Internet Explorer updates
- [x] Close unused folders in History and Favorites (requires restart
- [x] Disable script debugging
- [x] Display a notification about every script error
- [x] Enable folder view for FTP sites
- [ ] Enable Install On Demand
- [x] Enable offline items to be synchronized on a schedule
- [x] Enable page transitions
- [ ] Enable Personalized Favorites Menu
- [ ] Notify when downloads complete
- [x] Reuse windows for launching shortcuts

To show errors, make sure the Disable Script Debugging check box is unchecked and Display Error Notification is checked.

In Netscape, you should check the JavaScript console for the error message shown here:



```
Error:   uncaught exception: Error: Couldn't handle the data
Source File:
Line:  0                                   Column:  0
```

Cookies For many, the use of cookies is an invasion of personal privacy. The reality is that cookies are very useful to get around programming limitations caused mainly by HTTP protocol limitations. However, regardless of your personal take on cookies, it is important to know whether a site uses cookies and what they are used for. Some sites may even issue multiple cookies per visit, each with a different purpose. Careful inspection of cookie data can yield valuable clues to how a site works. If cookies are used, it is important to verify the site still works with cookies off. Also, if cookies are used, a statement indicating what they are used for should be available on the site.

Browser Support Probably the most well-known aspect of site testing is browser support. Many site testing protocols simply advise designers to test in as many browsers as possible. The reality is that you should attempt to create a matrix of the various browsers and perform the technology and layout tests within each browser individually. Oddly, you may find that there are subtle rendering differences in each browser, as well as numerous bugs. A large matrix showing all the different versions of each browser and operating system is the best way to conduct a browser test. Unfortunately, you may find that there are literally dozens of versions of just the 4.*x* generation of Netscape. Because of the difficulty of testing so many combinations, you may want to focus on those browsers that are known to use your site. In some cases, such as with an intranet, the browser being used may be obvious; but before guessing what browsers a site's users commonly use, consider accessing the log files to make sure.

Delivery Execution

How the site is delivered is extremely important to understanding the site's usability. Users appreciate fast downloads, but, as will be discussed in Chapter 17, speed of delivery is often influenced by many factors beyond the size of files being delivered. It is important to understand the server resources used to deliver a site, including both hardware and software used. It is also important to understand how the site is hosted. How the site eventually connects to the Internet can impact performance greatly. Using even simple network tools like "ping," it is possible to determine the responsiveness of a server. Many operating systems provide this tool; for example, under Windows, access the DOS prompt and type **ping** and a host name. If you typed **ping www.webdesignref.com**, you might see something like this:

```
C:\WINDOWS>ping www.webdesignref.com

Pinging www.webdesignref.com [66.45.42.235] with 32 bytes of data:

Reply from 66.45.42.235: bytes=32 time=32ms TTL=114
Reply from 66.45.42.235: bytes=32 time=66ms TTL=114
Reply from 66.45.42.235: bytes=32 time=27ms TTL=114
Reply from 66.45.42.235: bytes=32 time=95ms TTL=114

Ping statistics for 66.45.42.235:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 27ms, Maximum = 95ms, Average = 55ms
```

The round-trip time of data can be used to get a general sense of the responsiveness of the server. It is also possible to acquire other server and network information using tools like WHOIS, traceroute, nslookup, and others. On Windows, most of these tools are included in the operating system, can be found in the public domain, or are nicely packaged in network tools like WS_Ping ProPack (http://www.ipswitch.com/).

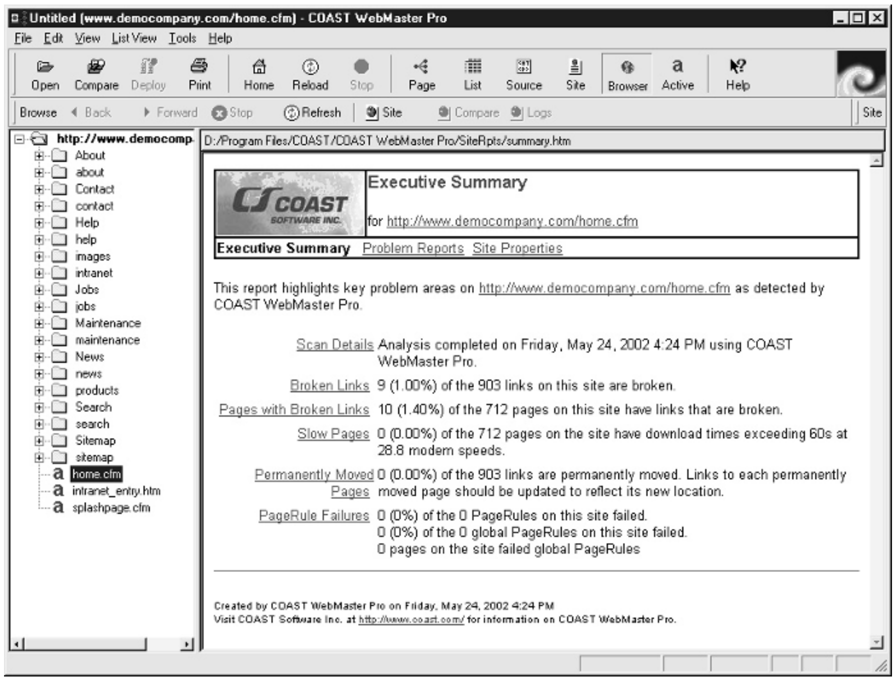
After server and network issues, the size of the pages delivered should be considered. Most site analysis tools will identify pages that are considered large. You can set the threshold for what is considered large, byte-wise, in most of the programs, but some consider anything over 30–50K (including any graphics in the page) as a large page, despite the rising popularity of faster Internet access. Theoretical download times under a variety of line speeds can also be determined with a site analysis tool, and most Web page editors like Dreamweaver even provide facilities to determine page weight and download speed. However, do not rely solely on theoretical times; test the site under actual conditions, if possible. Since network conditions are always changing site delivery, test results may vary greatly from moment to moment.

The Final Question

Now that you have evaluated many aspects of a site, consider what you would give the site as a final score. You don't have to be very scientific about your final rating. Given how much you know now about the site, do you think it is a great site or not? Were you able to accomplish the tests easily? Would you take away a positive, neutral, or negative feeling about the site? Consider listing a few of the reasons that made you skew one way or another.

Evaluation Reports

After finishing your evaluation, you should put together a report summarizing your findings. Make sure to illustrate your findings with as many frame grabs and diagrams as possible. Also, try to provide as many specific details as possible, as well as indications of where the errors are in the site and how they might be fixed. Complete reports should include a detailed analysis of a site, including the number of pages, the page weights, broken links, technology usage, and so on. Because of the tedious nature of compiling such information, we leave this part of the evaluation to tools. Consider using a maintenance or quality assurance tool to analyze the basic characteristics of the site. Quality maintenance tools such as Coast Webmaster (http://www.coast.com) can produce high-quality reports like the one shown in Figure 5-4. However, do not substitute tool use for a real expert evaluation because tools will miss many usability and execution errors.


Untitled (www.democompany.com/home.cfm) - COAST WebMaster Pro
File Edit View List View Tools Help
Open Compare Deploy Print Home Reload Stop Page List Source Site Browser Active Help
Browse Back Forward Stop Refresh Site Compare Logs Site
http://www.democomp
About
about
Contact
contact
Help
help
images
intranet
Jobs
jobs
Maintenance
maintenance
News
news
products
Search
search
Sitemap
sitemap
home.cfm
intranet_entry.htm
splashpage.cfm
D:/Program Files/COAST/COAST WebMaster Pro/SiteRpts/summary.htm
COAST SOFTWARE INC.

Executive Summary

for http://www.democompany.com/home.cfm

Executive Summary Problem Reports Site Properties

This report highlights key problem areas on http://www.democompany.com/home.cfm as detected by COAST WebMaster Pro.

Scan Details	Analysis completed on Friday, May 24, 2002 4:24 PM using COAST WebMaster Pro.
Broken Links	9 (1.00%) of the 903 links on this site are broken.
Pages with Broken Links	10 (1.40%) of the 712 pages on this site have links that are broken.
Slow Pages	0 (0.00%) of the 712 pages on the site have download times exceeding 60s at 28.8 modem speeds.
Permanently Moved Pages	0 (0.00%) of the 903 links are permanently moved. Links to each permanently moved page should be updated to reflect its new location.
PageRule Failures	0 (0%) of the 0 PageRules on this site failed. 0 (0%) of the 0 global PageRules on this site failed. 0 pages on the site failed global PageRules

Created by COAST WebMaster Pro on Friday, May 24, 2002 4:24 PM
Visit COAST Software Inc. at http://www.coast.com/ for information on COAST WebMaster Pro.

Figure 5-4. *Site Quality Report from a maintenance tool*

User Testing

While the evaluation process just described is useful to uncover many types of site problems, it is important not to limit evaluations just to inspection. Developers may focus on certain things and completely miss problems commonly encountered by users. Further, this form of evaluation does not adequately reflect how users actually use a site.

Looking at log files can provide valuable insight into how a site is used. Log files will show who is looking at a site (by IP address or domain name, mostly), what pages users commonly look at, when they look at these pages, the paths users take through a site, the links followed to get to a site, and even what kind of browsers are being used. The log file really does show if content is popular and may provide a great deal of information related to site usability. For example, a tremendous number of users leaving the site from a certain page may indicate a problem. Log files can be used to verify assumptions or even show places to look for problems.

While log files provide a great deal of useful information, they really say very little about a user's feelings about a site. An invaluable way to evaluate a site is to watch how users actually use a site and try to solicit feedback from them. Conducting a user test can be difficult. Be careful to focus more on what users do and not on what the say. Users typically don't want to look stupid and will often indicate that they understand something when they don't.

Rule: Pay attention more to what users do than to what they say.

The best way to deal with this problem is not to let users know that they are taking a test; you might even try to casually watch them without their knowledge. If you ask users to take a usability test, you may find that they pay more attention or try harder to figure things out than they might usually. The assumption almost seems to be that test administrators will be pleased at how proficient they are. At the opposite end of spectrum, on occasion testers will purposefully look for errors. In either case, it should be evident that testing conditions may not always be the same as user conditions.

A very important aspect of testing is making sure not to get too involved. For example, if you ask users to evaluate a site, don't guide them through it. If you co-pilot the users' browsing sessions, they will uncover only what you want them to and maybe not use the site as they might normally. If you talk too much, showing off the features of the site, you may not give users a chance to say what they think. User testing can be very difficult for site designers who want to put their work in the best light possible, and they may be very unwilling to listen to user criticisms.

Suggestion: Consider having a person not involved in the site design process conduct a user test.

You can certainly be very scientific about user testing: using two-way mirrors, recording mouse travel and keystrokes, and even monitoring pauses or mistakes made by the user during a typical task. Some might go so far as to watch facial expressions or even monitor the blood pressure of the test subject. However, the end result is often really the most important aspect of the test. Remember that, in the final analysis, probably the only real important things to users are whether they were successful in their mission and enjoyed the visit. This does not mean that the study of usability lacks reasonably measurable characteristics; it just suggests that, as imperfect creatures, humans may not always act logically and may even quickly forget the difficulty of performing a task if there is a wonderful reward at the end. Readers interested in understanding more about user testing and usability, particularly the theory and practice of conducting usability tests, should visit http://www.useit.com and http://www.usableweb.com.

Summary

Site evaluations serve both to provide quality assurance and to increase the skills and knowledge of developers. This chapter provided an overview of designer-directed evaluation, while focusing on execution and usability. The tips provided here in conjunction with the detailed checklist presented in Appendix B should uncover many of the common problems in Web sites. However, users may uncover more, and user evaluations should always be performed if possible because, in the end, the acceptability of the site will be determined by the users. However, do not discount developer evaluation, since it makes no sense to have users evaluate a site that is obviously built incorrectly or that exhibits known usability problems.

This page intentionally left blank.

The Complete Reference

Web Design

Part II

Site Organization and Navigation

This page intentionally left blank.

Chapter 6

Site Types and Architectures

Just as there are many types of software—from games to business applications—there are many types of Web sites. Sites can be grouped generally in categories like intranet or extranet sites, as well as specific-purpose sites like portals, entertainment sites, or personal home pages. Each type of site will have different design constraints related to the site's purpose. Organizing the site appropriately will help the site achieve its purpose. Numerous site structures—from simple linear organizations to complex mixed hierarchies—exist. Conventions, as well as heuristics from cognitive science and traditional GUI conventions, provide some clues as to which structures work well. However, the structure of a well-designed site isn't always apparent to the user—nor should it be.

Site Types

We begin by breaking sites into various groupings to understand the specific requirements of each group. Obviously there are numerous ways to categorize Web sites. Possible groupings include audience, level of interactivity, frequency of change, size, type of technology used, visual style applied, and of course the purpose of the site. The following three general categories of Web sites are universally accepted: *public Web sites*, *extranets*, and *intranets*.

Definition: A public Web site, an Internet Web site, an external Web site, or simply a Web site is one that is not explicitly restricted to a particular class of users.

An *external Web site* is, in a sense, a public place available to anyone on the Internet at large to visit. Not every user in the world may want to visit the site—the site shouldn't be designed for such a wide range of users—but there is no set limitation as to who can visit the site. At the opposite end of the spectrum would be an intranet Web site, generally called simply an intranet. An intranet site is generally very private, and is often available only to users on a particular private network.

Definition: An intranet Web site is a site that is private to a particular organization, generally run within a private network rather than on the Internet at large.

In between these extremes—an external Web site and intranet—would be a semiprivate site. An extranet is the most common example of a semiprivate site. An example of an extranet would be a site catering to company partners or resellers.

Definition: An extranet site is a Web site that is available to a limited class of users, but is available via the public Internet.

The major difference between the three basic site categorizations is audience. Public Web sites are completely open, while intranets and extranets are more exclusive. The more private the site, the greater understanding the designer will have about its potential users. As mentioned numerous times up to this point, understanding a site's users is crucial when designing a site. Consider that for a private intranet, a designer may actually be able to physically meet each and every potential user of the site. The designer may know the capabilities all the users, from their sophistication as computer users to the equipment or browser they use. On the opposite end of the spectrum is the public Web site. Designers of public sites often know relatively little about their users. They may rarely get to interact with their users directly and often will have little knowledge about the range of user capabilities. The design considerations will vary dramatically between the general Web sites, as illustrated in the following table:

	Intranets	Extranets	Public Sites
Info About Users	High	Medium	Low
Capacity Planning	Possible	Usually possible	Difficult to impossible
Bandwidth	High	Varies	Varies greatly
Ability to Set Technology	Yes	Sometimes	Rarely

This grouping of sites is the most generic partitioning by audience. We could go further and talk about sites geared for basic demographics like age groups (children, teens, adults, or senior-citizens) or gender (male or female). We could even further talk about specific characteristics such as ethnicity, socio-economic status, political orientation, and so on. However, these groupings begin to cross over too much into purpose issues and veer away from general characteristics common to all sites, so let's continue the discussion with more general groupings.

Grouping by Interactivity

Another way to classify sites is by how interactive they are. Many sites are not particularly interactive, but consist primarily of static content that a user may browse or search through. Such sites are often dubbed *static* sites because the user is unable to alter the site in a direct manner.

Definition: A static site is one where content is relatively fixed, and users are unable to affect the look or scope of the data they view. In short, the visitor has minimal ability to interact with the site's content other than choosing the order in which to view content.

Accessing a static site is like reading a paper magazine. A user can choose to flip back and forth between pages and read articles in a different order, but the presentation is relatively rigid. There is really no ability to do anything with the content of a static site other than read it onscreen, print it out to read on paper, or copy chunks of content for use somewhere else.

On the other hand many sites, particularly community related sites, allow users to contribute or modify content to some degree—such sites might be dubbed *interactive*. The degree of interaction the user may have with the site's content may range from the simple ability to comment to the creation of content either with or without further editing by the site's owner or other users.

Definition: An interactive site is one where the users of the site are able to interact directly with the content on the site or with other users of the site.

Of course to some degree, all sites have some interactivity in that users can choose how they want to browse content. However, truly interactive sites allow users to manipulate the content itself, and in some cases even add their own content. A site that allows a user to post technical support questions for other users to view would be considered interactive, while a site that only allows users to browse preexisting answers to questions would be considered static.

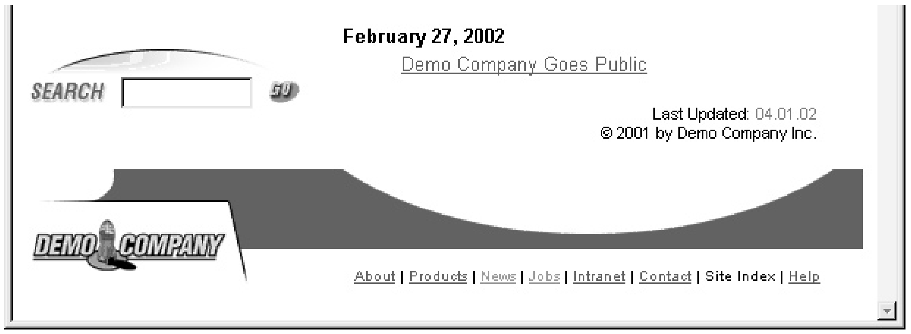
Grouping by Frequency of Change

Another dimension of site categorization is the frequency that content changes. Sites that never change might again be dubbed *static*; those that do change could be thought of as *dynamic*. "Dynamic" is used in this context in terms of page content and not page generation, which is a separate issue we'll discuss shortly. Most sites aren't absolutely static; changes are usually made to pages gradually over time. The more frequently the site changes, the more dynamic it could be thought to be. Content may change on a regular basis, like daily, weekly, and so on, or it may change in a less scheduled manner.

Sites may also change on a continual basis. For example, a personalized site is one that changes per visitor, often in response to either current or past visitor activities. A common example is an e-commerce site that may offer "specials" or suggest products based upon previous buying habits of the visitor or even the buying habits of other visitors to the page being accessed. Other examples of personalized pages are those for portals (my.yahoo.com) that provide so-called "my" style pages configured by users to suit their own particular interests.

Definition: A personalized site is one where content is directly geared towards a particular user, and the user generally can explicitly determine the content, look, or technology contained within a page.

Indicating to users when site content has changed is very important in dynamic sites. Often a small statement with the date of the last change is put on a page to show how fresh its content is. Often this is just a text line that is modified by the page maintainer or is output from a small JavaScript. Users may also look to copyright information on a page or other apparently trivial items to get a clue about page freshness.


February 27, 2002

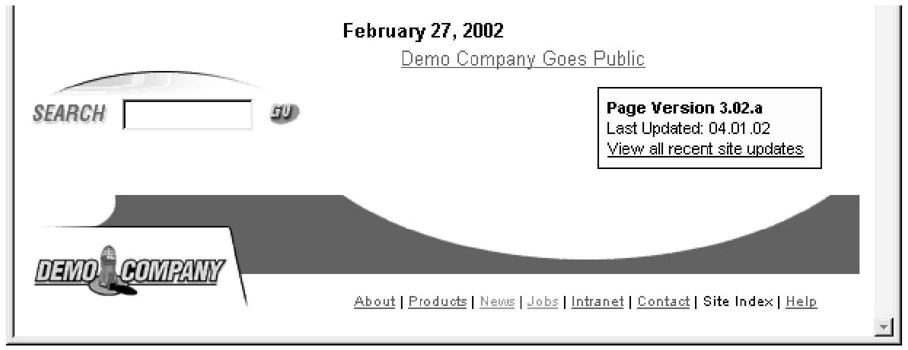
Demo Company Goes Public

Last Updated: 04.01.02

© 2001 by Demo Company Inc.

SEARCH
GO
DEMO COMPANY
About | Products | News | Jobs | Intranet | Contact | Site Index | Help

While modification dates may vary from page to page, sites may also exhibit more consistent update statements beyond just copyright information. For example, some sites include statements about the current day, week, month, or year in the page design to indicate how often the page is changing.


February 27, 2002

Demo Company Goes Public

Page Version 3.02.a

Last Updated: 04.01.02

View all recent site updates

SEARCH
GO
DEMO COMPANY
About | Products | News | Jobs | Intranet | Contact | Site Index | Help

Grouping by Time of Page Creation

When considering time in site designs, it's important to clearly state when pages are actually built for visitors. In many cases pages are static, in that they are created ahead of time for the user and change very little. In other cases pages may be built at a scheduled time as their content is created or altered. Finally a page may be generated just as a user requests them, often termed a *dynamically generated* page.

Definition: A dynamically generated page is created at request or view time for the user.

There are numerous benefits to dynamically generated pages. First the content can be customized to suit what the user may be looking for. Search result pages are a common form of dynamic page. Dynamic pages can also be created to take into account browsing conditions or technology restrictions. For example, a static site has only one form of presentation that all users must deal with, while a dynamic site may have multiple forms optimized for different browsers or bandwidth levels. The downside is that dynamically generated sites are significantly more complicated to create and often are very server intensive, as each page must be generated for users when they visit. Yet another benefit is that dynamically generated pages are often easier to maintain. For example, in dynamic sites, "page look" can be maintained in common templates, footers can be added to all pages, navigation held in common files, and so on.

Dynamically generated sites often use a database to store site content. In these sites, pages are constructed from content merged into page templates at request time to create the final page for delivery. Given the complexity and potential serving costs, pages should be dynamically generated only if necessary. For example, even if pages are stored in a database, unless they change per visit, they should not be uniquely created for each visitor. Doing so would be plainly wasteful, and caching such page content in the form of static pages outputted from the database will result in a much more responsive and scalable site. Conversely, though, if content does change often or per visitor, there is no value to trying to pre-generate pages, so don't try. More information on site serving considerations will be given in Chapter 17. A comparison between static and dynamically generated sites is shown in Figure 6-1.

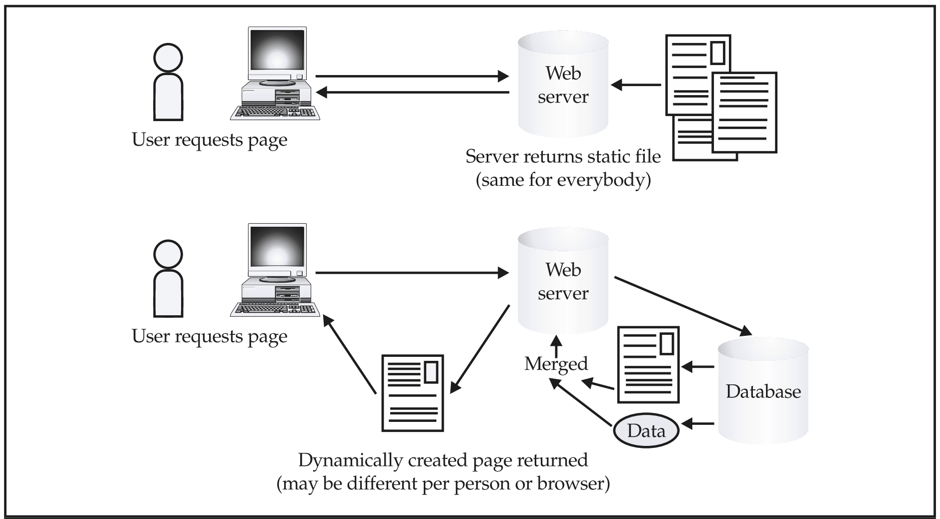

Web server
User requests page
Server returns static file (same for everybody)
Web server
User requests page
Merged
Database
Data
Dynamically created page returned (may be different per person or browser)

Figure 6-1. *A comparision between statically and dynamically generated sites*

Grouping by Size

Another possible consideration when grouping Web sites is to consider their size. Size doesn't mean much in many sites, particularly when they are generated from database-stored content; however, regardless of this fact, the number of pages continues to be used as a classification metric. While there are no precise breakdowns of what constitutes a large site or a small site, the following groupings seem useful:

< 10 pages	Very small site
10–100 pages	Small site
100–1000 pages	Mid-size site
1000–10,000 pages	Large site
> 10,000 pages	Very large site

The value of these groupings is that they reflect the effort and people involved. Very small and small sites are generally tended by very few people and often have limited technological considerations. Mid-size and large sites may be maintained by a small group of people and have more complex technology behind them. Finally, large and very large sites may have a considerable number of individuals maintaining them, given their complex technical and delivery requirements. Given the growing volume of Web-based documents, we could certainly shift the previous groupings and add breakdowns for sites in the hundreds of thousands and millions of documents range.

Grouping by Technology Usage

Grouping sites by their size or degree of interactivity often directly intersects with technical considerations. In general, we might consider technology in sites when we plot how document-centric or application-centric a site is. Recalling the discussion from Chapter 1, we see that many sites are not much more than simply brochures and thus are very *document-centric*. Other sites, such as online banking or shopping sites, might provide a great deal of interactivity, making them more *application-centric*. The continuum of sites grouped by their general technology use, from simple documents to full-blown Web-based software applications, is shown in Figure 6-2.

We can further classify sites by the specific type of technology used, such as HTML with presentation determined by tables or XHTML with presentation using style sheets. In particular, we might want to group sites that embrace standard technologies defined by the W3C versus those that continue to embrace older browser or vendor-specific technologies. However, further grouping sites by whether they use Java, ASP, ColdFusion, XML, or some other technology at this point adds little value to our discussion. We will return to these ideas later on when we look at the execution of sites.

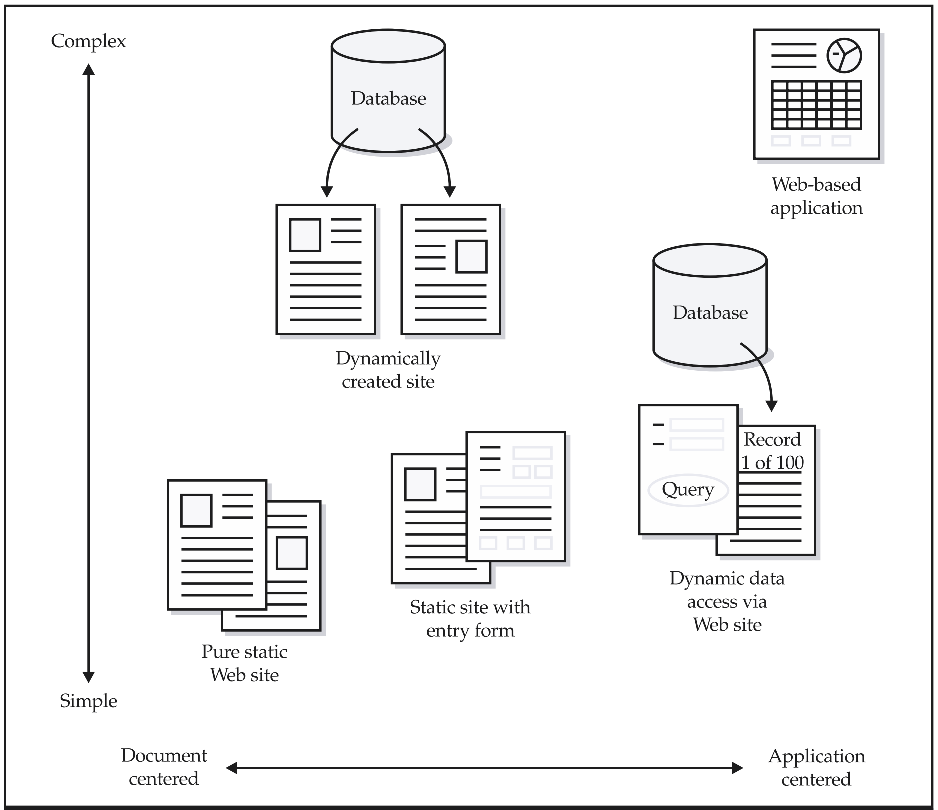


Figure 6-2. *Technical range of web sites*

Grouping by Look

We may also group sites by the visual design style used. Simple grouping might discuss how visual the site is. Does it rely on images or not? Are colors used? However, we probably don't have to be so simple—instead, we can categorize sites in four visual groups:

- **Text-focused** Focusing on text content with limited design and graphics
- **GUI style** Following graphical interface conventions
- **Metaphorical** Providing a rich interface often based upon a metaphor from the real world
- **Experimental** Breaking conventions and presenting content and site navigation in a new or surprising manner

Examples of each style are presented in Figure 6-3.

Grouping by Purpose

As we have seen there are numerous ways to characterize sites, including their audience, their frequency of change, their technology, or their look. However, these characterizations may seem too abstract at times. There are numerous genres of sites that use these abstract forms. We need to focus more on the reason for a site, namely its purpose. We'll focus in this discussion only on public Web sites, but characterizations of private intranet sites could also be made. One very general way to categorize public sites would be as *commercial*, *entertainment*, *informational*, *navigational*, *artistic*, or *personal*. The general goals, audience, and features of each type of site vary dramatically. Because of this, be cautious not to apply the same design philosophy to each form.

Commercial Sites

Commercial sites are those sites that are built primarily to support the business of some organization. Generally, the primary audience of a commercial site is made up of potential and current customers of the organization. A secondary audience often includes potential and current investors, potential employees, and interested third parties such as the news media or even competitors. Given such an audience mix, common purposes for commercial sites include

- **Basic information distribution** The site is used to disseminate information about products and services provided by the organization. Other basic information provided generally includes how to contact the firm via methods other than the Web.
- **Support** Portions of the site might be built to provide information to help existing customers effectively use products or services provided by the organization.

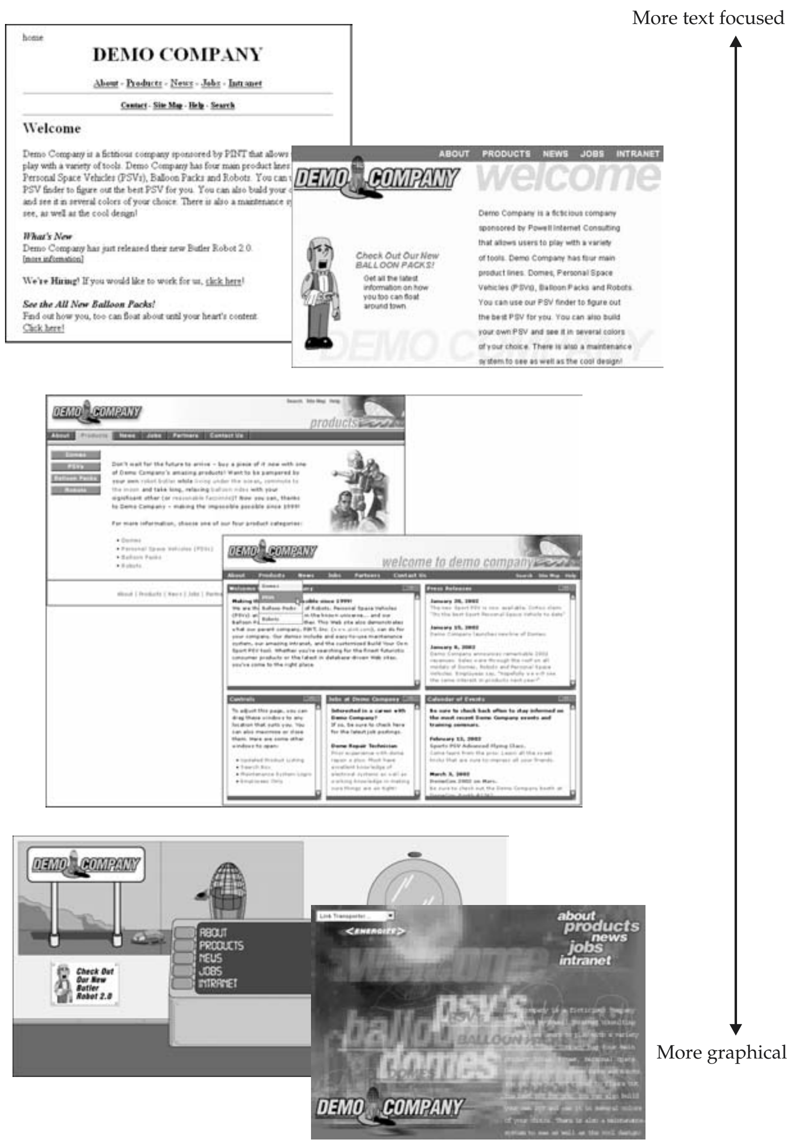


Figure 6-3. *Visual range of Web sites*

- **Investor relations** A public company or one seeking outside investment might build a site or a section within a site to disseminate information about the current financial situation of the company, as well as future opportunities for investment.
- **Public relations** Many firms use their Web sites to distribute information to various news gathering organizations, as well as to provide general goodwill information to the community.
- **Employee recruiting** A Web site is often used to post information about employment opportunities and benefits of working for a company.
- **E-commerce** A growing number of commercial Web sites allow a visitor, whether an end consumer or a business partner like a reseller, to conduct business directly on the Web site. Common facilities supported by e-commerce sites include transactions like ordering, order status inquiries, and account balance inquiries. Therefore, we might break out e-commerce-focused commercial sites. Such sites are usually termed *transactional* sites.

Look at all the potential purposes of a commercial site, and you'll see that the following premise follows directly:

Premise: The overriding purpose of any commercial site is to serve the user in a way that will benefit the company either directly or indirectly.

Given this premise, consider that the purpose of information dissemination is to try to get people to purchase a product or service from the company. Whether the method is a direct approach trying to persuade the user or an indirect approach of providing helpful information intended to foster a trusting relationship between the organization and the potential customer, the purpose is always the same—try to encourage a business transaction to take place.

Informational

Informational sites are different from commercial sites in that their main purpose is information distribution. Informational sites often have to do with government, education, news, nonprofit organizations, religious groups, or various social-oriented organizations. While the sites may be driven by some commercial factors, the primary purpose is to inform rather than cause a transaction to happen. Understanding the audience mix of an informational site is difficult, since it depends highly on the type of information being provided. About all that can be said is that the audience of the site is someone who has an interest in or is required to view the information provided.

The purposes of informational sites vary dramatically. A site at a university for a class might help educate visitors on a certain topic like American History. An informational site for some particular religious, social, or political group might have a primary purpose of convincing people to join or donate something to the organization.

News sites might have a primary purpose of informing people of current events in a helpful manner so that people rely on the resource enough to sell their attention to various advertisers. A government site might have a purpose of informing citizens of various law changes, convincing them to join civil or military service, or even getting them to vote a particular way. The crossover between commercial and informational sites can be great, but always remember that the main difference is that commercial sites are much more economic-driven than informational sites. Informational sites may be built to meet design criteria that may not make fiscal sense. A commercial site always has an underlying goal of trying to increase the profits of the firm, and its purpose is often more predictable.

Entertainment

Entertainment sites are generally commercial, but they have special considerations. The purpose of an entertainment site is simply to entertain the site's visitors—in some sense they are selling entertainment. In other words, they are trying to sell an enjoyable experience. While commercial sites such as e-commerce sites do want the site visitor to have a positive or even entertaining experience, entertainment is really a secondary objective. While a site selling clothes might have a jungle explorer theme and entertain the visitor with tales of visiting far-off lands, the bottom line is that the experience is to help sell clothes. If the clothes don't sell, the site doesn't work. In the case of an entertainment site, the purpose is to sell the experience itself.

Creating an entertaining experience—whether it be visiting a Web site, playing a video game, or watching a movie—isn't something that is easily engineered. Keeping the viewer occupied and happy can be difficult and isn't always as formulaic a task as people might believe. For example, Hollywood continually struggles to understand why some blockbuster movies bomb while an unknown independent movie succeeds. Novelty is about the only thing that seems to continually sell. If a story is too much like something a person has experienced before, it often seems boring or formulaic. Web sites that are built to entertain are often required to break with convention to be successful.

Premise: Entertainment sites may find novelty or surprise in design more useful than structure or consistency.

Navigational

A navigational site is one whose focus is on helping people find their way on the Internet. Oftentimes these sites are called *portals*, since the sites serve as major hubs pointing to other destinations.

Definition: A portal is a site that is generally a primary starting point for a user's online journey and serves to help people find information. Portals often attempt to provide as much information and serve as many tasks for the user as possible in order to encourage them stay or to at least continually revisit the site.

Navigational sites would also include search engines or site directories, which, coincidentally, are often the backbone of many portal sites.

Community

A community site is one whose purpose is to create a central location for members or a particular community to congregate and interact. Visitors come to the site, which is often very informational in nature, not just to find content that is interesting to them but also to interact with other like-minded individuals. Community sites are very interactive and are often dynamically generated and personalized. The content of a community site varies as greatly as with that of an informational site. Some communities may be very general in their membership, focusing on a broad demographic, such as women. Other communities may be very focused and target a select group of individuals, such as Asian American college students in southern California.

Community sites and informational or commercial sites often cross over. The main distinction between pure information or commercial sites and community sites is simply the ability for a site's visitors to interact with each other. If, over time, the ability to interact with other site visitors becomes commonplace on commercial and informational site, the special distinction of community sites will be lost.

Definition: A community site is any site that allows easy interaction between site visitors and serves as a meeting area for site visitors rather than simply a viewing area for visitors to view canned content.

Artistic

An artistic site is a site that is purely the expression of the individual or artist. The purpose of the site would be to inspire, enlighten, or entertain its viewers. In some cases, the site may simply be the product of the artist just trying to express his or her feelings. The site's creator may not really care what the viewer thinks of the site. As long as the site makes the artist happy, it is successful. Artistic sites may be user driven only in that they encourage thought and may go out of their way to avoid convention or logic.

Premise: The design of artistic sites may purposefully defy common Web conventions.

Personal

Like an artistic site, a personal site—often called a *personal home page* or just a *home page*—is often an expression of its creator. Personal pages may be built to inform friends or family, or they might just be built as a way to learn a new skill, like knowing HTML. Some personal pages appear to be literal shrines to their creators in some vain attempt to become famous through the Web. Other personal pages are mere résumé sites, useful to show to potential employers during job searches. A new form of personal page serving as an online journal or diary, dubbed a *blog*, has also become popular. Like artistic sites, personal sites will not be discussed to any major degree in this book, because often their main purpose is simply to make their creators happy. However, you would do well to consider that many personal sites could certainly improve their look, structure, usability, or technology.

The social implications of personal Web sites warrant a short aside before we move on. In some sense, the purpose of the personal page is to personify the individual on the Web. Unfortunately, this can be a rather dangerous concept. While it would seem obvious not to post your credit card number, social security number, bank account numbers, and so on to your personal page, the degree of details posted on many personal pages is frightening. Many people post intimate details of their lives, from pictures of friends and family to literally their daily diary. While such online exhibitionism might seem harmless, consider the possibility of stalking or profiling. Users should consider that stating all your likes and dislikes online in the form of a personal page is a direct marketer's dream. Profiles are easy to build from such information and may result in highly targeted and potentially intrusive junk email. Far worse might be the possibility for stalking or even identity theft from personal Web site-related information. Just remember that posting a personal Web page isn't too different from posting information on a local bulletin board in a town square. You never know who is going to look at the information and what they might do with it.

Site Structure

Given the type of site and other information, we can begin to apply structure to it. We find that there are two structural aspects to any Web site—logical structure and physical structure. A logical structure will describe documents that are related to other documents. The logical structure defines the links between documents. However, the logical location of documents within a site may not relate to the actual physical location of a document. A physical structure describes where a document actually lives, showing, for example, the document's directory path on a Web server or its location in a database.

Premise: A Web site's logical structure is more important to a user than its physical structure.

Users generally don't care where information comes from as long as they can find it. A user doesn't need to know what disk drives contain what data and how you have decided to organize your file tree. For example, a particular file might live in a deep directory on a file system, with a path like D:\WebSite\DemoCompany\Assets\Product\RobotButler\index.htm. However, from a user perspective, the URL might appear as http://www.democompany.com/RobotButler/. Resist the urge to expose paths to users. As the maintainer of the site, you will have to have explicit knowledge of the site's physical structure, but a user should not have to.

Rule: Do not expose physical site file structure, if possible.

The benefit of not showing real paths should be clear. By abstracting away paths, you are free to change the location of files freely as long as they map to the appropriate URL known to users. Fortunately, all modern Web servers support mapping facilities to create virtual paths, so there is no requirement to directly mimic your logical structure in a physical file system.

Rule: A site's logical document structure does not have to map to directly match physical structure.

Note *From a programming point of view, think of your site's URLs as your public interface. Every URL exposed is a potential address to access your site that will have to be maintained. If you are able to avoid exposing all URLs, using anything from frames to dynamic pages, you increase your ability to change the implementation of the site underneath.*

Site Organization Models

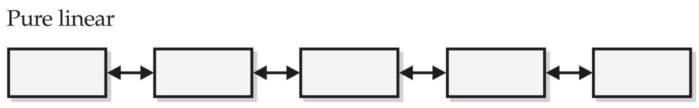
There are four main organizational forms used in Web sites*: linear, grid, hierarchy,* and *web***.** Variations on some of the schemas are common, as are combinations of each within a larger site. Choosing the correct site organization is important in making a site usable. For example, an online sales pitch would benefit from a linear form where slide two follows slide one. In some sense, the user is almost forced to see the content in the order the designer wants. If the presentation were organized in another fashion, such as a tree form, it might encourage users to access slides out of order, possibly reducing the impact of the sales pitch. Other information, such as answers to technical support questions, might be better suited to a non-sequential access form, because forcing the user to wade through pages of needless information would be extremely frustrating. The goal is to pick the most appropriate organization form for the content, so complex content can be made clear.

Linear

A *linear* form is the most familiar of all site structures because traditional print media tends to follow this style of organization. For example, books are generally written so that one page follows another in a linear order. Presenting information in a linear fashion is often very useful when discussing a step-by-step procedure or completing a process such as checkout in an e-commerce site, but there are times when supplementary information may be required. Linear forms can be modified slightly to provide more flexibility, but will eventually result in a grid, hierarchical, or pure web form when extended too much.

Pure Linear

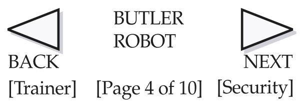
A *pure linear* organization facilitates an orderly progression through a body of information, as shown by the illustration here.



On the Web, this form might be good for a presentation like a "slide show" to give new visitors an overview of a company and its products. By using a controlled sequential organization like a linear form, the designer can ensure that the user receives the information in the intended order.

The linear style of organization provides a great deal of predictability because the designer knows exactly where the user will go next. Because of this knowledge, it may be possible to *preload* or *prefetch* the next bit of information to improve perceived performance of the site. For example, while the user is reading the information on one screen, the images for the next screen can be loaded into the browser's cache. When the user advances to the next screen, the page is loaded from the cache, giving the user the illusion that the page downloads very quickly. Preloading is not a viable solution unless the user's next path can be anticipated, as is the case with a linear organization.

Because there is really no choice but to move forward or back, a user may find a linear form to be very restrictive. Because of this, it is often important to let a user know how far they are in a linear structure, and what is previous and what is behind the page being viewed. Indicating that a user is on a page in a series could be as simple as putting a label on the page, like "Page *X* of *Y*" where *X* is the current page number and *Y* is the total number of pages.

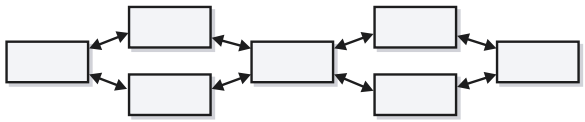


Note *A pure forward linear form can be difficult to implement on a Web site because of the browser's backtrack feature, so it is generally assumed that all linear forms are bidirectional unless the site is programmed to act otherwise.*

Linear with Alternatives

While a linear organization is useful to present information in a predetermined order, it may provide little room for the user to interact with the information. A *linear with alternatives* organization simulates interactivity by providing two or more ways to leave a page, which eventually ends up pointing the user back to another page within the sequence, as illustrated here.

Linear with alternatives

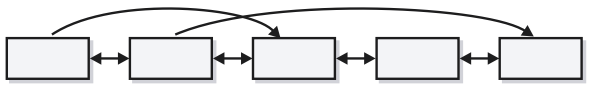


The uses of this form are numerous. Imagine a quiz site that prompts the user for a Yes or No answer to a question on each page and then advances the user to the next page based on the answer. Though it might appear to the user that there is some back-end technology at work, in reality the two tracks are already established, and the user is just presented with an illusion of interactivity. A health care site might use a general health quiz to attract people's interest. The quiz might begin with a question such as, "Do you smoke?" Users who answer "yes" advance to a page that describes the hazards of smoking while users who answer "no" see a message congratulating them on their to decision to abstain from cigarettes. Regardless of their answers the first question, both users advance to question two. Though the pages are static and there is no dynamic generation of pages, to the user it appears that there is some interactivity. Despite its appearance of choice, the linear with alternatives structure preserves the general linear path through a document collection.

Linear with Options

A *linear with options* structure is good when the general path must be preserved, but slight variations must also be accommodated, such as skipping particular pages. This type of hypertext organization might be useful for an online survey in which some users might skip certain inapplicable questions. Given that the linear with options structure generally provides a way to skip ahead in a linear structure, this organization is often called *linear with skip-aheads*. An example of this structure in action might be a bicycle presentation. While some core pages may be common to all bikes, certain pages may be skipped based on a user's particular interest in mountain bikes or road bikes. In paper documentation, a survey that asks the taker to skip to a particular question based on some criterion matches the linear with options form. The basic idea of this site structure is shown here.

Linear with options

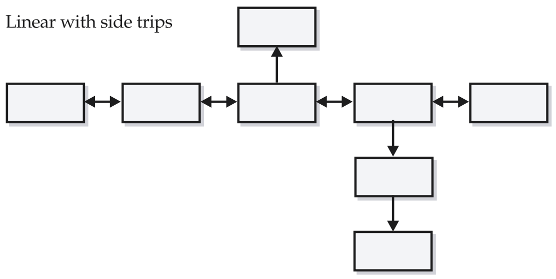


Again, this organization simulates an intelligent system even though it is often nothing more than static files in a well-designed hypertext structure.

Linear with Side Trips

A *linear with side trips* site organization allows controlled diversions. Although the user might take a short side trip, the structure forces the user back to the main path, preserving the original flow. Perhaps an article about frogs is presented in a linear fashion. A hyperlink on a particular word such as lily pad would lead to a tangential page with the definition of the word and maybe a short series of pages discussing how frogs and lily pads are related. Eventually the side trip dead-ends or returns the viewer back to the main path. A side trip to a linear progression is like a sidebar to a magazine article. Rather than distracting the user too much from the main path, this bit of information enhances the experience. Making the side note part of the main linear progression would dilute the continuity of the primary message. However, when many side trips are added into the linear progression, the structure begins to look like the common tree or hierarchy form discussed later in the chapter.

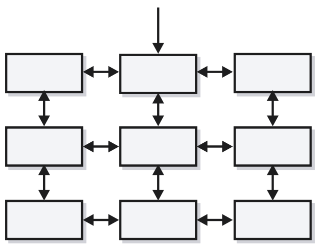
Linear with side trips



Grid

A *grid* is a dual linear structure that presents both a horizontal and a vertical relationship between items. Because a grid has a spatial organization, it is good for collections of related items; however, a pure grid structure is (so far) uncommon on the Web. When designed properly, a grid provides horizontal and vertical orientation so the user will not feel lost within the site. For example, items in a clothing catalog might be organized into categories like shirts, pants, and jackets. Another way to organize information would be by price. A grid style would allow a user to look across a price category, as well as within a particular line of clothing very easily.

Grid



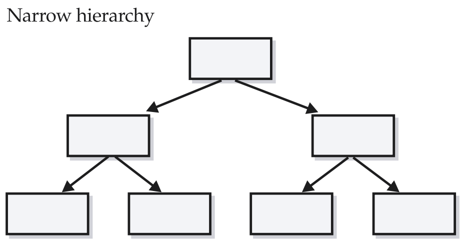
While a grid structure is highly regular and may be easy for a user to navigate, not many types of information are uniform enough to lend themselves well to this organization style. One notable exception is product catalogs.

Hierarchy

The most common hypertext structure on the Web is the tree or hierarchy form. While a hierarchy may not provide the spatial structure of a grid or the predictability and control of a linear structure, the hierarchy is very important because it can be modified to hide or expose as much information as is necessary. Hierarchies start with a root page that is often the home page of the site or section. The home or root page of the site tree serves as a "landmark" page and as such often looks much different than other pages in the site. Site landmarks such as home pages are key to successful user navigation. This is further discussed in the next chapter. From the home page, various choices are presented. As the user clicks deeper into the site, the choices tend to get more and more specific, until eventually a destination, or leaf page, in the tree is reached. Because of this arrangement, trees tend to be described by their depth and breadth.

Narrow Trees

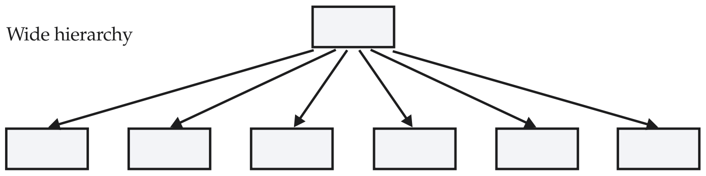
A *narrow tree* presents only a few choices but may require many mouse clicks to get to the final destination; this organization emphasizes depth over breadth.



A narrow tree may require the user to make many choices to reach a leaf page, but for some sites this is a very effective way of quickly funneling users into the correct category. For example, a Web site for an employment service generally has two main audiences: job seekers and employers looking to hire. Making this distinction obvious on the home page and requiring the user to choose a category facilitates quick and easy access to relevant sections of the site. Expanding the top-level choices to include the specific options for job seekers and for employers could be distracting. Using a narrow hierarchy as a means of progressive disclosure can help keep the user focused. However, it may increase the number of clicks required for the user to get to the ultimate destination. It is important to balance these two factors and to avoid putting up unnecessary barriers between users and the information they desire. One indication that a site hierarchy is too narrow is when there are many pages that are purely navigational beyond the home page. Remember that users want “payoff”—clicking endlessly through pages provides little more than frustration.

Wide Trees

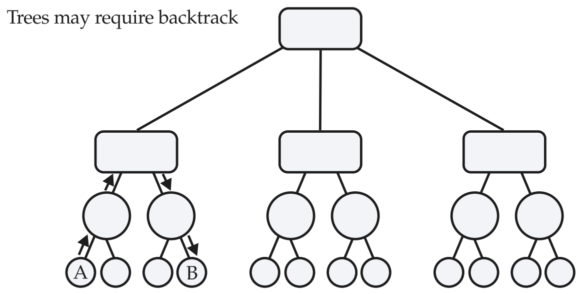
A *wide tree* or *wide hierarchy* is based on a breadth of choices. Its main disadvantage is that it may present too many options as pages have numerous choices emanating from them.



While the user only has to click once or twice to reach the content, the time spent hunting through all the initial choices may be counterproductive. Many people think that everything important must go on the home page. However, if everything gets a link from the home page, then the hierarchy is not preserved and information may lose its effectiveness—in some sense becoming lost in a crowd. Choosing the appropriate balance between site depth and breadth will be discussed later in the chapter.

Web Trees

The reality of the Web is that the typical pure tree structures are rarely used. In a pure tree, there are no cross-links, and backtracking is often required to reach other parts of the tree. Imagine that a user is at page A in the structure shown here; to reach page B, they have to back up two levels and then proceed forward.



While backtracking on the Web is possible using the browser’s Back button, links going backward are often added to pages so that users who reach a page not through its primary path can navigate the site. In many cases, pages are cross-linked by means of a navigation bar or explicit back-links to help users quickly navigate the site structure. Consider the site diagram shown in Figure 6-4.

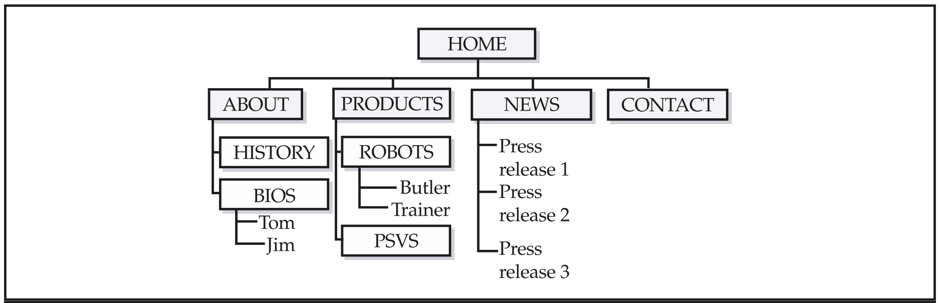
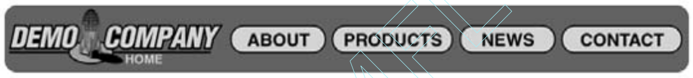


Figure 6-4. *Simple site hierarchy*

It would be common to create a navigation bar for such a site that contained the main sections of the site such as Home, About, Products, News, and Contact, like so.



With such a navigation bar, it would be much easier to jump from section to section without a significant degree of backtracking. However, the site diagram would be much more complex and look something like the one in Figure 6-5.

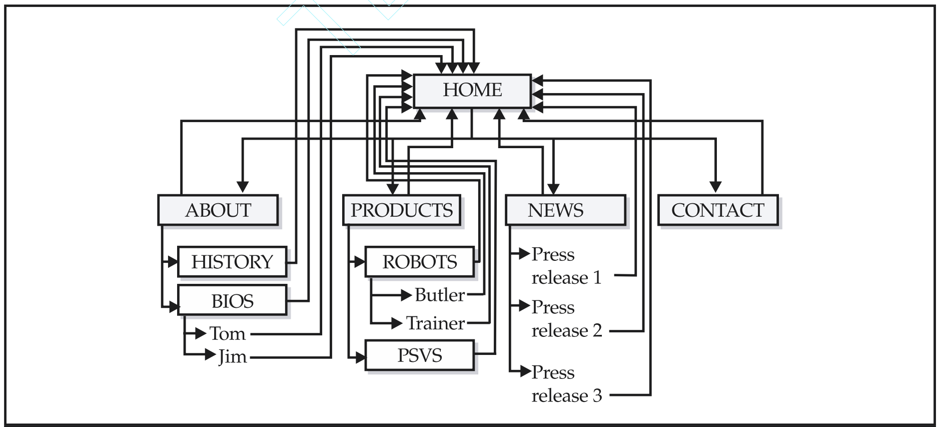
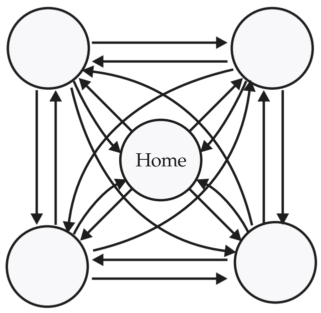


Figure 6-5. *Site hierarchy with back-links shown*

The back-links and cross-links within the site increase the complexity greatly. In this case, keep in mind that only main section pages are cross-linked. Imagine if the whole site were linked this way.

Full Mesh

A site that links every page to every other page could be considered to exhibit a structure called a *full mesh*. The following illustration shows a full mesh for a site with five pages.

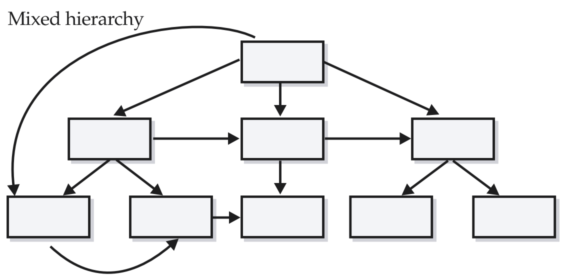


In a full mesh, the number of links is equal to the number of pages multiplied by the number of pages minus one (links = $p \times p - 1$, where p is the number of pages). This means for a 5-page site, there are 20 links. For a 10-page site, there are 90 links. For a 100-page site, there are 9,900 links (100 × 99), and for a 1,000-page site, there are nearly one million links! A full mesh doesn't really work out that well from a usability perspective, especially considering the 7 +/– 2 discussion presented in Chapter 2, nor from a visual design perspective. In practice, most sites tend to use a partial mesh style with cross-links to only the most important pages.

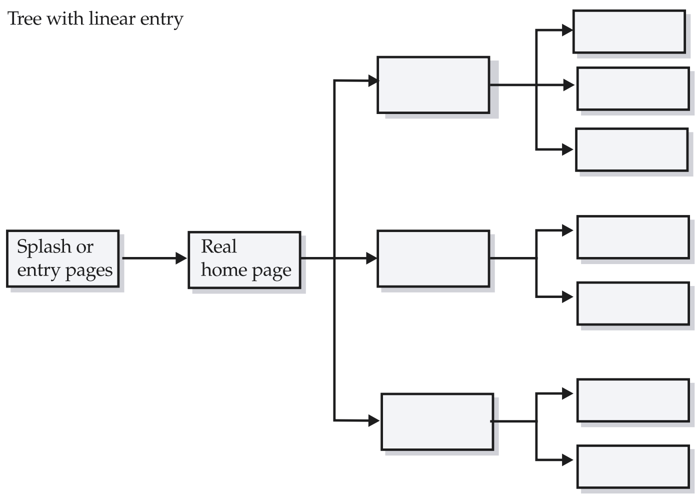
Mixed Forms

While a wide tree may present too much, too narrow of a hierarchy will hide too much information. A linear approach may provide too little user control, while a pure Web approach provides too much.

In some cases, there will be a need to augment the hierarchy to allow choices to bubble up to the top. This structure is called a *mixed form* or a *mixed hierarchy*, as the tree is the dominant form of the structure. A mixed form is probably the most common form of site organization used on the Web. Linear devices, skips, and even grids may be contained within a mixed form. Consider a site that contains Download Now or similar buttons that skip deep into a site structure. This is somewhat like a linear-with-skips structure. Other sites may contain linear tours available only from certain pages in the site. Though spatial organization is not as pronounced as in other site structures, a hierarchy is still generally evident in most mixed sites.



One common mixed style is the use of a linear structure to enter a site with a tree once the real home page is reached. Sites or sections of sites that have splash pages, installation procedures, or other linear constructs leading up to a central page that a user can explore from use this type of structure. A structural diagram of this form is shown here:



Another style, which is not really unique, is termed the *hub and spoke* structure. Many sites consist of main pages called hubs and then subpages that are reached via spokes. To visit other pages in the site, the user is forced to return to the hub page. Many portals use this style to encourage page revisits. However, there is really no difference between the hub and spoke model and a typical tree as shown in Figure 6-6.

One benefit of hub and spoke is that it may provide an easy way to conceptualize a site: central sections of content (the hub), with spokes of related content that the user briefly visits before returning to the hub. Another, related reason that designers may like to think in terms of hub and spoke designs rather than simple trees is that it may

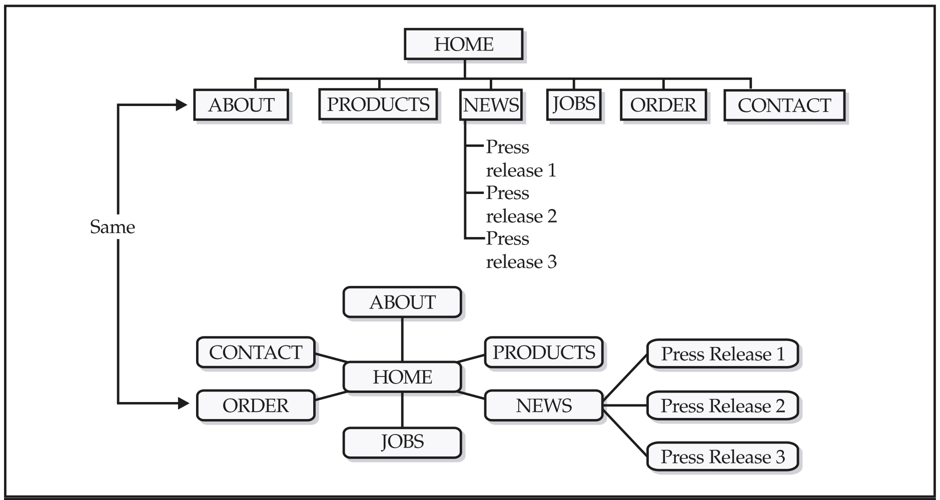


Figure 6-6. *Hub and spoke and tree structures are the same*

provide a good way to visualize site content. For example, some site-mapping tools present site diagrams in this style because they are easier to lay out than a tree structure. See Figure 6-7 for an example of a hub-and-spoke visualization of a site.

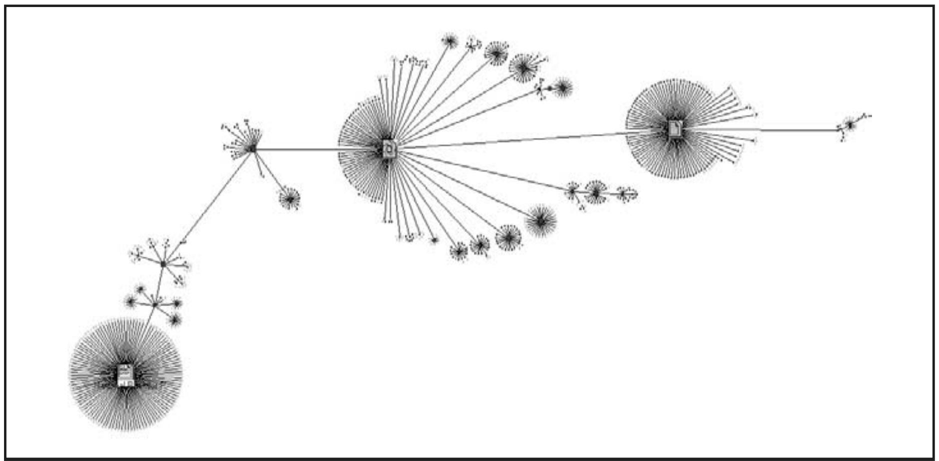
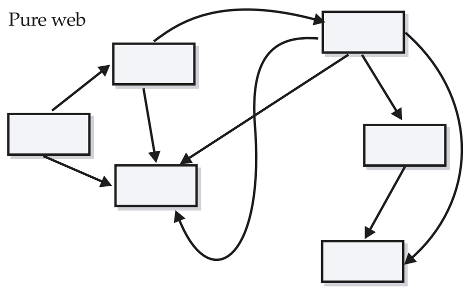


Figure 6-7. *Hub and spoke is good for site visualization*

Web Style

When too many cross-links, skip-aheads, and other augmentations are made to a structured documentation collection, the form will become unclear to the user. When a collection of documents appears to have no discernible structure, it is called a *pure web*, as shown in the illustration here.



A pure web structure can be difficult to use because it lacks a clear spatial orientation. Though information can be accessed quickly if the correct choice is made, it may be difficult to orient oneself in a Web site with an unclear structure. If a site's structure is unclear or unfamiliar to the user, they may resort to a home-page-based navigation, always returning to a top level when beginning a new task.

The benefit of a less structured form is that it provides a great deal of expressiveness. For example, a technical paper might provide links to related diagrams, supporting statements, and papers, and even excerpts from outside resources. The organization of the site may not easily fit any one of the more structured forms. While some might argue that the confusing pure web structure may cause the user to lose focus and make it difficult for participants to form a mental map of the site, this may actually not be a problem when the information or task is properly designed.

Usability and Site Structures

While a linear structure may be easier for users to comprehend than a mixed tree or pure web, users do not necessarily memorize the layout of a site or visualize a flowchart in their head of pages as they move around. In some sense, information structure may not matter if the user's focus can be retained. Whether something is back, next, or up from a current page in the site should not be the user's focus. The important things are what the users are doing and what information they are accessing. If users are content and accomplishing their goals, they really aren't lost. When organizing a site, always attempt to retain the perspective of the user visiting the site. Many, if not most, of the visitors will be relatively unfamiliar with the site and its structure. Don't assume that the organization will be clear to them, and remember that underlying organization may not have to be clear if the site is providing satisfactory utility to the user.

Consider that a user really goes through three phases upon reaching a site. Phase 1 is entry to the site. In phase 2, the user moves around the site, which could be termed the "visit phase." Phase 3 is the conclusion of the visit, in which the user exits from the site either happy, having reached a successful conclusion, or unhappy or neutral, having failed or given up on the task. Figure 6-8 shows a conceptual overview of how this might work for a site with a single entry point and single primary conclusion page, such as an order confirmation message in an e-commerce site.

In reality, sites are generally not so simple. Often there are many entry points to a site, and many exit points as well. During the visit, users may make a variety of moves both towards and away from their eventual conclusion. They are probably not completely aware of the underlying site structure of the visit and are happy as long as they feel they are making progress towards the goal state. Figure 6-9 shows a conceptual overview of a site's structure and possible user paths through the site structure.

Of course things aren't really ever as simple as the structures we've described so far. Another consideration that must be addressed is whether a visit is standalone or part of a much larger continuous session that nearly seamlessly covers multiple different sites. For example, consider a user trying to book a vacation online. Users may visit a single site trying to accomplish their goal or they may visit a search engine or portal site and

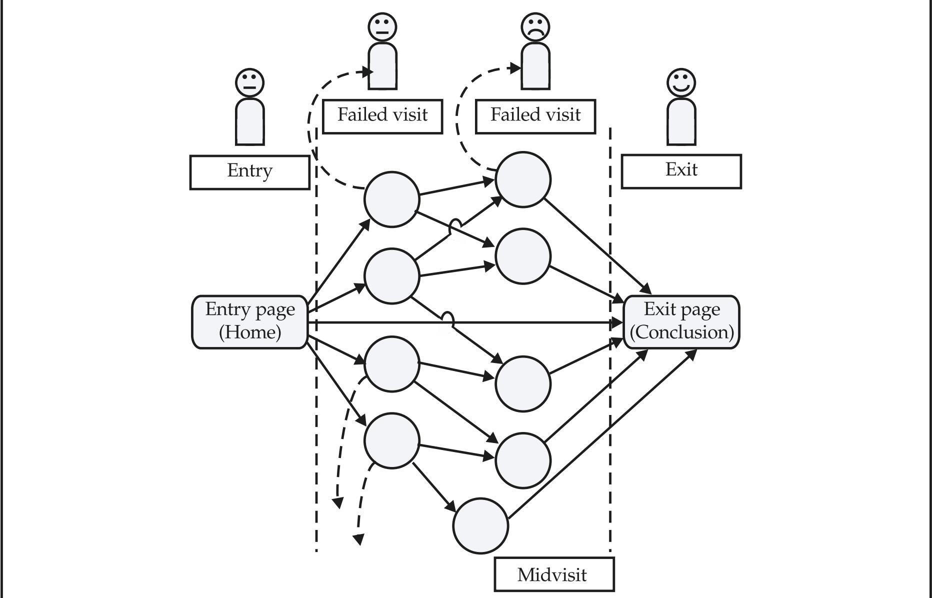


Figure 6-8. *Simple site structure from a user perspective*

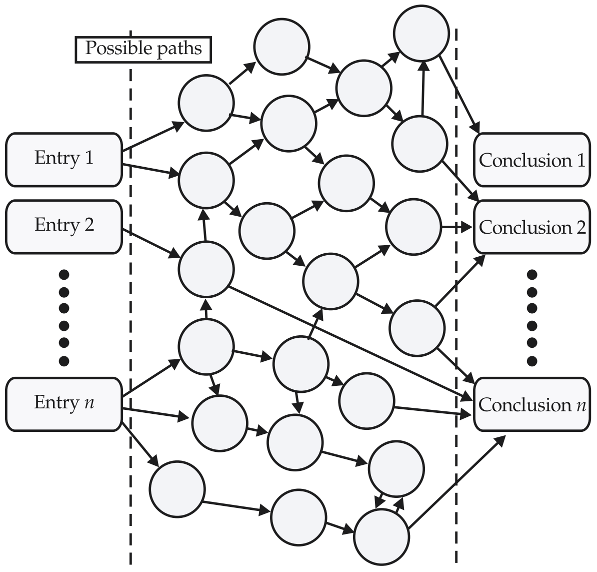


Figure 6-9. *Actual site path can be complex*

begin to bounce over numerous sites comparing prices and destinations, entering and exiting numerous sites while trying to reach their final goal. How users perceive site structures in these different navigation scenarios is important and is shown in Figure 6-10.

Even though users may not focus heavily on site structure, don't throw out logical information structuring like linear, grids, and hierarchies in favor of a pure web structure that gives up spatial information. Remember that people are spatially oriented and

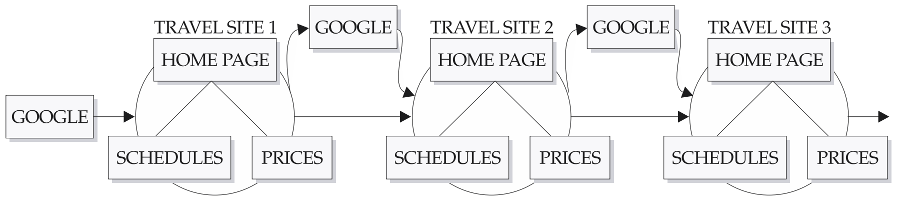


Figure 6-10. *Site structure in the context of a complex user session*

prefer to navigate in terms of location. Web sites are locations. People generally talk about "visiting" sites, not about reading them. We'll study navigation issues in depth in the next chapter.

Porous and Solid Site Structure

The previous discussion suggests that entry and exit are really the key milestones for the user. Therefore, another way to categorize Web sites would be on the number of entry points to a site. Using exit points isn't realistic because every page in a site can be considered an exit if the user just decides to quit. When a site exposes all documents with public URLs, it could be said to exhibit a "porous" structure. A porous site does not force users to enter through common points such as the home page, major section pages, and so on. Most users will probably enter through such pages, but theoretically any URL, however deep in the site structure, could be an entry point. In contrast, a site with a "solid" structure would be one that severely limits the entry points to the site to a few URLs or even a single URL. Figure 6-11 presents a graphical representation of porous and solid site structures.

The advantage of a solid site structure is that it does not expose all the inner workings of the site. By hiding such information, the underlying site content can be changed easily. Another advantage of a solid site is that by forcing users to enter through known points, their experience can be controlled much better. Users entering through known points can be exposed to important announcements, setup tasks can be performed more easily, and they can be oriented to the site in a consistent manner. However, the downside is that the user will not be able to directly enter any particular URL in the site. Power users may be extremely frustrated by the inability to save their place within a large structure.

The table below summarizes the basic pros and cons of the two site forms:

Site Type	Pros	Cons
Porous form	+Puts user in control +Allows the user to enter any URL directly or enter by bookmark	–Decreases ability to change deep pages without addressing outside linking –Does not easily provide a common entry point for announcement, setup, or orientation information
Solid form	+Does not expose site structure, making modification and maintenance easier +Forces user to enter through known points +Makes tracking of users more predictable	–Removes user from control. –May limit the effectiveness of outside search engines

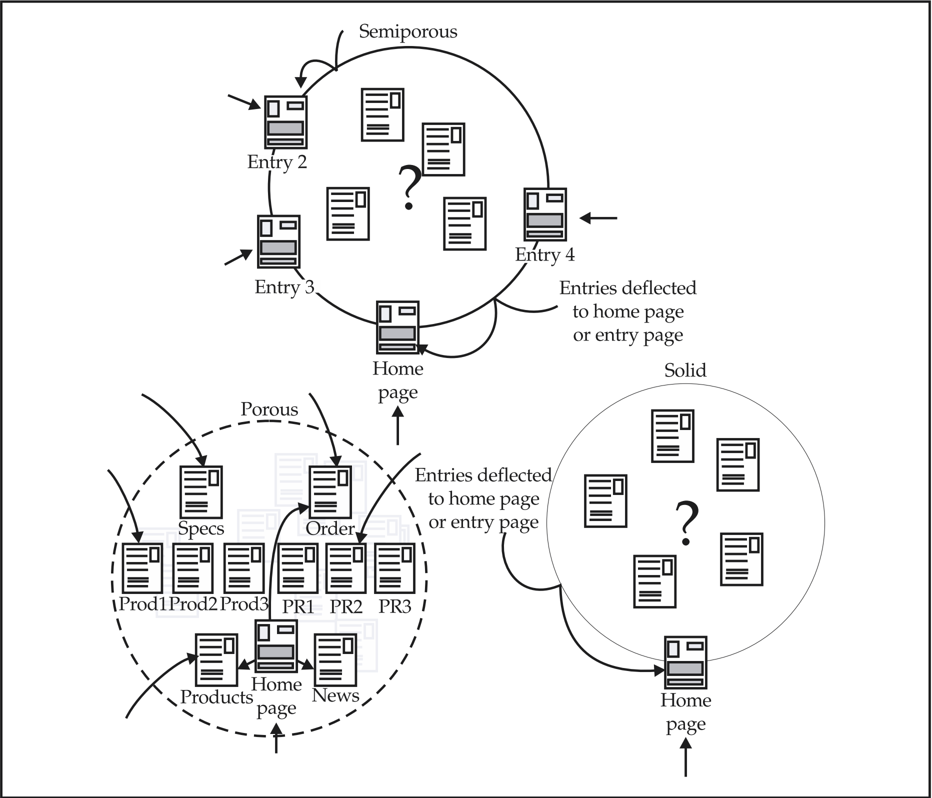


Figure 6-11. *Porous and solid site structures*

Some readers may wonder why sites should be made solid or semisolid and how this may be accomplished. First of all, understand that sections of sites have long been made this way. Consider, for example, a shopping cart checkout procedure. Letting users bookmark deep pages during a procedure does not make sense. While the user may bookmark the location, the site will apply some form of session management to expire pages or deny users from entering a process midstream. We could also check page requests to see what the referring page is and limit access. Other examples of less porous site structures include some types of dynamic content and secured sections of sites that require login. Over time, the use of defined access points to site content will have to take off if sites are to be easily changed.

Deep vs. Shallow Sites

Another way to characterize sites would be the number of clicks required to reach a destination. Consider the choice between a narrow tree and a wide tree structure. A narrow tree would require the user to click numerous times to reach pages deep in the site. A wide tree would require fewer clicks, but would require users to look among numerous links for the one that interests them. Obviously, a balance between link breadth and site depth is the best choice. Various Web studies suggest that users prefer sites that require fewer clicks and are more satisfied with a wide selection of choices. A good and highly advisable rule of thumb is to consider aiming for a depth of three clicks to get users to the content they are looking for.

Suggestion: Aim for a site click depth of three.

The three-click suggestion makes sense when considering the limited number of locations for different navigation bars on pages, traditional GUI conventions, and memory limitations of users. Inspection of Web site access logs should back up the three-click rule. In fact, many sites seem to exhibit bailouts in only one or two clicks.

Of course reducing a site's depth to three clicks or fewer is not always possible. Remember that *progress* towards an end goal must be made and shown to the user within three clicks (and ideally every click).

Suggestion: Aim for positive feedback indicating progress towards a destination with every click, with a maximum of three clicks without feedback.

Consider, however, that as a result of making a shallower site by putting numerous links on the pages, the design may inadvertently favor extremes. When faced with many choices, users may focus on extremes when making a choice.

The phone book serves as a good example of the attempt to stand out from many competing choices. For example, in alphabetical listings of non-preferential choices, observationally the letter A and Z sections are often selected. Notice how in the Plumbing section of the phone book how many firms have names like AAA Plumbing or Z-1 plumbing. To combat the effects of first choice and last choice in a large listing such as the phone book, boldfacing, color, and display-style advertisements are used to help choices stand out from the crowd.

Similarly, Web designers try to call attention to certain areas with larger sizes, bolder color, animation, or blinking—the digital equivalent of shouting. While at first these persuasion techniques may work, they may also cancel each other out or leave the user feeling annoyed. Over time a user will become accustomed to any extra stimulation and the attention-grabbing techniques lose their power; this is what is called *sensory adaptation*. Ideally, users should be given the ability to distinguish what is important from what is not and to be able to easily find the choice they are looking for.

Given that a breadth-oriented site structure seems best to reduce clicks, would the 7 +/− 2 idea related to short-term memory recall of choices make sense? Probably not, given that five to nine choices is far too few choices for many sites, consider instead five to nine clusters. Each of the clusters of links will use a different attractive technique like a color, animation, or graphic. With a maximum of five to nine clusters and five to nine items per cluster, a page could hold anywhere from 25 to 81 links.

Suggestion: Even for wide site structures, consider a range of 25–81 links per page when page links are ideally clustered.

Unfortunately, with dozens of links, users are bound to make mistakes, and important links may be lost in the clutter. Because of this potential for user mistakes, many sites favor a redundant link approach, in which numerous links lead to the same conclusions. Convention suggests that the number of links to a particular page is proportional to its importance.

Premise: The more important the page, the more redundant links should be provided to it.

Consider how many links in a site point to a home page—or to software download pages or a purchase page—and it becomes apparent that redundant links are commonplace within many sites. Increasing the number of links pointing to successful conclusions just increases the odds of the user hitting the right link. Be careful not to add too many redundant links, though, lest the users feel they are being pushed towards a particular page. Again, the control issue becomes apparent. If nearly every link in a page pushes a user towards a particular conclusion, the user may feel frustrated with the lack of control.

Suggestion: Redundant links in a site should be no more than 10 to 20 percent of a page's total exit links.

Despite the likelihood that users are better able to deal with flat site structures, many sites completely avoid building sites this way. Certainly some of the reason could be attributed to developers being unaware of the idea, but many times the rules of thumb are avoided on purpose. Consider a site whose revenue is primarily from banner advertisements. For such a site, the more banners viewed by the user per visit, the better. In the mind of the owner of such a site, a design that gets users quickly to their destinations is one that takes money out of the site's own pocket. Many banner-driven sites favor hub and spoke site design or deep tree structures as a way of forcing the user to click through numerous pages and view more banners.

Of course, there is a limit to the "click more, view more ads" approach, in that unsatisfied users won't continue to click if they get frustrated. In some situations, site designers will design to reduce clicks to the lowest tolerance level without overly confusing a user with too many choices. In other situations, they will want to increase clicks to the maximum tolerance level without frustrating the user. Oftentimes the specific type of site being built drives the type structure used.

Picking a Site Structure and Type

The idea of picking the correct structure for a Web site by organizing information into a collection of pages is often called *information architecture*. Choosing the correct structure for a site is complex and can be influenced by many factors. For example, the data itself may suggest a particular method of organization. This could be considered a bottom-up approach. For example, a slide show really should be organized in a linear fashion, since the logical order of the presentation would be lost if the information were presented in another form, such as a tree.

Another way to consider organizing information would be more top-down, based upon the use of the data. This approach would give priority to who is using the site and how the data it provided is consumed. For example, linear structures will provide little control for the user and limited expressiveness, but they will be very predictable. Novice users will prefer simple structures such as linear structures or deep trees, since the choices to be made in such structures are relatively easy.

Premise: Novice users prefer sites with predictable structure and may put up with extra clicks or a lack of control to achieve a comfortable balance.

Of course, a power user will often find a site with a very rigid structure or one that requires a large number of clicks to be very restrictive. Spatial feedback is not as important to the power user as control or flexibility of navigation.

Premise: Power users or frequent site users want control and will favor structures that provide more navigation choices.

Each site structure style has its own pros and cons. Figure 6-12 shows the relationship between the expressiveness and predictability of the different site structures. While linear is very predictable, it provides a limited relational view. While a pure web form is very expressive, it can be confusing. The hierarchy and mixed structures share the middle ground, allowing users to move progressively closer to end results in a predictable manner. When building sites that are not dynamic, aiming for the middle ground is the best bet. Given this observation, it is no wonder that most sites tend to exhibit some form of hierarchy.

Proper information design is key to the development of a successful Web site. If a site has great content and a great interface, but poor information architecture, it may be relatively useless. If the user cannot easily find the information, the site loses its effectiveness. Most sites now use a mixed hierarchy approach that is familiar to many Web users. Depending on the goals of the site, several types of structures might be combined. For example, while the overall structure of a site might be a hierarchy, a pure linear structure could be used to provide an introduction to a company, and a narrow hierarchy or even a grid could be used in the technical support section.

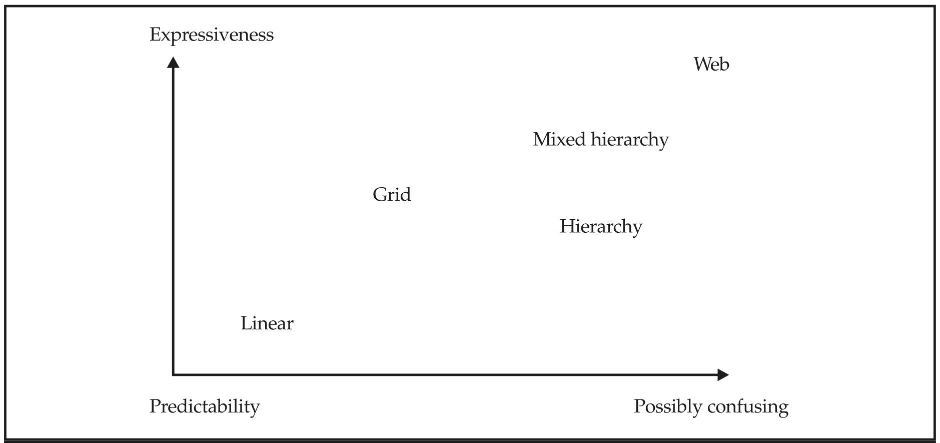


Figure 6-12. *Site structure: expressiveness versus predictability*

The key point of site structure is to make the site easier for the user to navigate. Always remember that users are not going to intimately understand the underlying site structure—nor should they have to. Remember that from the user's point of view, they enter the site, move around the site trying to accomplish their goal, and then eventually leave. Users will not care about structure as long as they achieve what they want in a positive way. So, any structure that we choose for a site should help users navigate around and improve their likelihood of success. The next chapter focuses on site navigation and organization.

Summary

One way to categorize Web sites is by their audiences. Public Web sites tend to have loosely defined audiences, while a private intranet's audience may be very well known to the site creator. Audience considerations greatly affect the design considerations of a site. Sites can also be categorized by size, technology, and visual designs, but the most important grouping is related to the purpose of the site. Obviously, all sites do not have the same purpose and thus do not necessarily share the same design considerations. Commerce pages have much different considerations than entertainment pages. Designers should always be careful not to apply the same design criteria to a site regardless of audience or purpose. However, despite audience or purpose, most sites share similar organizations. Some sites have simple architectures, like a linear progression of pages, while others exhibit complex hierarchies or mixed forms. When building the site's structure, always consider cognitive science issues and attempt to balance click depth with link breadth. Designers should understand that the logical organization of the site and the physical organization do not have to match. In fact, the structure of the site is often more useful to the designer than to the user. While structure can improve a site's organization, users may not always be aware of a site's form as they navigate toward desired content or attempt to complete a particular task.