Ann Navarro

effective WEB DESIGN

second edition

Master the Essential Principles of Building an Effective Web Site

Fully Revised to Cover XHTML and the Latest Web Design Technologies

Sample Code and Over a Dozen Web-Development Utilities

Full-Color Insert Is a Learning Tool and Reference!

SYBEX

25 YEARS OF PUBLISHING EXCELLENCE

Effective Web Design, Second Edition

Ann Navarro

SYBEX®
Associate Publisher: Cheryl Applewood
Contracts and Licensing Manager: Kristine O'Callaghan
Acquisitions and Developmental Editor: Raquel Baker
Editors: Joseph A. Webb, James A. Compton, Colleen Wheeler Strand
Production Editor: Dennis Fitzgerald
Technical Editor: Marshall Jansen
Book Designer: Maureen Forys, Happenstance Type-O-Rama
Graphic Illustrator: Tony Jonick
Electronic Publishing Specialist: Maureen Forys, Happenstance Type-O-Rama
Proofreaders: Nelson Kim, Nancy Riddiough, Leslie E.H. Light
Indexer: Ann Rogers
CD Coordinator: Christine Harris
CD Technician: Kevin Ly
Cover Designer: Design Site
Cover Illustrator/Photographer: Dan Bowman

Library of Congress Card Number: 2001088112
ISBN: 0-7821-2849-1

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1
Having a presence on the web is no longer the exception—it's the rule. Everyone has web sites—even people's pets! Unfortunately, there is a huge difference between building a web site and building a *good* web site. On the Internet you can find thousands of examples of poorly designed web sites with broken links, bad color schemes, inscrutable menus, and the like. You can also find a few examples of really good web sites.

A good web site has a clean, functional design. It is accessible by everyone. It doesn't use proprietary features so that it is portable among operating systems and browsers. If you are building a site for your pet, it may not be important to you that it be a good web site; if you are setting up an online take-out ordering system for your restaurant, it is essential. After all, the goal of any web presence should be that your users are not distracted, can find what they want easily, and will continue to use your resource. A clean, efficient, and portable design will help ensure your users have a positive experience. Developing a good web site is not always easy—but it is almost always required. Through *Effective Web Design*, you can learn the techniques that will make your web presence, whether for your pet or your restaurant, one that people will return to again and again.

—Shane McCarron, Applied Testing and Technology, Inc. Editor, W3C Modularization of XHTML Proposed Recommendation

Dedication

To Scott Artigue, my favorite Cajun, who gave me the ability to find the peace of mind that let the words flow.
For Dave Navarro, my husband, business partner, and dearest friend, for putting up with all this again. Now we can head under the sea for that cheeseburger in paradise.

—Ann

Acknowledgments

Any book project requires the effort and dedication of a great many people beyond the author credited on the front cover. That includes editors, reviewers, publishers, production staff, and various contributors and personal supporters of the author. This is where I get to acknowledge each of you for the role you've played in making this book the best that it could be.

At Sybex, thanks to Raquel Baker, my acquisitions editor; project manager Dennis Fitzgerald; editor Joe Webb; and technical editor Marshall Jansen.

Thanks goes to my agents David Rogelberg and Neil Salkind along with the support staff at Studio B Productions, who so deftly handle the business aspects of my writing career. Your efforts clear the way for me to concentrate on transferring the thought from the brain, through the fingers, and into the computer.

I can't forget Tim Berners-Lee, for having invented that first system that blossomed into this incredible online world that we all now live in.

Many thanks must go to Shane McCarron, my friend and colleague on the W3C HTML Working Group, who went above and beyond the call of duty in surrendering numerous hours of his own scarce spare time helping me work through bugs, queries, and other trivia that results when writing about a specification that's not quite finalized. Beyond your technical expertise, your wit helped to keep me afloat when I needed it most.

And finally, a very special thanks to Susan Zachman and Bob Kennedy; you know why.

—Ann Navarro

Table of Contents

Introduction

To get the most out of this book, please take the time to read these next few pages. I'll discuss how the book came into being, who the intended audience is, and what you can expect to learn. By reviewing these items, you'll know quickly when your expectations will be met and which portions of the book may be useful as a quick reference if you need to dive into a specific segment first before starting with Chapter 1.

Why This Book?

This book was born out of the frustration of reading other texts on HTML and Web design and being told repeatedly that one had to perform one set of tasks in order for a site to look good in the Internet Explorer browser and one other set of tasks for the site to be viewed in the Navigator browser. Surely, it wasn't necessary for designers to go to those lengths simply to produce a functional and effective Web site, was it?

After much rumination and quite a bit of experimentation, I brought together techniques that obviated the need for such client-dependent authoring. The result is known as the *cross-compatibility concept*.

Cross-compatibility relies on two basic principles: A valid document is exponentially more likely to perform as the author intended than an invalid document. When a browser can't handle something in a valid document, perhaps because it's an older browser faced with new-yet-valid content, the result shall be a *graceful degradation* of the intended result or, at worst, simply passing the element content through in its original state to the browser.

By subscribing to the cross-compatibility concept, Web authors will rarely, if ever, need to provide browser-specific versions of their Web sites or exclude users who don't have a certain browser. That, of course, is the essence of *Effective Web Design*.

Who Should Read This Book

This book can serve two audiences. The first group consists of those who are just coming into Web design or have traditionally relied upon design tools to create Web sites and for the first time want to learn how everything under the hood works. The second audience is composed of designers who know HTML and have even written it by hand extensively but are now looking to learn the new XHTML specification, either as an end in itself or as a step toward working with XML. Both the novice and the experienced designers should come away from this book with new knowledge and techniques to incorporate in their design arsenal.

Who Should Read This Book

This book can serve two audiences. The first group consists of those who are just coming into Web design or have traditionally relied upon design tools to create Web sites and for the first time want to learn how everything under the hood works. The second audience is composed of designers who know HTML and have even written it by hand extensively but are now looking to learn the new XHTML specification, either as an end in itself or as a step toward working with XML. Both the novice and the experienced designers should come away from this book with new knowledge and techniques to incorporate in their design arsenal.

Part I: XHTML Fundamentals

HTML knowledge is not a prerequisite for learning from this book—hence this part on fundamentals. However, the HTML enthusiast should not skip past this part. You will learn about XHTML in the same fashion that you'd learn HTML, assimilating the changes between HTML and XHTML along the way. The novice to either language will simply learn XHTML from the start and may find the references to the differences enlightening from a historical perspective.

- **Chapter 1** The first chapter gives a short discussion of the history of HTML, beginning at the lab in Switzerland where Tim Berners-Lee first envisioned the system up to HTML 4 and XML 1.
- **Chapter 2** This chapter reviews the history and availability of Web browsers, including new advances in browsing from alternative devices such as PDAs and cellphones.
- **Chapter 3** The third chapter teaches you how to build a basic XHTML document, beginning with selecting your editor and introducing the required segments of every XHTML document.

- **Chapter 4** This chapter goes further into XHTML document content and covers text, images, and special characters.
- **Chapter 5** The pages here present techniques for creating tables, including advanced features such as row or column grouping and shading and spacing options.
- **Chapter 6** The chapter talks about working with frames, including creating the shell for the frameset and supplying content to the framed window spaces.
- **Chapter 7** The seventh chapter tells you everything you need to know to get working with Cascading Style Sheets and presents a brief introduction to style in XML using the Extensible Stylesheet Language.
- **Chapter 8** The last chapter of Part I introduces you to the world of user feedback through XHTML forms. You'll learn how to collect data in a variety of formats.

Part II: The Site Design Process

Knowing how to work with elements and attributes will get you started, but effective Web design also requires effective presentation.

- **Chapter 9** Chapter 9 discusses preproduction planning, designing navigational menus, and the construction of image maps used for navigation.
- **Chapter 10** The next chapter digs into the mechanics of search engines and Website indexers such as Yahoo! and Alta Vista.
- **Chapter 11** Chapter 11 introduces the process of validating your work and correcting errors both in your XHTML and CSS documents.
- **Chapter 12** The next chapter describes the information about your audience that you'll want to collect and how that data needs to be managed to secure privacy and user confidence.
- **Chapter 13** Planning is important, and Chapter 13 teaches you how to plan the design of your Website, step by step.
- **Chapter 14** The next chapter reviews effective visual-presentation techniques, including the use of images, text and white space, and designing to proven predefined grids.
- **Chapter 15** Chapter 15 covers the diversity of fonts available and tips for getting the most out of text for both design effect and readability.
- **Chapter 16** The next chapter provides a full primer on color theory as it applies to the output on a computer monitor or other digital-display device.
- **Chapter 17** Chapter 17 teaches how to create professional-looking graphics and examines the tools currently favored by Web designers and what to do if drawing is not exactly your forte.
- **Chapter 18** The pages here review the advancements made in multimedia presentation on the Web, including Shockwave and Flash presentations and streaming video, and audio.
- **Chapter 19** Chapter 19 covers doing business on your site. E-commerce options for the small business are discussed, including information on server security, shopping carts, and taxation issues.
- **Chapter 20** The last chapter in Part II looks at accessible design techniques, that is, providing for equal access for people with disabilities. Accessible design techniques often mimic the needs of users of small or nontraditional devices such as WebTV, PDAs, and cellphones.

Part III: Moving Forward with XHTML

In this section, I'll look at where XHTML is going, now that XHTML 1 has become a full W3C Recommendation. Many more features are coming out of the standards process, and this is your chance to learn what's on the cutting edge.

- **Chapter 21** Chapter 21 introduces the next wave of XHTML development: Modularization. XHTML 1.1 is built using Modularization techniques to create a closer-to-XML version of XHTML than was found in XHTML 1.
- **Chapter 22** The last chapter guides you through the process of reading and authoring document type definitions, the basis of XHTML modules and XML documents.

Appendices

- **Appendix A** The XHTML Element and Attribute Reference is your one-stop source of quick information about each XHTML element and the attributes available for them. Indicators for inclusion in XHTML 1 Strict vs. Transitional are also provided.

- **Appendix B** Each of the Abstract Module Definitions found in the Modularization of XHTML specification is found here for easy reference.

Color Section

This 8 page section is a selection of images that bring you the most benefit when seen in color, versus the grayscale found elsewhere in the book.

What's on the CD-ROM

The CD-ROM contains a series of files that will be used for the tutorials and exercises in the book. In several places in the book, you will be directed to the CD-ROM, either to read a file at the beginning of an exercise or to see the illustrated result of an exercise.

I've included a copy of the most recent versions of popular browsers, so you can be sure to have the most up-to-date tools available while learning about XHTML. You'll find XHTML design tools, such as text editors, validation tools, and other editing programs. Graphics programs are also loaded on the CD, giving you a chance to try out some of the most popular programs available on the Web or at retail outlets.

Contacting the Author

Ann may be reached at <`ewd2@webgeek.com`>, or you can visit `http://www.webgeek.com/books/ewd2/`. Known errors, omissions, or other errata will be listed on the Web site until a new print run takes care of the matter. Please check the Web site before reporting a problem.

Conventions Used in This Book

Tips provide extra information that will be helpful in completing the task at hand.
When you see this icon, the note discusses matters pertaining to the cross-compatibility concept.
Warnings alert you to situations where particular care must be taken to avoid problems.
WWW The globe icon alerts you to where to find specific information and resources on the World Wide Web.

Sidebars

Sidebars develop specific ideas and expand upon information that is related to the chapter's primary information. Sidebars also present related troubleshooting techniques and sources for additional information.

As you read this book and learn about effective Web design, remember one thing above all—have fun!

Chapter 1: XHTML and the Cross-Compatibility Concept

Overview

What makes a specific Web design effective? The answer takes up the remaining 22 chapters in this book. Effectiveness, however, is a subjective measurement. How can you judge whether something is effective? Web designers might ask the following questions: Is your site being used? Can the visitors you hope to attract access your site without barriers? Do visitors experience your site in the manner in which you intended? Does your site get your message across? Does the site entertain or inform? Can you quantify the site's success through increased sales, decreased support calls, or inquiries from markets you've not previously been able to enter?
These questions are not exhaustive but should spark your imagination about the multitude of ways that *effectiveness* can be defined for Web design.

In order to give you a context for better understanding what effective Web design can mean for you, this chapter examines the history of the Internet, looks at the emergence of XHTML, and provides a thumbnail sketch of the Web-browsing community.

This chapter covers the following topics:
- What makes design effective?
- How the Web was born
- Evolutions in HTML
- A new frontier in XHTML
- The cross-compatibility concept

In the Beginning

As improbable as it may now seem, the vast infrastructure and massive network we now know as the World Wide Web was the brainchild of a single man: Tim Berners-Lee. He was a scientist at CERN (`http://www.cern.ch`), the Centre Européenne Recherche Nucleaire, also known as the European Laboratory for Particle Physics, and had become frustrated by the necessity of using several computer terminals to access various stores of information belonging to the laboratory. Even more frustrating was the need to be proficient with multiple programs that were peculiar to each terminal type.
Berners-Lee envisioned a world in which access to data would be a simple task, accomplished in a consistent manner regardless of the terminal or program in use. The concept of *universal readership* was formed, embracing the idea that any individual, on any type of computer, in any location, should be able to access data by using only one simple and common program.

Tip I don't believe Berners-Lee could have envisioned the browser war of today (see Chapter 2). But in many ways, his ideas and proposals have survived and flourished on the "modern" Web. He remains an integral part of the Web community, serving as director of the W3C—the World Wide Web Consortium. (For more about the W3C, see Chapter 3.) He published a book about the creation of the Web and its evolution in September 1999, titled *Weaving the Web* (Harper). For more information, see http://www.w3.org/People/Berners-Lee/Weaving/Overview.html.

Linked Information Systems

Before the Web, most information storage and retrieval methods were *hierarchical* in nature. So each bit of data was sorted in a structured manner. The files on your computer are sorted in that way by default; files that begin with numbers come first as a group, then files that begin with letters appear next. Within each group, files are sorted from lowest value (zero in the numbers group, *a* in the letters group) to highest value.
A *linked information system* doesn't require such order or structure. You can travel from the number *1* to the letter *q* and back to the number *9*, all the while having the information in each bit of data relate appropriately. The system is the basic structure—or perhaps I should say nonstructure—of the World Wide Web. Documents are stored on thousands of computers, or *nodes*, around the world. Despite the unordered nature of the storage system, a document on a server in California can provide an entry point, or *link*, to a document stored on a server in Finland. Furthermore, you can link directly to the pertinent information; you won't have to search through the entire site.

Such fluidity, combined with the programming efforts that brought about what we now know as browsers, has fulfilled the vision that Tim Berners-Lee had way back in the virtual "dark ages" of the Internet: the World Wide Web.

From Links to Hypertext to the W3C

When Berners-Lee began laying out his plan for a linked information system in 1989, he was unaware that a term already existed for the process—*hypertext*—which was coined in the 1950s by Ted Nelson. It can be defined as "human readable information linked together in an unconstrained way."

In the beginning, these systems often used proprietary interfaces. As early as the late 1980s, work was underway to standardize hypertext systems. These efforts resulted in the Internet arena in the development of the Hypertext Markup Language, or HTML, established by Berners-Lee. The original version is now known as HTML 1.

In 1993, a young student at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign named Marc Andreesen created a graphical user interface for the Web known as Mosaic. It was originally developed on the X Windows platform, a Unix-based environment. (At the time, Unix was the most common operating system for Internet-connected computing systems—and it still is.)

Later that year, the Mosaic programmers began devising custom extensions to HTML to expand on the capabilities of the Mosaic browser. Little did they know the trend they would set in motion! In mid-1994, the extensions, along with suggestions from other individuals and institutions, were combined into an updated version of HTML, HTML 2, under the supervision of the Internet Engineering Task Force (IETF).

The W3C was formed in late 1994, with a top priority of guiding the structure and growth of HTML. The W3C has published an activity statement that summarizes its involvement in the standards process and defines its goals and vision for the future. You can find this document online at `http://www.w3.org/MarkUp/Activity`.

The Current HTML Standard

The current direction in HTML was determined in large part as the result of a two-day workshop held by the W3C in May 1998 near San Francisco. The workshop, entitled "Shaping the Future of HTML," was convened to answer a number of questions and concerns being raised in the Web community, such as:

- With the advent of XML, would the W3C abandon HTML?
- If HTML continues, how will it interact with XML and all the other XML-based languages?
- How and when should the W3C go about creating the next version of HTML?
- In general, what should the goals of any new version address?

It became clear at the end of the event that participants, composed of academics, software vendors, implementers, and end users, felt that placing additional features into a new HTML 5 recommendation wouldn't be easy or even advisable. HTML would definitely need to get along with XML, and sufficient differences existed between the two that simply building upon HTML with new features wouldn't work.

Comments on HTML 4 collected from users and implementers needed to be incorporated into a "cleaned-up" version of HTML; that version was placed at the top of the list of deliverables for the new HTML Working Group. The result was HTML 4.01, which remains the most recent HTML standard to be published.

The XML Challenge

What is XML, and why do Web designers need it? *XML*, the Extensible Markup Language, is a framework for creating new markup languages, or *vocabularies*, as they're becoming known. What does it mean to know how to "write XML"? That's like asking if you "write English." But do you write Haiku? Poetry? Children's stories? Technical manuals? Biographies? Perhaps science fiction? Each of these genres can be written in English, but none of them define what English is.

If I take the writing analogy a bit further, poetry can be written in many different forms. Sonnets, limericks, and pantoums are all different expressions of poetry; in essence, they are different *vocabularies* within the framework of poetry.

How does the analogy apply to markup languages and Web design? XML allows individuals and companies to develop their own vocabularies for marking up documents; in essence, it is a new form of writing. New vocabularies are created by defining elements and attributes that describe structures unique to the needs of your environment, such as a `deposit` element in banking, or an `ingredient` element in cooking. By storing document data in structures that are descriptive and meaningful for their environment, specific values and strings stored in those structures can readily be identified and manipulated with a style sheet for presentation in a Web browser or any other application that can parse XML.

Web designers will soon be called upon to work with data stored in these new vocabulary structures just as they have been called upon to interact with flat-file data stores and relational databases as the Web has become more and more dynamically generated. Starting out with HTML in XML form can significantly shorten the learning curve when the time comes to work with these new vocabularies.

XHTML: The HTML-XML Bridge
Considering the future of XML, a new metaphor emerged for the W3C's work: A "bridge" needed to be built between HTML and XML. The result? *XHTML*, the Extensible Hypertext Markup Language. In late 1999, The W3C published XHTML 1 as a full *recommendation*. In W3C terminology, a recommendation is what most of you think of as a specification. It's a document considered both defining and final.
But XHTML isn't just a bridge—it will stand on its own for an undetermined time. The task of the HTML Working Group (HTML WG), the group responsible for developing XHTML, continues with the updating and Modularization of XHTML (XHTML 1.1 is a modularized version of XHTML 1) and the development of XHTML Basic, a version intended as a starting point for small devices. *Modularization* allows for easy integration of logically related portions of elements, which have HTML semantics, with other custom XML vocabularies. As you work through this book, I'll explore these exciting new opportunities that can impact everyone from the true novice to the professional IT manager.

The HTML WG also has several exciting complementary projects underway, including the following:

- **Synchronized Multimedia Integration Language (SMIL)** Pronounced "smile," SMIL allows Web designers to bring television-like content to the Web, without the traditional constraints of massive bandwidth requirements or learning complicated programming languages. SMIL can be authored directly in a text editor like HTML. You'll find the W3C's activity statement for synchronized multimedia at `http://www.w3.org/AudioVideo/Activity`.
- **Mobile Access** Accessing the Web doesn't happen just from a desktop computer or laptop any longer. You have wireless access with PalmPilots and other hand-held devices, and even with many cellular telephones! Access to the Web and Internet from these devices is a great interest as the country continues to move into an "access-anywhere" environment. A discussion of the W3C's work in this area is online at `http://www.w3.org/Mobile/Activity`.

Innovations in Web technologies continue to come at a furious pace. As a Web designer, it's your job to keep yourself apprised of new developments. Not many will be able to immediately adopt each new technique as soon as the details hit the Net, nor should you expect them to.

Balancing the properties of current HTML and XHTML recommendations with the abilities of the browsers in use by the general Web population and resisting the temptation to quickly incorporate new innovations in your designs can be difficult.

Spanning Platforms, Versions, and Technologies

When asked which Web browser they use, there's a good chance the average surfer will respond, "Netscape" or "Microsoft." But as anyone who's watched even a smidgen of news in the past year knows, Microsoft and Netscape are the names of companies, not products. So the answer is like saying you have a General Motors when asked what kind of car you own. GM produces dozens of different models in any number of model years and sells them under a variety of nameplates (Chevrolet, Pontiac, Buick, and so on). The same is true for Web browsers. Browsers are categorized by *platform* (the operating system being used on the computer in question—for example, Windows, Mac, and Unix), by product name (Navigator from Netscape, Internet Explorer from Microsoft, for two examples), and by version number (similar to the model year of a car, though they often come out more frequently than that).
As you might suspect, a particular browser's features are slightly different within the same version number across platforms, based on the abilities and strengths of each operating system. Of course, nearly all of them are significantly different across versions; features are added or taken away and hopefully improved. Differences across versions may be as minor as which font is the default—a sans-

serif font on a Mac versus a serif font on a PC—or as major as whether the browser supports Java. For more information than you'll probably ever want to know on the availability, features, and popularity of browsers, visit Browser Watch at `http://browserwatch.internet.com/`.

They Don't Call it the *Bleeding Edge* for Nothing

New technology is often described as being on the "cutting edge." The phrase sounds sexy, high-tech, and awe inspiring. With the latest and greatest constantly evolving in the online world, chasing after that cutting edge can be like juggling kitchen knives: You're likely to get nicked in the process.

How then, do you, a Web designer, balance the requirements of the existing technology with the frequent demands of clients or superiors who want the nifty new Web gizmo they saw on someone else's site last night?

The Cross-Compatibility Concept

Your goal is really quite simple. It stems from Tim Berners-Lee's original concept: Anyone, on any type of computer, anywhere in the world, should be able to access your document and achieve the expected results. To achieve the ideal of universal readership, you need to keep in mind that:

- People access the Web from three major computing platforms—Windows, Macintosh, and all Unix variants—and on many less common platforms, such as NeXT, Be, and VMS.
- Dozens of Web browsers are available for public use. Although the two most popular, Microsoft's Internet Explorer (IE) and Netscape's Navigator, are used by the vast majority of the installed user base, hundreds of thousands of netizens do use other browsers every day.
- Even among IE and Navigator users, half a dozen or more variations exist for *each platform*. Both IE and Navigator have had five or six major releases and several minor ones for each major release. (You can learn more about browsers in Chapter 2.)
- Not every Web user is a Web junkie. Average users don't necessarily know how to upgrade their browsers nor are they inclined to do so if they did. "If it ain't broke, don't fix it" is a common sentiment.
- Most computer users never change the defaults set on their systems at the factory. So they often browse at a screen resolution of 640 × 480 pixels with 256 colors on smaller (15-inch) monitors. Or perhaps they use a resolution of 800 × 600 on a 17-inch monitor more standard with new systems, but again with 256 colors. (For more on resolution, see Chapter 17.)

You *can* accommodate these often conflicting visitor needs with careful consideration and planning. By focusing your design efforts on the XHTML features that can be interpreted by the widest array of browsers on the widest array of platforms, you'll accommodate the widest array of visitors. That doesn't mean, however, that you always have to cater to the lowest common denominator—that is, the minimal abilities of Lynx, a text-based browser still often found in academic settings. (See Chapter 2 for more information on Lynx.)

In this book, you'll learn which techniques work on which browsers. You'll become familiar with the XHTML recommendations of the W3C and how to incorporate "valid" (correct) XHTML markup into your documents. Every step of the way, I'll show you tips and tricks for including some of the more exciting, cutting-edge Web-design skills, all the while producing documents that degrade gracefully when viewed on less capable browsers.
Degrade? It's not as ugly as it sounds. *Graceful degradation* is simply when a browser can't render your documents the way that you intended but still produces a visually and functionally acceptable result.

Who's Out There?

Even though the Web originated in Europe, its growth exploded in the United States, mostly due to the intellectual gold rush in the U.S. computing industry. Another factor, though less important, was the existing framework of ARPAnet—the network of the United States Defense Advanced Research Project Agency, a testing ground for new technologies to link universities and research centers together. In terms of raw numbers, the Web is still dominated by U.S. interests, but by no means should it be interpreted as only a U.S.-based phenomenon. More than 240 countries around the world are wired into the Internet.

Tip *At http://www.ics.uci.edu/pub/websoft/wwwstat/country-codes.txt, you can find a list of all countries that have a two-character Internet-addressing country code.*

Even in the United States, most of the population is not yet online. Among those who are, hardware configurations are likely to be significantly less powerful than that used by those who frequent the Silicon Valley geek emporiums as often as most people zip through a drive-thru at the local burger joint. WWW The 10th annual survey of Georgia Tech's Graphics, Visualization, & Usability Center (http://www.gvu.gatech.edu/user_surveys/survey-1998-10/), which took place October 10-December 15, 1998 and is the most recent version of the study available, measured characteristics of 5,000 Internet users who participated in the study through various means of self-selection. They may have responded to a notice on a Usenet newsgroup, clicked a banner ad, or responded by viewing an ad in traditional media such as print and television. The study revealed the following statistics:

- Thirty-six percent of U.S. respondents were women, a slight downtrend; however, the difference was close enough to be attributed to the sampling method.
- The vast majority of respondents were white (87 percent), though the racial diversity broadened in the younger age ranges.
- The average age of an Internet user was 37 years, a substantial increase (more than two years) since the last survey. It is noteworthy that in Europe the average age has decreased to just more than 30 years.
- The lesser the number of years respondents reported being online, the greater their age, meaning older users composed a great portion of the novice-user segment.
- Women have the highest representation in the age range of 25–50.
- An interesting new question was added regarding the area in which the respondent lived. The majority (48.9 percent) of respondents lived in a suburban setting, 37.3 percent in urban locations, and 13.8 percent in rural areas. Europe had a significantly higher percentage of urban users, at 62.9 percent.

Summary

Effective Web design is about usability. It's about inspiring your visitor to do or feel what you intended. To achieve effective design, you first must understand the origins of the medium, where it's going, and how people are using it.

Chapter 2: Browser Basics

Overview

Although most people use either Netscape Navigator or Microsoft Internet Explorer, many other browsers are available for the Web. In order to design accessible Web pages on your site, you should become familiar with not only the most common browsers but also with any others your audience might be using.

This chapter covers the following topics:

- Browser history
- Browser usage and development
- The browser war and you
- Internet Explorer
- Netscape Communicator
- Lynx
- Opera
- Browser alternatives

A Bit of Browser History

The World Wide Web was begun at CERN (Conseil European pour la Recherche Nucleaire, now known as European Laboratory for Particle Physics). Although the idea for the Web came from many different places, a proposal was initially written and circulated at CERN in 1989. CERN was also responsible for developing the first Web browser, which was released in 1990 (a very long time ago in Web years).

That first browser was developed as both a browser and editor on a NextStep machine. Unlike the browsers with full graphical interfaces that are so common today, this browser only displayed text. Other browsers were soon developed, including Erwise, Viola, and Lynx. Today, many machines are still using these nongraphical browsers, especially Lynx. Lynx is used primarily on Unix and VMS systems, although it is available for all major platforms.

Later, the NCSA (National Center for Supercomputing Applications), based at the University of Illinois at Urbana-Champaign, became involved with the Web and began developing a browser. This browser, Mosaic for X, was released to the public for X, PC/Windows, and Macintosh platforms in September of 1993.

An undergraduate student at NCSA who worked on the project, Mark Andreessen, left NCSA with five others in 1994 to form the Mosaic Communications Corporation, which later became the Netscape Communications Corporation. Netscape released the first version of Netscape Navigator, nicknamed "Mozilla," at the end of 1994.

Meanwhile, it seemed like everyone was getting in on the browser craze. Microsoft developed Internet Explorer (IE), which has become Netscape's main competition. Sun Microsystems developed HotJava, and America Online (AOL) developed their own AOL browser. More recently, a company called WebTV introduced a browser that runs though a television using what is commonly called a set-top box. Other "information appliances" that provide connectivity to the Internet and e-mail without the learning curve required of a "real" computer have followed. Now, even the most technophobic can surf the Web with ease!

Current Browser Usage

The two main browsers in use today are Netscape Navigator and Microsoft Internet Explorer. Because it can be difficult to get accurate numbers, individual market share is very difficult to gauge. Browser Watch, a site popular with those keeping up with browser statistics and other Internet-related topics, reports that 58 percent of its visitors used Internet Explorer in the month of October 2000, and 22 percent used Navigator. The remaining 30 percent included upcoming browsers, such as Opera and NeoPlanet, or specialized versions of IE and Navigator that are recorded as a unique browser type. In the general population, the estimate for IE and Navigator use is higher, ranging between 80 and 90 percent.

If, as a Web site designer, you assume 90 percent IE and Navigator usage, it's still important to consider the other 10 percent. Although 10 percent does not sound like much, so many people are on the Web that 10 percent could be a hefty number. For instance, if your potential audience is 10 million users (even more people than that are on the Web today), 1 million potential users will be locked out of your site if you don't accommodate them. Many of the features found in the two most popular browsers are also found in some of the less common browsers, such as Opera. But the features don't perform as well if the sites have been "designed for" IE or Navigator. Other browsers, such as Lynx, only handle the more basic elements and attributes of XHTML.

But different browsers are only part of your challenge. Along the way, browsers have been released in different versions, with each release supporting new features. When designing a Web site, it is important to realize that while the vast majority of your audience may be using Navigator or IE, a significant portion of them will not be using the latest version with the latest features. Even though it is fairly easy to get the latest version, browsers are released so often and so frequently contain bugs that many people are satisfied staying with their current setup, or they may be operating under rules set down by a corporate IT department about specific software usage. Also, new browser versions often require more computing power, and some people simply can't or won't upgrade their machines.

Testing, One, Two, Three…

Most software companies undertake a period of end-user testing known as *beta testing*. A beta test is the second round of testing for a product (with *alpha* testing normally being an internal test by company staff members). Microsoft and Netscape both release beta versions of their software to the general public. Many bugs and problem areas are resolved during beta testing, and improvements are incorporated from user suggestions.

Many Web developers love to live on this "bleeding edge" of cutting-edge technology and download the new betas the minute they're available. Remember, though, that new features found in the betas are often not supported by earlier versions of the browser, and the general public is traditionally far slower to upgrade than the hard-core Net community.

The interfaces for IE and Navigator have begun to resemble each other over the years; they have the same basic icons and functionality available in the first quarter or so of the screen. Figure 2.1 shows the Navigator 4.76 interface, and Figure 2.2 shows the IE 5 interface; both are shown on the Windows 98 platform. If you only have one browser on your machine now, consider downloading at least the latest versions of Navigator and Internet Explorer, if not one or two of the lesser-used browsers. When visiting sites mentioned in upcoming chapters, look at them through each browser. See if you can detect any subtle differences in the way the browser renders the page.

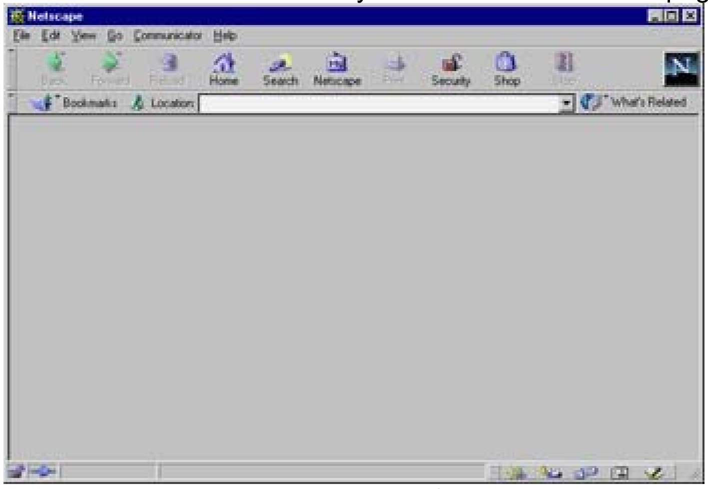

Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: What's Related

Figure 2.1: The Navigator 4.76 interface

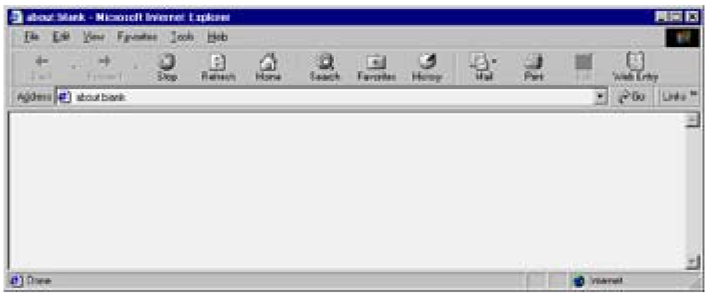

about:blank - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Web Entry
Address about:blank
Go Links
Done
Internet

Figure 2.2: The Internet Explorer 5.5 interface

Tip *Netscape Navigator can be downloaded from* http://home.netscape.com/download/index.html. *Microsoft's download page for Internet Explorer can be found at* http://www.microsoft.com/ie/download/.

The Browser War

Companies such as Netscape and Microsoft release new versions of their software on a seemingly constant basis. Whereas other types of software updates can take years, new browsers are released every few months. Probably the biggest reason for the frequent updates is what is commonly referred to as the *browser war*.

What Does That Mean?

The browser war is a race between companies (namely Netscape and Microsoft) to establish market dominance for their browser, often by being the first to release new features that work only with their particular browser.

The idea is that people (such as you) who develop Web sites will want to use a great new HTML element even though it's only supported by one of the browsers—let's say IE. In order to take advantage of the element, you'll attempt to get your users to use IE, perhaps by providing a link so they can download it.

If the browsers are often very inexpensive or even free, then why is everyone clamoring to get their browser on your desktop? The browser companies make other software, such as server software, which is much more expensive. They are counting on the name recognition from all the free or low-cost software to propel buyers to the higher-priced purchases.

WWW Another issue in the browser war is that of standards. In the software industry, a push usually exists to develop standards, or protocols, that similar pieces of software all use. In the Web industry, the standards are often about HTML, and now XML and XHTML. The World Wide Web Consortium (`http://www.w3.org`) is responsible for setting the standards for these languages. Browser companies frequently released HTML elements that were not official HTML elements as defined by the W3C. These browser-specific HTML elements were dubbed *HTML extensions*.

The setting of standards in software is not a one-way street, as if the World Wide Web Consortium decides the standards and everyone goes from there. Rather, it is back and forth. Extensions from companies such as Netscape and Microsoft are so commonly used that the software companies have a hand in setting the standards, not just following them. In Table 2.1, you'll see examples of the browser companies and the World Wide Web Consortium working together to set standards.

Table 2.1: A Brief History of Browser Development

DATE	DEVELOPMENT
October 1994	Netscape introduces Netscape Navigator 1, free to users via the Internet.
March 1995	Netscape Navigator 1.1 is announced. It includes support for advanced layout capabilities using HTML 3 tables and graphical backdrops. These capabilities allow more sophisticated page presentation, including multiple text columns and flexible image placement.
September 1995	Netscape introduces Netscape Navigator Gold 2, which enables users to easily create, edit, and navigate live online documents. Naturally, the editor that's a part of the Gold package supports the Netscape extensions. Capabilities include support for frames; a

Table 2.1: A Brief History of Browser Development

DATE	DEVELOPMENT
	page-presentation capability that enables the display of multiple, independently scrollable cells on a single screen; and HTML 3 tables and backdrops.
January 1996	Netscape announces *plug-ins*, which are small pieces of software that enhance browser capabilities.
April 1996	Netscape announces Netscape Navigator 3 beta. It supports new HTML tags, including background colors in tables and audio and video embedding functions.
	Microsoft Internet Explorer 2 for Macintosh is released. It supports the Shockwave plug-in, HTML 2 and 3 tags, QuickTime, and Virtual Reality Modeling Language (VRML).
May 1996	Microsoft Internet Explorer 3 beta debuts.
June 1996	Netscape announces that more than 130 plug-in developers are creating plug-ins to work with Netscape Navigator. It also announces that its Internet site receives more than 80 million hits a day and has accumulated a total of more than 10 billion hits since its inception.
	Microsoft's team continues the furious pace of development of IE 3, which will introduce extensibility through ActiveX controls.
August 1996	Netscape announces the availability of Netscape Navigator 3, which supports both Java and JavaScript. Several third-party developers make public plans to develop plug-ins to take advantage of Netscape Navigator 3 functionality. A total of 175 plug-ins are already announced for Netscape Navigator.
	Microsoft launches Internet Explorer 3. Top Web sites offer free content that can only be viewed by users with Internet Explorer 3.
October 1996	Netscape announces Netscape Communicator, which integrates Netscape Navigator 4 browser software, Netscape Composer HTML authoring software, Netscape Messenger electronic mail, Netscape Collabra group-discussion software, and Netscape Conference real-time collaboration software.
	Netscape announces Netscape Navigator 4, which includes support for absolute positioning, layering and style sheets, new HTML fonts for authoring, and support for Netscape ONE (the open network environment).
January 1997	Microsoft ships the final Internet Explorer 3 for Macintosh. Microsoft Internet Explorer 3 offers full support for HTML 3.2, tables, frames, and enhanced frames (borderless and floating). With this version, Internet Explorer becomes the first browser to allow Macintosh users to view Web pages created using the HTML standard Cascading Style Sheets.
April 1997	Microsoft announces Microsoft Internet Explorer 4. Improved style-sheet support, Dynamic HTML, and the Active Desktop are touted as the new wave in browsing.
July 1997	Microsoft endorses the World Wide Web Consortium's HTML 4 and announces support in Microsoft Internet Explorer 4.
January 1998	Netscape announces open-source access to its full Communicator 5 product.
July 1998	Netscape Communicator 4.5 is released for beta testing.

Table 2.1: A Brief History of Browser Development

DATE	DEVELOPMENT
January 1999	Microsoft Internet Explorer 4.5 for the Mac debuts at the Macworld show.
March 1999	Microsoft ships Internet Explorer 5 for Windows.
December 1999	W3C releases errata update to HTML as HTML 4.01
January 2000	XHTML 1 becomes an official W3C recommendation.
March 2000	IE 5 for the Mac ships after several developmental delays.
April 2000	Netscape allows a first look at Communicator 6, which is based on the Mozilla open-source project.
July 2000	Microsoft releases IE 5.5 for Windows with improved CSS support and new support for SMIL (Synchronized Multimedia Integration Language), as well as additional proprietary Dynamic HTML features.

From a software company's point of view, the advantage of setting standards is that they will have been implemented internally, whereas your competition will have to catch up. The downside, however, is that the W3C may decide against an implementation of a new feature, thereby not providing support. The emergence of the idea of dynamic HTML in late 1997 and positioning methods used with Cascading Style Sheets (CSS) (see Chapter 7) is one such example: Netscape's layer method was rejected by the W3C in favor of positioning through CSS.

As a Web designer, you have a responsibility to keep up with which elements are "official" and which are browser-specific extensions. Always keep in mind that reliance on such "browserisms" can literally keep visitors from being able to view your site—and nobody wants that!

Browser Developments

So what about the developments in the browser war? (And why should you care?) The developments are so important because they vastly affect your ability to design effective, accessible Web sites. It is the business of every professional Web designer to stay on top of browser developments and how they are going to affect Web design. Table 2.1 shows a timeline of browser developments that have already taken place.

How Do Browser Developments Affect the Users?

Now that you've looked at the history of browser developments, you can probably see how such developments affect your potential Web sites. I'll take a look at the ways the browser war and the general evolution of Web standards affect your work, both positively and negatively.

The Upside

The browser war results in newer, better features that you can implement on your Web sites. The blistering pace of development leads to regular updates and bug fixes in browsers, as well as timely changes in the user interface, brought about by consumer comments. If you've spent much time surfing the Web, you've probably seen excellent uses for advanced HTML capabilities, such as tables, frames, and forms. The features allow users to view information in columns or cells, navigate sites using toolbars, and give and receive feedback.

You've probably also seen cool implementation of Java and animation—such as pop-up information when someone mouses over an image, the calculation of mortgage loans, and interactive games presented in Shockwave or Flash from Macromedia. Used in moderation, such advances can add much to the user's Web experience.

Tip Browser releases frequently contain features that enhance the user's experience but do not really affect how you design a Web site. One example is the ability to read e-mail from within the browser.

As the software companies continue to try and outdo each other, both in market share and in the standardization process, end users often benefit. But this is not the case for every user, especially not for those who do not keep up with the latest software. That said, I'll take a look at the challenges provided by the browser war.

The Downside

As I've mentioned, the main challenge for the Web designer is designing a Web site that works for different browsers. To this end, the designer should keep several items in mind:

- The latest official World Wide Web Consortium HTML standards, including HTML 4.01, XHTML 1, and CSS2.
- Which features are supported by which browsers—not only the two main browsers but also the other browsers such as Opera, NeoPlanet, Lynx, WebTV, and wireless devices.
- Which features are supported by which *versions* of the browsers. An early version of a browser does not support all of the features supported by the latest version.
- Which types of browsers your audience will be using. If you are going to have a general-interest or commercial Web site freely available to the public, you can expect all kinds of browsers. If you are designing a site that will only be available to people in your company, all of whom use Netscape Navigator 4.5 for Windows, you might be able to get away with a bit more.
- Font design. Depending on how you choose to design your site, your audience can have the choice of which fonts and font sizes they use to view your site.
- Image-free viewing. The audience may choose to come to your site with images turned off.
- Whether the feature that requires certain browser capabilities is really worth the trouble. That is, whether the feature adds enough value to the site to justify locking out users whose browsers do not support it.
- Whether another way supported by more browsers exists to achieve a desired result. (You may not always come up with an alternate method, but it is good practice to try.)
- The likely speed of the connection your viewers will use. I haven't discussed speed, but it helps to know if many of your users are connecting via a 56Kbps modem (which is likely if your Web site is publicly accessible) or via a T1 connection (which may be the case over a closed, corporate intranet).

Although it may seem daunting, staying on top of these issues helps you make informed decisions for your design. Once you start considering these factors, it becomes second nature. Further, once you're caught up with the basics (hopefully, by reading this book), you just need to *maintain* your knowledge base.

Browser Security Issues

Security, though it does not affect your design as directly as other aspects, is nevertheless certainly worth mentioning. Frequently in the race to get browsers out the door, companies do not adequately test the security of their browsers. Usually, if you wait a few weeks after a major browser release, someone will find a security problem with the browser, and the company will issue a patch. Recently, security problems have related to code that downloads and executes on the user's computer, such as Java and ActiveX controls. Although patches have been released, these particular problems are key examples of how, in the race to beat the competition, software companies do not always test adequately.

As you've seen, you need to keep many points in mind about browsers when you design a Web site. As an overview, I am going to look at the major features of browsers in the last part of this chapter.

Extensibility through Modularization
You learned in Chapter 1 that XML allows authors to create their own vocabularies of elements and attributes. You also know that HTML provides easy-to-use constructs, such as paragraphs, headings, lists, and tables. Who should be responsible for providing support for the custom features? Should you lobby Microsoft or Netscape to include rendering instructions for your new elements? Should you write to the W3C and ask them to add it to HTML? Neither of those possibilities is very satisfying nor likely to succeed, because the number of custom elements and attributes the Web-authoring public may want can easily number in the thousands.

You can certainly write your document in XML and display it using an XML parser, but if you only need two or three new elements, why reinvent the wheel in XML for the rest of HTML's features?

XHTML is intended to provide a solution. The W3C is taking the work done on XHTML 1, bringing HTML into XML syntax, and expanding on XHTML 1 with the ability to append custom elements and attributes. The ability to append custom elements is facilitated through a process known as the *Modularization* of XHTML. The W3C broke down all the elements in XHTML into logical groupings, such as text-presentation elements or table-related elements. Each of those groups is known as a *module*. It also provided a method for users to create their own modules for the custom elements and attributes they desire. By combining both the modules provided by W3C and the ones you've built yourself, you will create the desired extensibility of HTML. So you needn't wait for a vendor to provide it for you, which would inevitably result in proprietary elements (such as Netscape's layer element or IE's marquee) that can't be used across all browsers. With an XHTML module, a style sheet for proper presentation instructions, and any browser with an XML parser built in (which has become increasingly popular), your documents with new elements can be viewed by anyone, which leads you to the land of an interoperable Web.

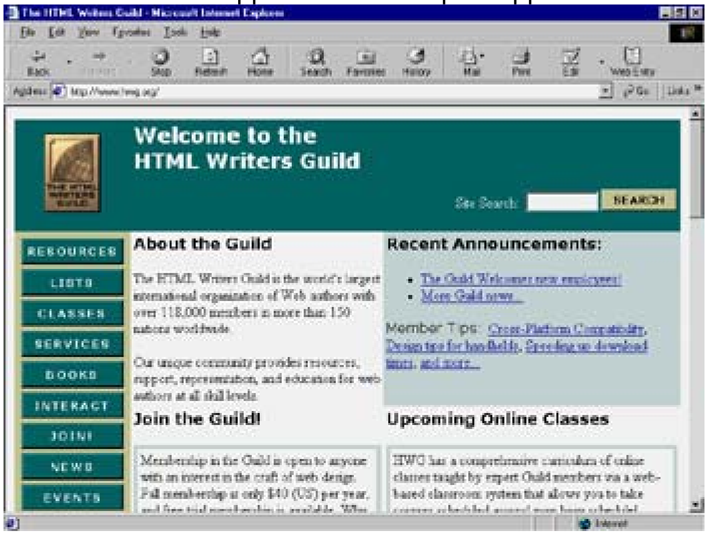
A Quick Tour of Today's Browsers

Sometimes the biggest battle, especially for the beginner, is figuring out what the browsers are capable of and what exists besides the Big Two (IE and Navigator). In this section, I'll take a quick look at IE and Navigator, as well as several of the other browsers.

Microsoft Internet Explorer

Arguably the most popular browser in use today, IE has gained dominance through inclusion with and integration of the Windows operating system. Microsoft was a later contender as a browser vendor, but with the automatic delivery of Internet Explorer with Windows 95, it has quickly caught up with and in some quarters surpassed Netscape, which, as recently as 1998, was the most popular browser on the Web. Microsoft Internet Explorer is currently in version 5.5, with Netscape Navigator having just been released in version 6.

WWW IE has just about the best support for CSS on the market today. Sophisticated designs can be constructed entirely with style rules, as seen on the HTML Writers Guild's (HWG) Web site (`http://www.hwg.org`). Figure 2.3 shows their home page. The buttons seen running down the left-hand side of the screen aren't graphical buttons at all; the colors, bevels, and shading were all produced with CSS. IE's support for CSS is quite apparent in version 5.5.


The HTML Writers Guild - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Web Entry
Address http://www.hwg.org/
Go Links
Welcome to the HTML Writers Guild
Site Search: SEARCH
RESOURCES
LISTS
CLASSES
SERVICES
BOOKS
INTERACT
JOIN!
NEWS
EVENTS
Internet

About the Guild

The HTML Writers Guild is the world's largest international organization of Web authors with over 118,000 members in more than 150 nations worldwide.

Our unique community provides resources, support, representation, and education for web authors at all skill levels.

Recent Announcements:

- The Guild Welcomes new employees!
- More Guild news...

Member Tips: Cross-Platform Compatibility, Design tips for handhelds, Speeding up download times, and more...

Join the Guild!

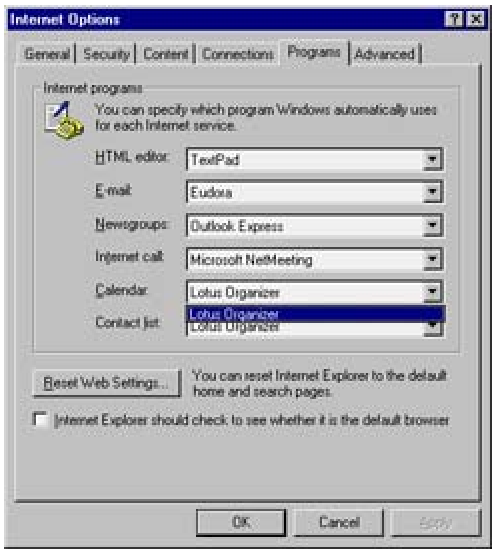
Membership in the Guild is open to anyone with an interest in the craft of web design. Full membership is only $40 (US) per year, and free trial membership is available. Who

Upcoming Online Classes

HWG has a comprehensive curriculum of online classes taught by expert Guild members via a web-based classroom system that allows you to take courses scheduled around your busy schedule!

Figure 2.3: IE 5.5 on Windows 98 shows its support for CSS.

One of the more useful features of IE 5.5, in my opinion, is the ability to set your preference for which program to use for tangentially related tasks, such as e-mail, Web-page editing, and contact management. These options are found in IE's Internet Options dialog box. To access the dialog box, choose Tools → Internet Options and then select the Programs tab (see Figure 2.4). I have two complaints, though, about this feature: not being able to link Act! 2000 (my choice of personal information-management software) to the calendar and contact-management functions and not having a Browse option to select a program not in the drop-down list.



Internet Options

General | Security | Content | Connections | Programs | Advanced

Internet programs

You can specify which program Windows automatically uses for each Internet service.

HTML editor: TextPad

E-mail: Eudora

Newsgroups: Outlook Express

Internet call: Microsoft NetMeeting

Calendar: Lotus Organizer

Contact list: Lotus Organizer

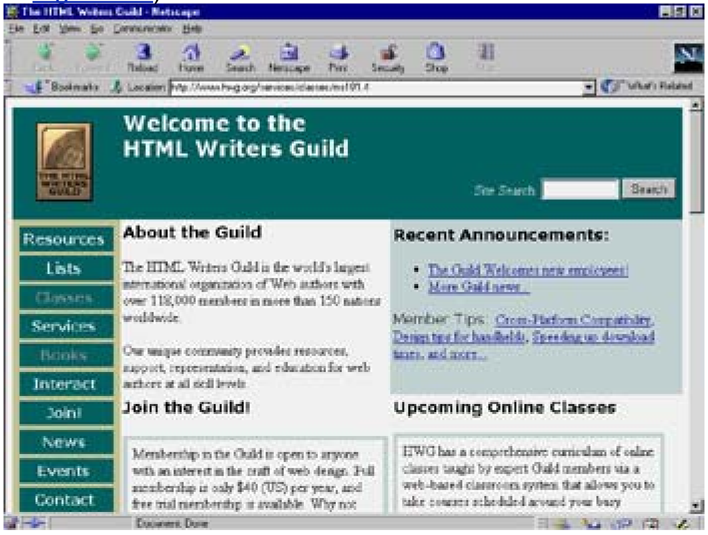
Reset Web Settings... You can reset Internet Explorer to the default home and search pages.

Internet Explorer should check to see whether it is the default browser

OK | Cancel | Apply

Figure 2.4: The Internet Options dialog box in IE 5.5

Netscape Navigator
As noted in Table 2.1, in 1996 Netscape began packaging their Web browser Navigator as part of a larger integrated application suite known as Netscape Communicator. Today's Netscape Communicator 4.76 package includes Netscape Radio, a rebroadcast of content from Spinner of 10 streaming channels of different musical genres. Netscape Radio is accessible by selecting Communicator → Radio menu from the main toolbar. Another new addition is the Shopping button on the primary toolbar; it links the user to Netscape's new shopping portal, <Shop@Netscape>. Ancillary applications include Netscape AOL Instant Messenger, WinAmp (an MP3 player), and the Macromedia Flash 4 plug-in. Navigator's handling of CSS falls short of the standard set by IE, primarily in the CSS Level 2 functionality. What follows is another look at the HWG home page, this time using Navigator 4.76 (see Figure 2.5).



Welcome to the HTML Writers Guild

About the Guild

The HTML Writers Guild is the world's largest international organization of Web authors with over 118,000 members in more than 150 nations worldwide.

Our unique community provides resources, support, representation, and education for web authors at all skill levels.

Recent Announcements:

- The Guild Welcomes new employees!
- More Guild news

Member Tips: Cross-Platform Compatibility, Design tips for handhelds, Speeding up download times, and more...

Join the Guild!

Membership in the Guild is open to anyone with an interest in the craft of web design. Full membership is only $40 (US) per year, and free trial membership is available. Why not

Upcoming Online Classes

HWG has a comprehensive curriculum of online classes taught by expert Guild members via a web-based classroom system that allows you to take courses scheduled around your busy

Figure 2.5: A CSS-heavy site as viewed in Navigator 4.76

To Netscape's credit, the site does degrade gracefully when it doesn't know how to handle a given CSS property or HTML element. So you have the same colors and shapes in the left-hand navigation bar when the site is viewed in Navigator, but you lose the shading and beveling that adds the graphical button look.

Netscape Communicator 6
In 1998, Netscape opened up the source code to Navigator to the general public for experimentation and improvement. The effort became known as the Mozilla project. Netscape 6 uses as its core the work produced by the Mozilla Project and has a stated goal of being standards compliant and forward looking in its rendering of HTML and XHTML documents. It doesn't even support some of its own HTML extensions that were never adopted by the W3C. The look "straight out of the box" is quite foreign to what you're used to seeing from Netscape (see Figure 2.6).

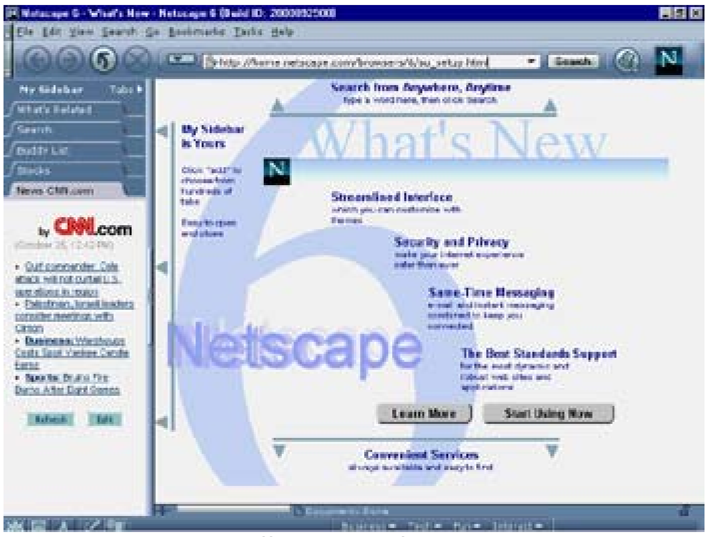


Figure 2.6: A quite different look for Navigator 6

Most of the top menu bar has remained the same, but the iconography in the toolbar is definitely different, as is the presence of the My Sidebar feature running down the left side of the screen. The sidebar can be closed by clicking the arrow in the middle of the separator bar (see Figure 2.7).

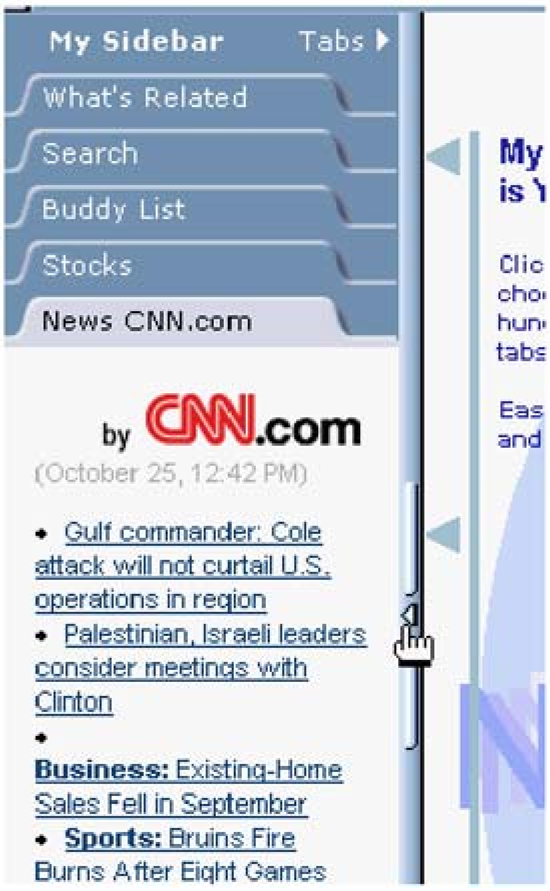
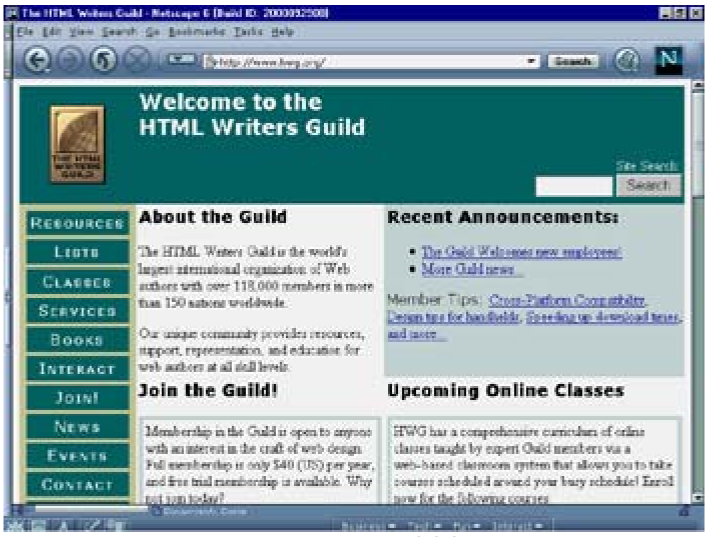

My Sidebar
Tabs
What's Related
Search
Buddy List
Stocks
News CNN.com
by CNN.com
(October 25, 12:42 PM)
Gulf commander: Cole attack will not curtail U.S. operations in region
Palestinian, Israeli leaders consider meetings with Clinton
Business: Existing-Home Sales Fell in September
Sports: Bruins Fire Burns After Eight Games

Figure 2.7: Closing down the My Sidebar feature

Netscape retains the familiar ancillary programs of e-mail, Composer, and Instant Messenger. A new service, Net2Phone, purports to provide easy Net-to-telephone access for rates lower than traditional long-distance telephone rates. Such activity is likely to be highly subjected to the ups and downs of Internet traffic.

I'll take a look at how Navigator 6 deals with the HWG Web site (see Figure 2.8).


The HTML Writers Guild - Netscape 6 [Build ID: 2000092500]

http://www.hwg.org/

Welcome to the
HTML Writers Guild

Site Search:

Search

RESOURCES
LISTS
CLASSES
SERVICES
BOOKS
INTERACT
JOIN!
NEWS
EVENTS
CONTACT

About the Guild

The HTML Writers Guild is the world's largest international organization of Web authors with over 118,000 members in more than 150 nations worldwide.

Our unique community provides resources, support, representation, and education for web authors at all skill levels.

Recent Announcements:

- The Guild Welcomes new employees!
- More Guild news...

Member Tips: Cross-Platform Compatibility, Design tips for handhelds, Speeding up download times, and more...

Join the Guild!

Membership in the Guild is open to anyone with an interest in the craft of web design. Full membership is only $40 (US) per year, and free trial membership is available. Why not join today?

Upcoming Online Classes

HWG has a comprehensive curriculum of online classes taught by expert Guild members via a web-based classroom system that allows you to take courses scheduled around your busy schedule! Enroll now for the following courses

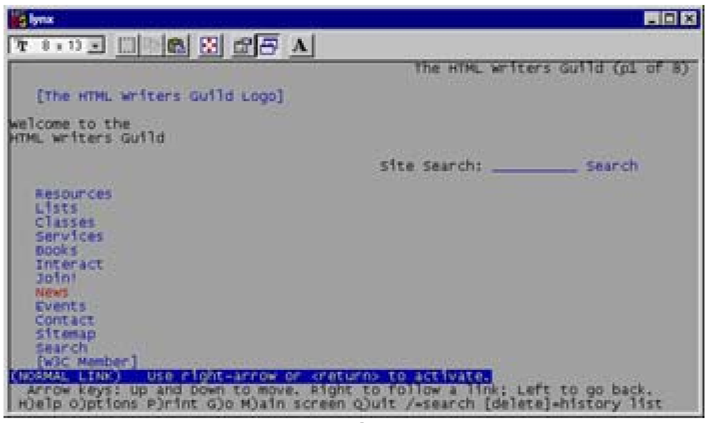
Figure 2.8: Navigator 6 viewing a CSS-heavy site

Despite Navigator 6's claims of superior standards support, the left-side navigation bar is still flat looking with no bevel or shading effects. Otherwise, the presentation remains about the same as that found in Navigator 4.76.
One nice feature of Navigator 6 is the ability to apply *themes* (which are becoming commonly known as *skins*) to the interface. For those of you too used to the traditional Navigator interface to be comfortable with its new modern look, Netscape has provided a classic theme that can be set by selecting View → Apply Theme menu. Navigator 6 ships with the default look, labeled Modern, and the look you're used to, labeled Classic. Plans for a Netscape Theme Park online with user-submitted themes are in the works.

Lynx

Lynx was developed at the University of Kansas to be used on its own system. The current version is 2.8.2. Lynx is an older, "bare-bones," text-based browser and has the advantage of running on older systems. It is used mostly on Unix and VMS systems, although you can get it for other platforms. It is particularly popular at universities and colleges, where students can often have a Unix shell account.

Lynx is also popular with some people with visual impairments because it can be configured as a text reader to read lines from Web pages. When a system is configured with the proper audio hardware, called a speech synthesizer, it can read out loud to the user. Additionally, many heavy Web users will fire up Lynx when they're looking for text-based information content in a hurry. Because it doesn't load images and many other large file-size objects, sites can be rapidly skimmed for relevant content.
Lynx is a text-only browser, so if you want to accommodate Lynx users, you'll have to provide alternatives to images, or let Lynx users download them. I'll go over alternative text for images later in Chapter 5.
Lynx is a perfect example of something you see frequently in the software world—a trade-off between features (such as images, formatting, and frames) and system requirements (such as platform issues and RAM). Although Lynx does not support many of the newer features, you also don't need a ton of RAM and a newer machine to run it. With Lynx's command-line interface, it can take some getting used to if you are accustomed to pointing and clicking. You can see the Lynx interface in Figure 2.9, which also shows how it handles the CSS-heavy HWG Web site.



```
                                          The HTML Writers Guild (p1 of 8)

   [The HTML Writers Guild Logo]

Welcome to the
HTML Writers Guild

                                       Site Search: ________ Search

   Resources
   Lists
   Classes
   Services
   Books
   Interact
   Join!
   News
   Events
   Contact
   Sitemap
   Search
   [W3C Member]
(NORMAL LINK)   Use right-arrow or <return> to activate.
  Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figure 2.9: Lynx and the HWG Web site

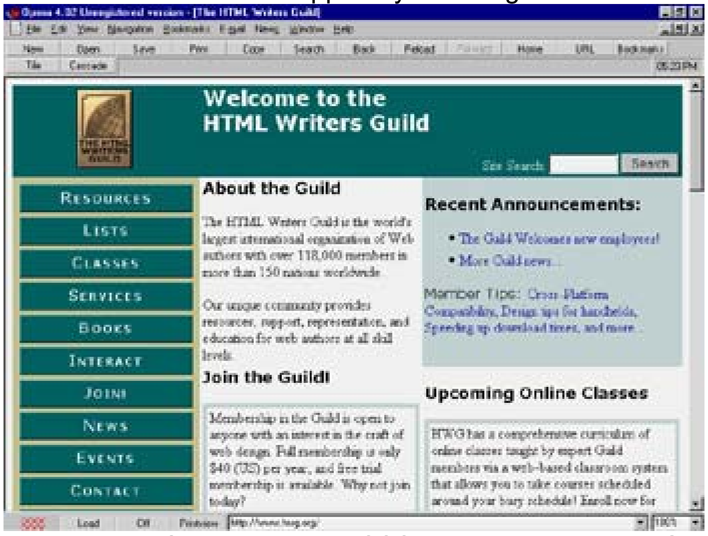
Lynx displays the alternative text for the logo and the search box across the top of the page. Because the rest of the site is built in a two-column table, it presents the contents of the left-hand cell first, which is the navigation menu. Scrolling down, Lynx will "page" through the rest of the document.

Opera 4.02

Opera is one of the few entries into the browser market that still carries a price tag. However, many devotees feel that it's worth its $39 price tag many times over. Opera Software, the makers of the Opera browser, scored a bit of a coup when they hired Håkon Wium Lie away from the W3C to serve as Chief Technology Officer. Håkon has been involved in the Web since the early days at CERN and helped the W3C formulate the idea of Cascading Style Sheets. That Opera is gaining a user base despite its price tag speaks volumes about its features and ease of use. Tech-savvy users recognize the quality put into the product at the behest of Chief Technical Officer Lie.

WWW With Lie's experience, it's understandable that Opera takes Web standards seriously. A full run-down of its support for W3C standards can be found online at `http://www.opera.com/opera4/specs.html`.

I'll take a look at that support by revisiting the HWG home page one more time in Figure 2.10.


Opera 4.02 Unregistered version - [The HTML Writers Guild]
File Edit View Navigation Bookmarks E-mail News Window Help
New Open Save Print Copy Search Back Reload Forward Home URL Bookmarks
Tile Cascade
05:23 PM
Site Search
Search
Resources
Lists
Classes
Services
Books
Interact
Join!
News
Events
Contact
Load
Off
Windows
http://www.hwg.org/
100%

Welcome to the HTML Writers Guild

About the Guild

The HTML Writers Guild is the world's largest international organization of Web authors with over 118,000 members in more than 150 nations worldwide.

Our unique community provides resources, support, representation, and education for web authors at all skill levels.

Join the Guild!

Membership in the Guild is open to anyone with an interest in the craft of web design. Full membership is only $40 (US) per year, and free trial membership is available. Why not join today?

Recent Announcements:

- The Guild Welcomes new employees!
- More Guild news...

Member Tips: Cross-Platform Compatibility, Design tips for handhelds, Speeding up download times, and more...

Upcoming Online Classes

HWG has a comprehensive curriculum of online classes taught by expert Guild members via a web-based classroom system that allows you to take courses scheduled around your busy schedule! Enroll now for

Figure 2.10: Opera takes the CSS challenge at the HWG

Browsing on Alternative Devices

The general public has been inundated with messages about accessing the Web "on the go." Palm Pilots or other hand-held devices are now de rigueur for the well-connected executive. Cell phones and even devices in cars are getting into the act. A television commercial from a major computer company shows a young man sitting in a piazza in Italy managing his investments through a computer viewed from a tiny device worn on his head that projects an image in front of him and responds to voice commands.

The future of such access is here. From my cell phone, I can check movie times for theaters in any zip code in the United States. The information that can be displayed on these devices is limited, to say the least, but advances in delivery and display are occurring exponentially. Throughout this book you should keep these alternative access methods in mind as you learn about XHTML. Web designers will soon find themselves having to deal with a quickly increasing alternative-access population.

The bevels and shadows are back in the left-hand navigation column, though it's significantly wider in size than in most other browsers—a curious side effect that might deserve some looking into by the site manager.

Summary

WWW Whew! And to think I only covered a handful of the browsers available! It is important to remember that literally dozens of additional browsers are in use by more than just a few on the Web. You can experiment with many of these by downloading them from CNet's Browsers.com site, at `http://www.browsers.com`.

You've covered a great deal in this chapter, including:

- How browsers have changed and evolved
- How changes in browsers affect you and your ability to design effective Web sites
- How to make informed decisions about which features to include or exclude from your Web sites

Chances are you'll refer back to this chapter as you design your Web sites—don't try to remember it all. Just know that the many types of browsers have different capabilities. But fear not—as you progress through the book, you'll learn tips and tricks to get around the challenges that the plethora of browsers presents.

Chapter 3: Designing with the XHTML Document

Overview

An XHTML document provides the structure for your Web page. In this chapter, you'll work with and create a basic XHTML document. First I'll discuss XHTML editors and how to select one, and then I'll move on to the actual document. I'll cover several types of XHTML elements and how they are constructed, along with any options you have for their use.
The basic structure of the document is set up with the `DOCTYPE` declaration and the html and body elements. The elements nested within the head element describe your document.

This chapter covers the following topics:

- Selecting an editor
- XHTML must haves
- Creating an XHTML document
- Describing your document with title and meta elements
- Defining text and background looks using the body element
- Headings and text placement

Selecting an Editor

Before you create your first XHTML document, you need to select a text editor to work in. You can choose from many; which one you end up using is a matter of personal preference and availability.

Some markup purists insist that no "helping features" be available in an editor, as they consider it to be the only way to acquire markup skills. So they use either Notepad on Windows or the vi editor for Unix. I don't go that far.

What to Use
Literally dozens of HTML and XHTML authoring tools are on the market today in retail, shareware, or even freeware format. These may simply be enhanced text editors that automatically highlight your XHTML tags in various colors or make a spell checker available to you, or they may provide built-in macros to handle repetitive markup tasks. Quite a few of the authoring tools profess to provide WYSIWYG control over your layout—that is, "what you see is what you get." However, that really should be amended to "what you see is what you *and only you* get."
As you learned in Chapter 2, each browser version may display document elements in a different manner. Sometimes in a *very* different manner! So you know that a Pentium III-powered PC running Windows 2000 and Netscape Navigator 6 will present a very different picture than the old 486 running Windows 95 and Microsoft Internet Explorer 3. So how can a tool guarantee that everyone will see your page the way you do? For the most part, *it can't*—at least not completely.

Now, no one is purporting that the creators of these tools have developed a group case of dottiness. To the contrary, many of these tools can be very handy, when—and only when—the user is aware of how the programs produce the Web page. Do they produce markup that makes use of browser-specific tags? Do they heavily rely on tables for layout? Will they allow you to tweak the code yourself?

Keep this point in mind: Before you can use authoring tools intelligently, you must have a sound comprehension of the underlying XHTML processes. Each skill in this book will provide you with the background behind each element, in order to help you choose the best possible presentation for your projects. Once you become experienced in making the choices, you may pick an authoring tool that saves you time by automating some choices and processes for you. Until then, it's best to follow the tradition of "learning by doing."

Editors Available for the PC

Many editors are available for the PC. Some offer only the most rudimentary of features, whereas others offer color syntax highlighting and other "helper" features that some authors find useful in their everyday work. The primary purpose of choosing a text editor over any form of WYSIWYG authoring tool is for the direct control you have over the structure of your documents, both HTML and XHTML. Eventually, text editors will be able to deal with CSS and other related files. The following is a description of several commonly used editors for the PC:

- **TextPad** A true favorite! TextPad can be customized to behave like familiar programs such as Microsoft applications, WordStar, BRIEF, and others. It provides integrated spell checking, a powerful search-and-replace utility, sticky indents, macro recording, and an innovative "clip library" for inserting frequently used text such as HTML tags. The most recent version from Helios Software Solutions, as this book went to print, was version 4.3.2. This product has minimum system requirements of Windows 95 or Windows NT 3.51. It is also supported on Windows 98, Windows Me, Windows NT 4, and Windows 2000. It is shareware, and you can find it on the Web at `http://www.textpad.com`. (See Figure 3.1.)

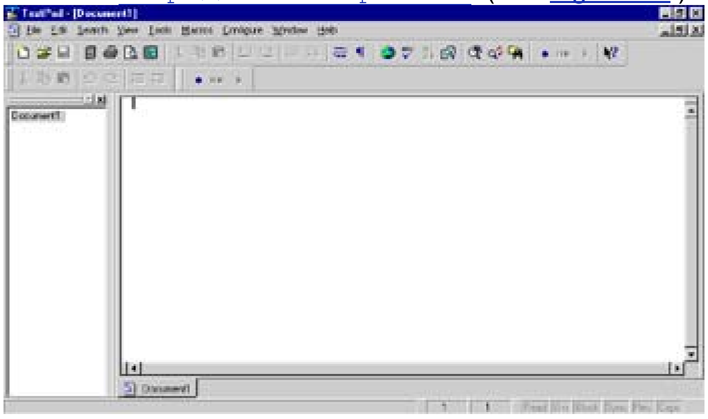


Figure 3.1: The Text Pad 4.3.2 interface (shown in Windows 98)

- **Allaire HomeSite** Allaire scores high marks with its popular HomeSite 4.5 HTML editor. As with other programs designed for use with HTML, HomeSite adapts quickly for XHTML use. The default document template is currently set to provide the HTML 4 Transitional `DOCTYPE` declaration. However, you can easily rectify the setting by entering the appropriate XHTML `DOCTYPE` declaration, saving a similar blank document as a new template, and then directing HomeSite to use the XHTML version as its default template (see Figure 3.2).

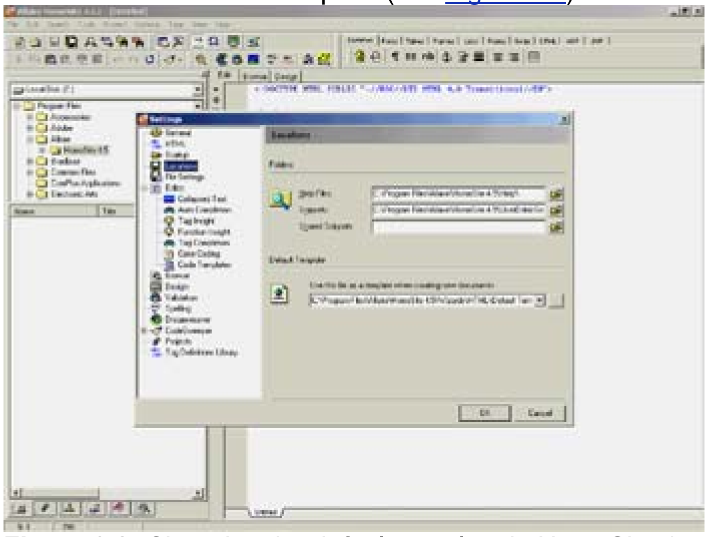


Figure 3.2: Changing the default template in HomeSite 4.5

- **Notepad** The Microsoft Windows classic, Notepad, is about as basic as it gets, with the capability for search, cut and paste, and word wrapping. Notepad ships and installs with all standard Windows setups (from early Windows 3.1 all the way through Windows Me and Windows 2000).

Editors Available for the Mac

The same variety of editors available for the PC can be found for the Mac. Some programs have both PC and Mac versions; others were created expressly for the Macintosh. Either way, your desire is the same: to have complete control over the characters and structures placed in your XHTML documents. Here are a few of the more popular Mac editors:

- **BBEdit** The 1996 winner of the *MacUser* Editors' Choice Award (the magazine is no longer in circulation) and currently a nominee for Editor's Choice Award at the

MacWorld show in early 2001, this program includes a powerful search-and-replace mechanism, customizable behavior settings, macros, and much more. BBEdit is the standard in Mac-based HTML editing. It is available from Bare Bones Software and works with System 7 or later. (See Figure 3.3.)

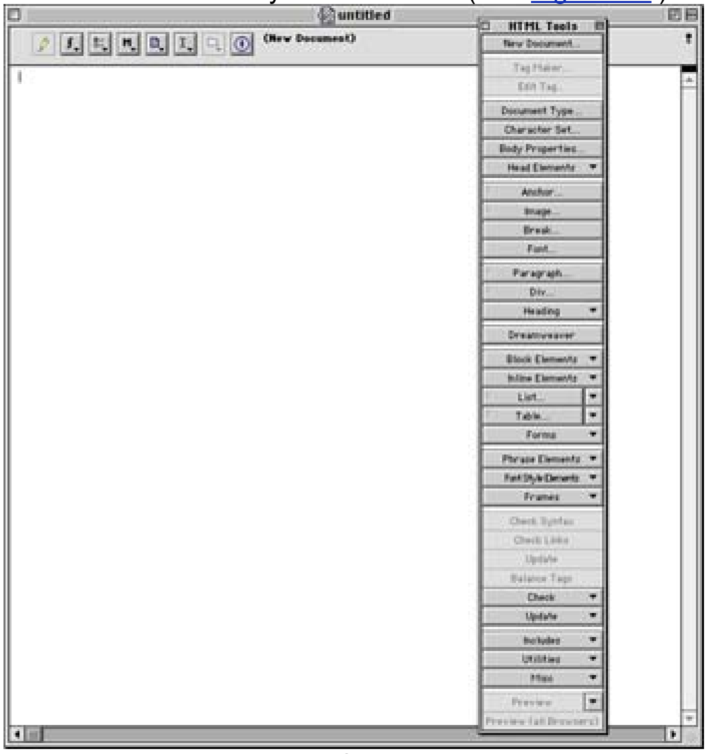


Figure 3.3: The BBEdit interface on a Mac

- **Text-Edit Plus** When you need a more powerful text editor that includes drag-and-drop, multifile search and replace, and even Speech Manager support, this shareware product from Trans-Tex Software fits the bill. You'll need System 7 or later to run it.
- **SimpleText** This classic Mac text editor is included with your OS software. File sizes are limited, which may inhibit your ability to Web edit with this program. SimpleText is published by Apple Computer, Inc. With the new Mac OS X, the SimpleText editor has been replaced by a new program called TextEdit. It has similar functionality and should serve Mac users just as well as SimpleText has for years.

Tip A slew of other editors is available. Two good resources for shareware, freeware, and product demos are Tucows (The Ultimate Collection of Windows Software) at http://www.tucows.com and CNet's Download.com at http://www.download.com.

In the Beginning

Every XHTML document has several components that it must have in order to be processed correctly: the `DOCTYPE` declaration and the elements `html`, `head`, `title`, and `meta`. I'll look at each of them in turn. In the days of HTML, when those components were optional, some went so far as to say that a document wasn't an HTML document without them—much like a sentence isn't complete without punctuation and at least a noun and a verb. XHTML removes nearly all of the optional facets of the markup, such as the previously optional closing `</p>` tag. The removal of the ambiguities of the structure means that you can be confident your XHTML document is fully compliant.

To begin creating your XHTML file, you must choose what type of document to create. Sounds obvious, doesn't it? You're working on an XHTML file, right? Actually, though, several different versions of XHTML are available to you. You might compare them to dialects of a single language or, more casually, different "flavors." With HTML, the dialects or versions evolved around the proprietary features provided by specific browsers. With XHTML, the version depends on the document type definition, which I'll discuss subsequently.

The Document Type

One of the most cryptic XHTML components you'll run into is the `DOCTYPE` declaration. It identifies your file as an XHTML document and also identifies the specific document type definition, or *DTD*, that your document utilizes. The DTD is a file that defines the syntax of a language based on SGML. SGML is the

Standard Generalized Markup Language, a method created in the late 1960s at IBM for structuring documents. It continued to be developed for several decades into its current state. Although SGML is very powerful, it is also cumbersome and confusing for the average user. When HTML was developed, it was based on the attractive properties of SGML and is even referred to as a subset of SGML. Because XHTML 1 is a reformulation of HTML in XML syntax, it too makes use of DTDs.

Alphabet-Soup Organizations

Okay, so just what are the Internet Engineering Task Force and the World Wide Web Consortium, and how do they impact what you're doing?

The IETF is an open community of network engineers, operators, vendors, and researchers who are concerned with the smooth operation of the Internet and the development of Net-related architecture. The organization is open to any interested individual.

The IETF compiled the DTDs for HTML through version 3, at which time control over that activity was assumed by the W3C. IETF currently produces recommendations, specifications, and informational documents dealing with a wide variety of Internet protocols.

The W3C is an international industry consortium, jointly hosted by the Massachusetts Institute of Technology Laboratory for Computer Science (MIT/LCS) here in the United States, Institut National de Recherche en Informatique et Automatique (INRIA) in Europe, and the Keio University Shonan Fujisawa Campus in Asia. The consortium was formed in 1994 to develop common protocols for use on the Web. Tim Berners-Lee, creator of the World Wide Web, serves as director, and Jean-François Abramatic is the current chairman.

Commercial members fund the W3C. Although those members may be key industry players, like Netscape and Microsoft Corporation, the organization strives to remain "vendor neutral" while working to produce the specifications that will be recommended for inclusion in Internet-related products.

The most commonly known recommendations delivered by the W3C are the specifications for each new version of HTML, and now XHTML and XML. Each version begins as a public draft, and after a period of comment and refinement, it is published in final form. With the rapid expansion of the Web and the extremely short development cycles for new browser products, developers will often incorporate draft versions of HTML, XHTML, CSS, and other technologies long before they become a final standard.
Both the IETF and W3C are involved in far more than what I've mentioned here. For further information on their activities, you can visit their Web sites at `http://www.ietf.org` and `http://www.w3.org`, respectively.

An XHTML DTD is the entire collection of elements and attributes that may be used in that version of XHTML. The DTD also includes all the possible variations on elements that are allowed under its particular structure, which is known as an element's *content model*. You might think of a DTD as the definition of XHTML grammar.
The Internet Engineering Task Force, in cooperation with the W3 Consortium, created the DTDs for HTML 2 and HTML 3. Additional definitions have been created for specific browsers in an attempt to reproduce the behavior of the individual programs. The definitions can be handy when you know that your Web audience will be using a specific browser publisher (such as Netscape) or a specific browser version. Such is most commonly the case in *intranet* settings—internal corporate Web sites that aren't accessible to anyone outside the company. The system administrators would have greater control over what browser programs are used in an intranet. Although some of the new DTDs have not been developed or endorsed by the browser publishers, they will serve the purposes of most XHTML developers in intranet situations with few or no difficulties.
The `DOCTYPE` declaration tells a *parser*, or processor, what DTD it should use while processing the document. XHTML documents go through a browser parser before display, and (aside from being required) the `DOCTYPE` declaration is instrumental in allowing you to quickly review your XHTML syntax for errors, a technique that will be covered later in Chapter 11, "Validating Your Work."

The *DOCTYPE* Declaration

The `DOCTYPE` declaration always appears in the first line of your HTML document. It takes the following general form:

<!DOCTYPE html PUBLIC "*public identifier*"

"*sysid*">

The `public identifier` is the assigned name for the document type declaration (DTD). Some common public identifiers are listed in Table 3.1.

Table 3.1: Common DTD Public Identifiers

IDENTIFIER	USED FOR
`-//W3C//DTD XHTML 1.0 Strict//EN`	The version of XHTML that's a transformation of the former HTML 4.01
`-//W3C//DTD XHTML 1.0 Transitional//EN`	The XHTML version of HTML 4.01 Transitional
`-//W3C//DTD XHTML 1.0 Frameset//EN`	The XHTML version of HTML 4.01 Frameset

Tip A very broad catalog of DTDs is available at http://validator.w3.org/sgml-lib/catalog. The list was originally compiled for use with the Kinder, Gentler HTML Validator developed by Gerald Oskoboiny. He went on to become an employee of the W3C, turning the Kinder, Gentler HTML Validator into the W3C Validation Service.

In most of my XHTML examples, I'll be using the DTD for XHTML 1 Transitional. The construction of the `DOCTYPE` declaration will therefore appear as:

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

Warning Public identifiers are case sensitive. When inserting them in your documents, be sure to keep capitalization, spelling, and spacing exactly as shown. Users of HTML DOCTYPE declarations should especially note that the root element html in XHTML DOCTYPE declarations is presented in lowercase, not uppercase as occurred in HTML 4/4.01. Those familiar with HTML DOCTYPE declarations should also note that in the previous declaration, the sysid http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd (the URL pointing to the DTD at the W3C Web site) is no longer optional.

Seems like an awful lot of information for the first component required in your file, doesn't it? Don't worry, that's the worst of it for this section.

The *html* Element

Each XHTML document must be defined as such. After your `DOCTYPE` declaration, the second line of each document will be your *root element*, the outermost container-type element, which in this case is the `html` element. The `html` element has several new components in XHTML: the *namespace* and the *language identifiers*.

The namespace is a unique representation of something that's actually an abstraction. It's both a very simple concept and an incredibly complicated and controversial one. For now, what you need to know is that a namespace is an abstract collection of *names*, which for purposes here are the elements defined in XHTML. Namespaces, though an abstract collection, need a unique identifier. The best unique identifier the W3C could agree on was a *URI*, or Uniform Resource Identifier (what most people refer to as a URL). The namespace for XHTML has been defined as `http://www.w3.org/1999/xhtml`.

Tip Much of the confusion over namespaces has to do with the expectation of being able to follow the URI used to identify the abstract collection and actually find something. The Namespaces recommendation specifically states that a definition of the namespace is not to be found at the given URI. Therefore, it's best to think of the namespace as a label and not a Web address.

The namespace is included in the `html` element as the value of the `xmlns` attribute:

```
\<html xmlns="http://www.w3.org/1999/xhtml">
```

Finally, two attributes for language identification need to appear in the `html` element. An XML document would require just a single attribute, `xml:lang`. With XHTML still bridging HTML and XML, an attribute that today's HTML-based browsers will understand is also needed, and that's the `lang` attribute. The value of these attributes is a two-letter code identifying the language in which the document was written. For purposes here, that will always be `"en"` for U.S. English. Put all together, the `html` element now looks like this:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Tip You might have noticed that the XML version of the language attribute has xml: appended to it. This practice is known as namespace prefixing. The lang attribute used in XML is different than the lang element used in HTML and needs to be identified as unique when used in the same document with the html element. The prefix differentiates them. Any formal XML-related attributes are prefaced with xml, as in xml:lang.

Here, I've gotten the document started with the opening half of the `html`. At the very end of each XHTML document, the element is closed with the `</html>` tag.

Tip A closing tag provides a container of sorts. That is, the closing tag causes everything between it and the opening tags to be treated in a specific manner. The opening tag, the closing tag, and everything in between compose the full element. Though many people use the terms interchangeably, the tag is only the portion contained within angle brackets, such as <html>; the element includes the tags and the content between them.

The *head* Element

The `head` element contains information about your XHTML document. Think of it acting like letterhead. It names the document, provides information about its contents, names the author, and perhaps contains other identifying information. Like the `html` element, it has a closing tag to demarcate a specific section within your document. Most XHTML authors place this tag on the third line of the document, following the `DOCTYPE` declaration and `html` tag. It looks like so:

<head>

…descriptive information…

</head>

The *title* Element

Inside the `head` element, you'll find a `title` element. The element is a container for the text describing your document, so a closing tag is required. The text indicated by the `title` element will be displayed in the title bar of your Web browser, as shown in Figure 3.4.

Figure 3.4: The title bar at `http://www.sybex.com`

The text used should be short but informative. The title seen in Figure 3.4 is self-explanatory. Your title can include accented characters or special characters, but it cannot contain additional tags. The markup in the XHTML document would look like this:

```
<title>THE SYBEX HOME PAGE</title>
```

Many businesses like to keep the company name in the title of all pages with a brief description of what that particular page contains, such as:

```
<title>Widgets, Inc: Product Catalog</title>
```

Your choice of title text does have implications when the site is bookmarked or indexed by a search engine, as I'll discuss in Chapter 10—so choose your titles carefully!

The *meta* Element

The `meta` element is used to describe the properties of a document and assign values to those properties. The `meta` elements are not typically displayed to people viewing your page, but search engines and other programs designed to utilize them can access the elements. The `meta` elements use defining pieces of information, called *attributes*.

Tip The XHTML specifications, like the previous HTML 4 specifications, don't provide a standard set of meta properties. You can therefore be creative, but your usage should be standardized at least across the Web site, if not across all of your work. The properties I'll describe are those in common use on Web sites today.

Understanding Attributes

I'll look at how the `meta` element is constructed, as it is more complex than the ones I've looked at so far. In the example that follows, `name` and `content` are attributes of the `meta` tag. The attributes, in this case, are the names of the properties I'll be defining for the document. The information defined inside the content quotation marks is the value for those properties. The equal sign makes the name of the property synonymous with its value. Here's the code:

```
<meta name="Author" content="Ann Navarro" />
```

So, I now have a property of `Author`, which has a value of `Ann Navarro`. The `Author` property can be especially useful when more than one individual is working on a given Web-site project, because the author responsible for each page is clearly identified.
Other frequently seen document-identification `meta` elements include a copyright statement and a date. Both of these have some unique characteristics, as shown:

```
<meta name="copyright" content="&copy; 2000 WebGeek, Inc." />
```

The construction of the element is the same as the previous `Author` property. The name of this property is `copyright`, and the value is the text within the content attribute. But what does `&copy;` mean?

XHTML and "Empty" Elements

The `meta` element looks a little different than the elements you've encountered so far, in that it has a/character at the end. In XHTML, the concept of an *empty* element, as was often found in HTML, doesn't exist. To be compatible with XML's requirements, HTML's previously empty elements had to have either a closing tag added to them or a shorthand created that allowed the element to be open and closed within the same tag. That's what you're seeing here with the `meta` element. I could code it without the extra space, as:

```
<meta name="Author" content="Ann Navarro"/>
```

The format is fully acceptable in both XML and XHTML. However, the slash at the end of an element can confuse pre-XHTML browsers. Therefore, for compatibility's sake, you should place a space between the final content of the element and the slash, like so:

```
<meta name="Author" content="Ann Navarro" />
```

Such coding techniques will be discussed later in the XHTML-to-HTML compatibility guidelines in Part III and can also be found online in Appendix C of the XHTML 1 Recommendation at `http://www.w3.org/TR/xhtml1/`.

The code `&copy;` is a *character entity*, a description of a character in a format that the browser will understand and interpret for display. In this case, `&copy;` is the character entity for the copyright symbol (©). For more about character entities, see Chapter 4.

The next sample, the Date property, uses long-standing Internet traditions for date presentation:

```
<meta name="date" content="23 Oct 2000 18:54:32 GMT" />
```

The date is presented in what Americans generally think of as the international format of *date-month-year*. The month is entered with the standard three-letter abbreviation, without any punctuation. The year should be the full four numerals. The `18:54:32 GMT` value represents the current time in 24-hour *hour:minutes:seconds* format, at Greenwich Mean Time.
Even if you're not familiar with working in 24-hour time formats, the calculation is pretty simple. Before noon, all times are as you would write them if you were using A.M. notation. After the noon hour, simply add 12 to the P.M.-formatted time. To translate the example time back into familiar notation, subtract 12 from the hour notation of 18, and you get 6. The time noted is 6:54:32 P.M. Greenwich Mean Time.

Tip Keeping track of how many hours variance are between your local time and GMT can be confounding. GMT is eight hours ahead of Pacific Standard Time, or seven hours ahead of Pacific Daylight Time. GMT is five hours ahead of Eastern Standard

Time and four hours ahead of Eastern Daylight Time. You may see references to Zulu or UCT instead of Greenwich Mean Time. Both refer to the same time, the current hour in the time zone for Greenwich, England. GMT is traditionally the world's "reference time," a time notation that people will be able to calculate worldwide. UCT, or Universal Coordinated Time, is a "modernization" of the name for that time zone. Zulu is the long-standing military term for GMT.

Meta *Elements and Search Engines*

Probably the most common usage of `meta` elements is for indexing by search engines. Several of the major Internet search engines will first look for `meta` information and make use of it in their listings before using their own preprogrammed method of collecting information from your pages. A solid understanding of what they're looking for can help your site find advantageous placement within search results. To learn how the most popular search engines work, see Chapter 10.

The two most common indexing `meta` properties are `description` and `keywords`. The value of `description` is a short statement about your Web page:

```
<meta name="description" content="Our first basic HTML document." />
```

Many search engines will limit your description to 20–25 words or about 200 characters. In order to prevent your description from cutting off in midsentence or even worse, in midword, you should limit your value, too.

The `keywords` property is a list of words or short phrases—separated by commas—that you want to be used to make reference to your Web page from an index. In the sample XHTML document, you might want to have the page indexed as a resource and example for beginning Web developers. You could choose a `meta` element that looks like this:

```
<meta name="keywords" content="XHTML, HTML, basic, beginner, Web

developer, Web design, tutorial" />
```

As with the `description`, some search engines will limit how many keywords they will index for any given page. My recommendation is to pick no more than 20 and to list them in your perception of their order of importance or relevance to your topic.

It's All About Readability

Like the written page, an XHTML document is easier for humans to read if it's organized into neatly defined sections of text. No hard and fast rules exist about how many elements or how much information may be on each line within your XHTML document. However, industry experience shows that the fewer tags per line, the easier it will be for you or anyone else to go back and review later. For example, you may string your `head` elements all together in a single line, such as:

```
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"

  lang="en"><head><title>My Page</title></head>
```

However, most people find an arrangement such as the following to be easier to decipher and find errors:

```
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>

  <title>My Page</title>

  </head>
```

The issue becomes much clearer as your XHTML documents become more complicated with the addition of tables, lists, frames, and combinations of many other tags and elements. Though a clean markup may seem unnecessarily fussy, it will pay off in hours saved later during modification.

Tip An unfortunate by-product of the growing Internet has been the attempt by some to "tip the scales" in their favor on search engines by loading a document with keywords. Loading is the entering of a keyword repetitively to score a higher relevancy rating within search results. For instance, a real-estate agency could overload the Keywords section with multiple instances of the word property *or* real estate in order to get a higher ranking with search engines that count the number of instances a keyword appears in a document. Major search-engine providers have countered loading in a variety of ways, the most punitive being to completely remove any reference to that site from their archives.

The body *of Work*

Now that you've set your `DOCTYPE` declaration, opened the `html` container, and filled in all the `head` elements, it's time to get down to the actual document.
First, look at all the elements used so far, arranged in the XHTML document shown in Listing 3.1 (and on the CD-ROM):

Listing 3.1: A Basic XHTML Document, *basic.html*

```
<!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" xml:lang="en" lang="en">

<head>

<title>A Basic XHTML Document</title>

<meta name="author" content="Ann Navarro" />

<meta name="copyright" content="&copy; 2000 WebGeek, Inc." />

<meta name="date" content="23 Oct 2000 18:54:32 GMT" />

<meta name="description" content="Our first basic HTML document." />

<meta name="keywords" content="XHTML, HTML, basic, beginner, web

developer, web design, tutorial" />

</head>

</html>
```

The *body* Element

Like `head`, `body` is a container, so the closing tag is required. The `body` element holds all of the information from the end of the head container to the close of the `html` container. In other words, if it isn't in the `head`, it goes in the `body`.
As with the `meta` elements, `body` has a variety of possible attributes. The `body` attributes have more impact, though, in that they control how a number of circumstances happen or how items are displayed within the page. The most common attributes for `body` include settings for a solid color or graphic for a page background, and colors for text and hypertext links.

I'll add the following tag to the XHTML document and then dissect it attribute by attribute:

```
<body bgcolor="#FFFFCC" text="#000000" link="#0000FF"
alink="#FF00FF"
  vlink="#800080">
```

Backgrounds

Web developers have two options for defining what appears as a background for their Web pages: a solid color or a graphic. Both have their strengths and limitations. I'll take a closer look at the pros and cons of each.

Background Color

In the previous example markup, I've used the attribute `bgcolor` to assign a specific color to the background. In this case, I've chosen a light lemon-yellow color and have entered the *RGB value* of that color in hexadecimal notation (#FFFFCC). I've also used the hex notation for black text (#000000), blue links (#0000FF), red active links (#FF00FF), and purple visited links (#800080).

RGB is an abbreviation of Red-Green-Blue. Every color is made up of those three components, and a value is assigned to the amount of red, green, and blue that are blended to create the intended color. Values can range from 0 (none) to 255 (full color saturation).

Hexadecimal code, the base-16 numbering system, is used to store those values to keep the RGB value a three-byte value. A numeric value, converted into the binary code that computers understand, can hold a number up to 255 to remain in a single byte of memory space. Perhaps through a cosmic coincidence, hexadecimal notation tops out at FF—the representation of the number 255—before moving into three characters. So, the full range of 0 to 255 can be arranged in three bytes by using only two characters per color component.
To further translate the choice of lemon yellow, the color breaks down to a red component of 255 (FF in hexadecimal), a green component of 255, and a blue component of 204 (CC in hex). The value is preceded by the *hash symbol*—or # sign—to indicate that hex notation is being used. Table 3.2 shows 16 widely known color names and their corresponding RGB values in hex notation. For an in-depth look at color, RGB values, and hexadecimal notation, consult Chapter 16.

Table 3.2: 16 Common Color Values

COLOR NAME	HEXADECIMAL VALUE	COLOR NAME	HEXADECIMAL VALUE
Black	#000000	Green	#008000
Silver	#C0C0C0	Lime	#00FF00
Gray	#808080	Olive	#808000
White	#FFFFFF	Yellow	#FFFF00
Maroon	#800000	Navy	#000080
Red	#FF0000	Blue	#0000FF
Purple	#800080	Teal	#008080
Fuchsia	#FF00FF	Aqua	#00FFFF

Warning Though myriad color names have been assigned to various hex color representations, not all browsers understand much more than the basic 16 listed in Table 3.2. If you want to be sure of the color you've chosen for your visitor, use the hex notation instead of the color name.

Background Graphics

The other option for background treatment would be to choose a graphic as the background for all other text and images to come. Background graphics have a unique property applied to them known as *tiling*. Tiling is the endless repetition of an individual graphic. The browser will automatically tile background graphics if the Web page is larger in physical dimensions than the graphic. In most cases, tiling is very beneficial in that the size of the graphic file is smaller and the page loads faster than if you were to apply a single graphic. On Web pages, the image is placed in the top-left corner of the document and then repeated in a matrix to the right and down as many times as is necessary to appear behind all portions of the document. The process of choosing background graphics and a more in-depth discussion of graphic file sizes can be found in Chapter 17.

Figure 3.5 illustrates what I'll add to the `basic.html` document. It is a very wide graphic at 1,400 pixels but very short at only 5 pixels. The image contains a red bar on the left side with some shading to provide the illusion of a depth change between the red and the white portions of the graphic.

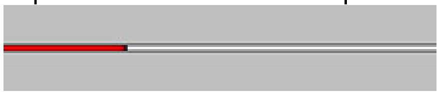


Figure 3.5: The background graphic `bar.gif`

I'll insert the image into the XHTML document with the following markup:

<body background="bar.gif" TEXT="#000000">

Tip The use of background graphics, especially ones with a color running down the left as shown in Figure 3.5, are becoming passé in Web-design circles. Much more can be done with traditional graphics and the creative use of color than with a background image that may run into trouble on some of the higher screen resolutions used today.

When using a graphic, the attribute used changes from `bgcolor`, used for solid-color rendering, to `background`. The value for the attribute is the name of the graphic file being used in *URL* format. The Uniform Resource Locator—sometimes referred to as a URI or Uniform Resource Identifier—is what we typically refer to as a Web-site address.

I'll discuss the topic of URLs in more detail in the upcoming section on links. For now, it will suffice to say that you can use just the filename for graphics that reside in the same directory on the Web server as your XHTML document.

When the new `body` tag is inserted into the `basic.html` document and the background graphic tiles, the document looks like Figure 3.6.

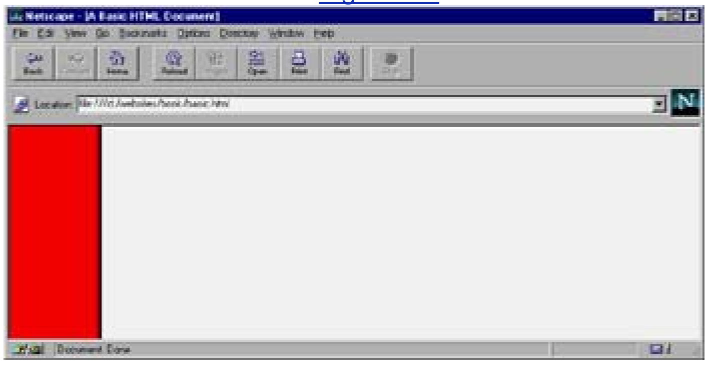

Netscape - [A Basic HTML Document]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location:
Document: Done

Figure 3.6: The `basic.html` document with a tiling background graphic

So why hasn't the graphic tiled to the right as I explained previously? In order to keep the intended appearance of a colored bar down only the left side of the page, I created the bar graphic at 1,400 pixels wide. That width will accommodate almost every screen size and browser window width without running out of horizontal space. If the graphic doesn't need to tile to the right because of its width, it will only tile downward.

Tip You may recall from Chapter 1 that the most common screen resolutions—the number of pixels across and from top to bottom on a monitor—are 640 pixels wide by 480 pixels tall and 800 × 600; some are even at 1024 × 768. What matters here is not monitor size but the resolution chosen. At 640 × 480, everything appears larger. If you set the resolution to 1024 × 768, you will increase your available space by fitting more (and smaller) pixels into the space available on your monitor. Several higher resolutions are available but are generally only used on screens at least 19 or 21 inches in size.

To recap, most of today's Web designs use solid background colors with the `bgcolor` attribute (or its corresponding style-sheet property, as you'll learn in Chapter 7). Background images depend significantly on the anticipated screen resolution and size of the viewer's monitor. Of course, finally, taste in design changes quickly, especially on the Internet. As such, background images are becoming passé and are often judged as the work of a novice designer. With background colors so much easier to use, why make more work than necessary?

The *text* Attribute

The `text` attribute defines the default color of your page text; I've chosen black to contrast well with the lemon-yellow background. The color is stated in hexadecimal notation in the markup (TEXT="#000000"), as are the colors in all other color-defining attributes.

Warning You may occasionally see specific color names in place of the hexadecimal notation that I've stated is required. In some later versions of some browsers—primarily Netscape Navigator and Microsoft Internet Explorer—the browser will understand the 16 common color names shown in Table 3.2. However, the practice did not gain widespread support, and it may not remain an option. By using hexadecimal notation, you can feel sure that your color choices will be rendered properly, regardless of new browser versions and future updates to the XHTML specifications. For example, the use of blue as a color name may cause a browser to interpret the color as B, LU, and E, which ends up as #0B000E, a near-black color.

Links

The remaining attributes of `link`, `alink`, and `vlink` all deal with hypertext links on a Web page in all possible conditions: visited, unvisited, and active. An active link is that brief moment during a mouse click.

The `link` attribute defines the color of hypertext links that have not previously been activated by the viewer. In the example markup, I've chosen blue (`link="#0000FF"`), which has been the defacto standard since the first color-enabled browsers came into use.

The attribute `alink` represents the color of the link while it is being activated. When you add a link to the basic XHTML example document, you'll be able to click it to see this property in action. While your mouse button is down, the link changes color. My choice for this attribute was the color red (`alink="#FF00FF"`).

The `vlink` attribute sets the color of hypertext links that your visitor has been to recently. Because Web sites aren't linear by design, it helps people remember where they've been and what information they may still want to view. I chose purple in the markup (`vlink="#800080"`), which is also the traditional color for visited links.

Tip Before the HTML 3.2 specification, which allowed your markup to specify text and link color choices, link colors were determined by the visitor's browser settings. By tradition, those settings defaulted to blue for an unvisited link and maroon or purple for visited links. The defaults gave the viewer an instant visual clue as to what areas of a Web site had previously been visited. Now that the document author can set these colors, care should be taken not to venture too far from these traditional expectations. Should you wish to use different colors, I recommend choosing hues that contrast from the regular text and from each other, like the traditional blue and purple do.

Adding Headings

You're finally ready to add some text! XHTML documents do carry over some conventions from the printed word, like headlines in newspaper and even the section headings in this book. The heading element is used to briefly describe the topic of the section it precedes. You can use six levels of headings, from level 1, the highest level, through level 6, the lowest.

The element is a container and has a format as follows:

<h1>*Your Heading Here*</h1>

The *align* Attribute

A common attribute for the heading element is `align`. It tells the browser how to align, relative to the page, the text contained in the heading. Values for `align` may be `left`, `center`, or `right`. In the absence of an `align` attribute (or a style-sheet instruction, which I'll cover later in Chapter 7), the element is aligned to the left. Center is the most common alignment for headings and is created as follows:

<h1 align="center">*Your Heading Here*</h1>

Adding Text

Are you ready for some typing? It's now time to actually write something in your document. XHTML provides you with two basic elements that control text placement: a paragraph element and a line break.

The Paragraph Element

Each new paragraph begins with the `p` tag. The tag normally produces a single white-space line before and after the paragraph text. The default alignment for a paragraph is flush left with a ragged-right margin. The alignment may be set with the `align` attribute, using a value of `left` (implied), `center`, or `right`.

Tip None of the alignment options provide for page-justified text. XHTML currently does not allow for such treatment, as the specific boundaries of a given page aren't possible for the author to define in XHTML alone. Monitor resolution and browser window size determine the dimensions of the viewable area. Style sheets, which I'll cover in Chapter 7, can do a bit more with margins and other box properties within the document.

XHTML and Quotable Attributes

If viewing the source HTML code of Web pages has become a habit of yours, as it has for many new Web page authors, you undoubtedly have noticed quite a few discrepancies when it comes to handling attribute values. Some are enclosed in quotes while others aren't. An obvious pattern that might help you deduce a set of rules may not even be evident.

In HTML, some rules governed whether an attribute value did or didn't have to be quoted. In order to conform to the well-formedness requirements—that is, conformity guidelines—of XML, all XHTML attribute values *must* be quoted. It is no longer optional in some cases.

The Line Break

You may occasionally wish to begin a new line without beginning a new paragraph, which would insert a blank line. HTML provides for this with the `br` element, which may be placed within the paragraph container without triggering an implied close of the paragraph.

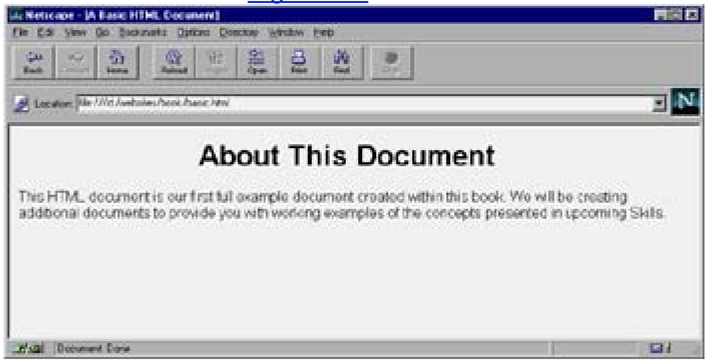
After the addition of paragraph text, the basic XHTML document now looks like Listing 3.2.

Listing 3.2: The Updated *basic.html* document, *basic2.html*

```
<!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"

  xml:lang="en" lang="en">

<head>

<title>A Basic XHTML Document</title>

<meta name="author" content="Ann Navarro" />

<meta name="copyright" content="&copy; 2000 WebGeek, Inc." />

<meta name="date" content="23 Oct 2000 18:54:32 GMT" />

<meta name="description" content="Our first basic HTML document." />

<meta name="keywords" content="XHTML, HTML, basic, beginner,

  web developer, web design, tutorial" />

</head>

<body> bgcolor="FFFFCC" text="#000000" link="#0000FF"

alink="#FF00FF"
```

```
  vlink="#800080">

<h1 align="center">About This Document</h1>

<p>This XHTML document is our first full example document created

within

this book. We will be creating additional documents to provide you

with working

   examples of the concepts presented in upcoming chapters.</p>

</body>

</html>
```

The document now has something other than background to display! The `basic2.html` file is shown in a Web browser in Figure 3.7.


Netscape - [A Basic HTML Document]
File Edit View Go Bookmarks Options Directory Window Help
Location:
Document: Done

About This Document

This HTML document is our first full example document created within this book. We will be creating additional documents to provide you with working examples of the concepts presented in upcoming Skills.

Figure 3.7: The completed `basic.html` file as seen in a Web browser

Summary

It is important to choose a text editor that you'll be comfortable working with. A whole army of different editors is available; editors have widely varying features, formats, and prices.
The XHTML document has a defined structure that must contain several key elements. The `DOCTYPE` declaration states which XHTML grammar is used. XHTML elements may be composed of opening and closing tags that act as containers, or they may be empty tags that use a unique closure method. The `html` container holds everything else in the document besides what is contained in `DOCTYPE`. The `head` container sets apart the descriptive information about the document: the `title` and `meta` elements. The title should accurately describe the content of your page. The page author, page description, keywords, and other information are stored in the `meta` elements.
The `body` tag contains attributes that set the look of your page. The attributes describe the background color or graphic, what color the text font is displayed in, and the colors of your hypertext links in various states. You also now know how to display basic text within your document using headings, paragraphs, and line breaks.

Your work so far may feel highly regimented and even a little intimidating, but you've now gotten the technicalities out of the way and can concentrate on your page design with the proper structure in place.

Chapter 4: Working with Text and Images in XHTML

Overview

Now that you've learned how to create a basic XHTML Web page, it's time to explore some of XHTML's nifty tricks! This chapter will get you started. First, you'll learn to connect your Web pages to each other and to link to pages halfway around the world. Next, you'll learn about XHTML's wide variety of text-placement and styling tools. You'll create numbered lists, lettered lists, bulleted lists, and definition lists, as well as quotations, citations, and preformatted text areas. Finally, you'll learn how the World Wide Web is truly worldwide by using international character sets and special text characters that can be created even if you don't have the keys for them on your keyboard.

This chapter covers the following topics:

- Adding links to your Web page
- Lists, lists, and more lists!
- Quoting other material
- Modifying text display with styling
- Using special character sets
- Placing images within the page
- Setting the flow of text around images
- Hints for optimal image behavior
- Bordering images with white space

Linking Out

What makes the World Wide Web such an exciting place—and what makes it quite literally a web of information—is the *hyperlink*. With the hyperlink, you can jump from any Web page to another page on the same site or to a page in a site located on the other side of the globe, all with a single click of a mouse.

Anchoring the Link

Links are created using a specific form of the `anchor` element, with its opening tag `<a>`. The tag surrounds the text or object that is to be linked, so the link is *anchored* to that text or object.

You can link to a file two ways. The link can refer to the entire file, in which case a Web browser will display it from the top. Or you can link to a specific point within a file, in which case the entire file will still be loaded, but the browser will display from the point specified.

Linking to a File

To illustrate these two linking behaviors and their syntax, I'll construct a new XHTML file. The beginning of the file looks like Listing 4.1.

Listing 4.1: links.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Example of Links</title>

<body bgcolor="#FFFFFF" text="#000000" link="#0000FF"

alink="#FF00FF"

  vlink="#800080">



<p>This page provides examples of two different types of links. The

first

  is a link out to another page. This link is to the

  <a href="http://www.sybex.com">Sybex Web site</a>.</p>
```

```
<p>The second link is to another file on this imaginary Web site.

This link

  is to

<a href="page2.html">Page Two</a> of our site.</p>



</body>

</html>
```

Warning For the purpose of conserving space in the listings, only the immediately relevant portions of the file will be shown in this chapter. So, in Listing 4.1, the meta elements are omitted. As discussed in Chapter 3, your working documents should always include them.

The only item in the file unfamiliar to you is the anchor tag. As you can see from the example, the anchor tag is a container tag and has a corresponding closing tag.

The `href` attribute represents the *hyperlinked resource* being linked to. That's a fancy way of saying that the value of the `href` attribute is the URL of the linked file or site. As with all other attribute values, the URL must be contained within quotes.

The URL can be written in *absolute* form or in *relative* form. Absolute URLs can be thought of as the "entire" URL, including the protocol, the domain, the path, and the filename. In Listing 4.1, the link to the Sybex Web site uses the URL `http://www.sybex.com`, which is written in absolute form.

Relative addressing lists the path and filename within the URL *relative* to the location of the file that contains the tag. In its simplest form, only the filename is declared, as in the second link in Listing 4.1, which only uses the filename `page2.html`. In the relative form, it is understood that the file *resides in the same directory* as the file that contains the tag. The system will know where to look for the page. So the second link in Listing 4.1 is short for the following line of code:

```
<a href="http://www.yourcompany.com/page2.html">
```

In both the absolute and relative forms of addressing in Listing 4.1, a browser will display the new file from the top.

It's All Relative: Understanding Relative Hyperlinks

Relative addressing really comes in handy when you're dealing with multiple file directories and the logical structure that most Web sites use.

For example, consider that Acme Widgets has a Web site at `http://www.widgets.com`. Their widget catalog is online at `http://www.widgets.com/catalog/`. They also have a collection of FAQ (Frequently Asked Questions) files located at `http://www.widgets.com/faqs/`. When the catalog page is developed, the Web designer can use a relative URL to link to the FAQ about each product without having to key in the full path of `http://www.widgets.com/faqs/` every time.

To use relative links, you must determine where your Web page is in relation to the image file. In the case of Acme Widgets, the catalog directory is at the same "level" as the FAQ directory—one level down from the root level.

The Web server understands a bit of basic shorthand that says, "Go back up one level": ../ (that's two periods followed by a forward slash).

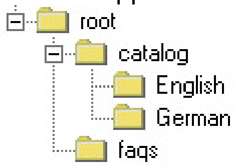
To use the FAQ file named `faq1.html`, the anchor tag would be written as:

```
  <a href="../faqs/faq1.html">
```

The code tells the server, "Go back up one directory level, then go down into the FAQ directory and display the file `faq1.html`." For Web-design purposes, it is the same as writing the following:

```
<a href="http://www.widgets.com/faqs/faq1.html">
```

The shorthand can be used several times in succession. Imagine that Acme Widgets has two versions of its catalog, one written in English and the other in German. Its Web site has two appropriately named directories below the catalog directory.
You need to code links from within the English subdirectory to FAQs in the FAQ directory. How to do it? Again, compare where your current page is *relative* to the entire directory structure. The FAQ directory is on the first level below the root. The English directory is on the *second* level below the root because it is a subdirectory of catalog, which is at the same level as the FAQ directory. Confused? Visualize how it would appear in a familiar directory tree on your hard drive, as shown here:



As with all relative addressing, you need to specify the level *above* the destination, in order to travel *down* the path into the desired directory. Because the root directory (the level above the FAQ directory) is two levels above the English directory, use the previous-directory notation twice, as shown:

```
<a href="../../faqs/faq1.html">
```

The code tells the server, "Go back into the catalog directory, back again into the root directory, then go down into `faqs`, and display the `faq1.html` file."

It may look more complicated than it really is. If you ever find yourself stumped over how many previous-directory notations to use, draw yourself a small chart!

Linking to a Point in a File

The second type of link directs the browser to deliver the user to a specific point in the file being loaded. For that to work, the Web developer needs to give the browser an electronic roadmap of sorts. To link to point A in the new file, some sort of marker must identify where point A is. That's where the anchor attribute `name` comes in. Take a look at Listing 4.2. The file `page2.html` was linked in the `links.html` file in Listing 4.1.

Listing 4.2: page2.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Example of Links - Page 2 version 2</title>

<body bgcolor="#FFFFFF" text="#000000" link="#0000FF"

alink="#FF00FF"

  vlink="#800080">



<h1>Page Two</h1>

<p>You arrived at this page from a hyperlink on the links.html

page.</p>
```

```
<p>Aside from linking from one page to another, you can link from

one

  section of a page to another, especially useful in long pages.

  Let's link to <a href="page3.html">where we'll try it out.</p>

</body>

</html>
```

Now I'll create a long page, one that will extend beyond the lower boundary of your browser window—if you can see the entire file, resize your browser so that you can't see the second heading. To fill the page, I've used a traditional passage of Latin text, as shown in Listing 4.3.

Listing 4.3: page3.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Example of Links</title>

</head>

<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" alink="#FF00FF"

  vlink="#800080">

<h1>Page 3</h1>



<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae

ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor

sit amet sed tibi nunc dimittis ut donae

ferente. Multum in parvo sed non in adipisce elim van dutter.

Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed

tibi nunc dimittis ut donae ferente.

</p>



<p>Now we want to link <a href="#down">further down</a> the

page.</p>



<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae
```

```
ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor
sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter.
Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed
tibi nunc dimittis ut donae ferente.
</p>

<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor
sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter.
Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed
tibi nunc dimittis ut donae ferente.
</p>

<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor
sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter.
Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed
tibi nunc dimittis ut donae ferente.
</p>

<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor
sit amet sed tibi nuncdimittis ut donae
ferente. Multum in parvo sed non in adipisce elim van dutter.
Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed
tibi nunc dimittis ut donae ferente.
</p>
```

```
<h2><a name="down">Further Down The Page</a></h2>


<p>Lorem ipsem dolor sit amet sed tibi nunc dimittis ut donae

ferente. Multum in parvo sed non in adipisce elim van dutter. Dolor

sit amet sed tibi nunc dimittis ut donae

ferente. Multum in parvo sed non in adipisce elim van dutter.

Multum in parvo sed non in adipisce. Non angli sed angeli. Amet sed

tibi nunc dimittis ut donae ferente.

</p>

</body>

</html>
```

You'll notice the anchor tag around the phrase "further down" has a URL with a hash mark—the `#` symbol, commonly referred to in the United States as the "pound sign." The tag is shown here:

```
<a href="#down">further down</a>
```

The tag instructs the browser to go to a specifically marked section on the same page. Here, that point is the `h2` element, which contains the anchor element `<a name="down"></a>`. The code is shown here:

```
<h2><a name="down">Further Down The Page</a></h2>
```

Take note that the attribute isn't an `href` attribute but the `name` attribute. Also, the value of the `name` attribute is the *same* as the previous `href` attribute, minus the `#` symbol.

Tip The labels used here follow the same naming conventions as traditional URLs. Spaces are discouraged, so many developers find that using mixed-case letters in place of spaces—PointA *instead of* pointa—can increase readability if anchor names need to be longer than a single word.

Block Elements

A *block element* is an XHTML element that causes a paragraph break. That means that a blank line is generally inserted before and after the element content when it's presented to the user. In Chapter 3, you were introduced to the paragraph element and to the headings elements, `<hx>`. This section will cover additional block elements, including lists, preformatted text areas, and block quotes.

Creating Lists

In XHTML 1, you can implement three formal types of lists: the *unordered* list, the *ordered* list, and the *definition* list. Each type of list can contain other block-and text-level elements, including additional lists nested within them. The only block elements that specifically are not allowed within lists are headings and the address tag.

Unordered Lists

An unordered list is what word processors typically call a *bulleted* list. Each item in the list begins on a new line, preceded by a round, black dot. In XHTML, the browsers do have some leeway in how to present such a list. Most commonly, you'll see the list indented from the left margin and a round, black disc used to set off each list item.
All lists are containers, so the unordered list tag, `<ul>`, has a closing partner of `</ul>`. Each item in the list is set off by its own list-item element, `<li>`.

A basic unordered list is constructed as follows:

```
<ul>
```

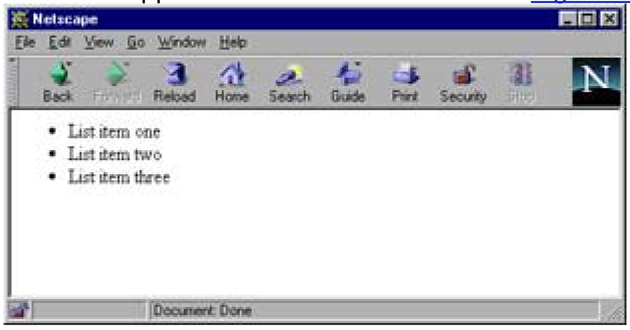
<li>List item one</li>

<li>List item two</li>

<li>List item three</li>

</ul>

The list appears in a browser as shown in Figure 4.1.


Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Document: Done

- List item one
- List item two
- List item three

Figure 4.1: An unordered list as seen in Navigator 4.72

The unordered list element has an attribute called `type` that changes how the bullet is presented. The available choices are shown in Table 4.1.

Table 4.1: Bullet Style for Unordered Lists

STYLE	DISPLAY
Disc (the default)	The familiar round, black disc
Square	A small, filled square
Circle	An unfilled circle

If, for example, the square option were chosen, the list tag syntax would look like this:

<ul type="square">

Ordered Lists

An ordered list shows a specific sequence, denoted by numbering or lettering. The basic syntax is identical to the unordered list. It is a container, so the opening tag, `<ol>` requires a closing tag, `</ol>`. List items are included with one or more `<li>` elements, as shown here:

<ol>

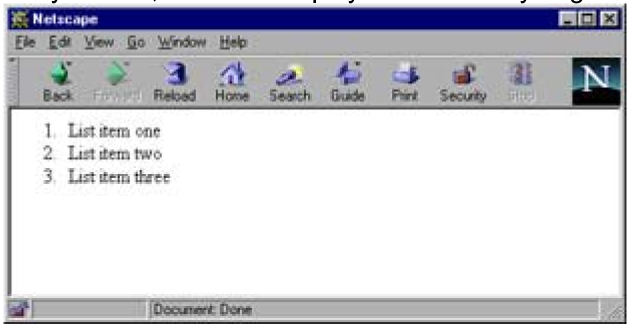
<li>List item one</li>

<li>List item two</li>

<li>List item three</li>

</ol>

By default, the list is displayed numerically beginning with the number 1, as shown in Figure 4.2.


Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Document: Done

1. List item one
2. List item two
3. List item three

Figure 4.2: An ordered list in Navigator 4.72

Ordered lists also make use of the `type` attribute. Instead of changing bullet style, as the attribute does in unordered lists, the `type` attribute changes the numbering or lettering system to be used, as shown in Table 4.2.

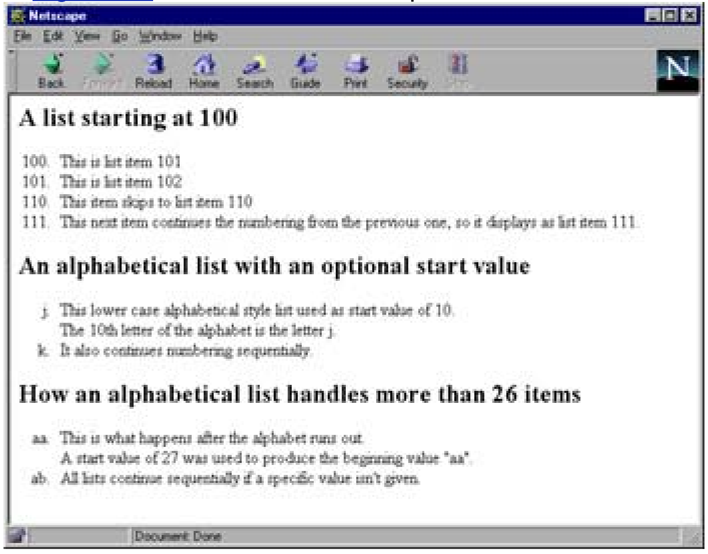
Table 4.2: Numbering Styles for Ordered Lists

TYPE	STYLE	DISPLAY
1	Arabic numbers	1, 2, 3
a	lowercase alphabet	a, b, c
A	uppercase alphabet	A, B, C
i	lowercase roman style	i, ii, iii
I	uppercase roman style	I, II, III

Additionally, you can set the starting point for any of these numbering styles using the `start` attribute or set the number value of a specific list item by using the `value` attribute. An example of these attributes is shown here:

```
<ol start="100">
<li>This is list item 101</li>
<li>This is list item 102</li>
<li value="110">This item skips to list item 110</li>
<li>This next item continues the numbering from the previous one,
so it
  displays as list item 111.</li>
</ol>
```

Figure 4.3 shows how the example list and two others look in a Web browser.


Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security
Document: Done

A list starting at 100

100. This is list item 101
101. This is list item 102
110. This item skips to list item 110
111. This next item continues the numbering from the previous one, so it displays as list item 111.

An alphabetical list with an optional start value

j. This lower case alphabetical style list used as start value of 10.
   The 10th letter of the alphabet is the letter j.
k. It also continues numbering sequentially.

How an alphabetical list handles more than 26 items

aa. This is what happens after the alphabet runs out.
    A start value of 27 was used to produce the beginning value "aa".
ab. All lists continue sequentially if a specific value isn't given.

Figure 4.3: Three examples of ordered lists using various numbering styles and `start` or `value` attributes

Experiment with different starting values and specific list-item values. You should start to see how well XHTML handles lists.

Definition Lists

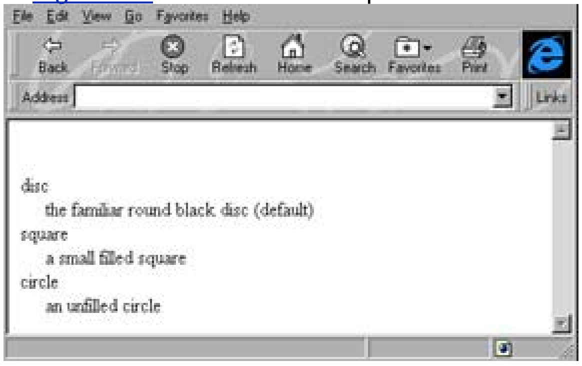
Definition lists are a little more complex than bulleted or numbered lists. They're used when terms are introduced with definitions, as in glossaries or terminology references.
Definition lists, like all other lists, are containers, so an opening `<dl>` is paired with the closing `</dl>`. Instead of having a single line item, `<li>`, for each item in the list, a definition list has both a term

element, `<dt>`, and a definition element, `<dd>`. At least one term and one definition must be included in the list.

If this book were an XHTML page, a definition list could have been used to display the three bullet-style options available within unordered lists. To do so, you'd use the following markup:

```
<dl>
<dt>disc</dt>
<dd>the familiar round black disc (default)</dd>
<dt>square</dt>
<dd>a small filled square</dd>
<dt>circle</dt>
<dd>an unfilled circle</dd>
</dl>
```

Figure 4.4 shows this markup rendered in a Web browser.


File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print
Address
Links

disc
    the familiar round black disc (default)
square
    a small filled square
circle
    an unfilled circle

Figure 4.4: A definition list as seen in Internet Explorer 5.0

XHTML has only two rules regarding what can or can't be included in these two elements. The term element `dt` can only contain text-level elements. The definition element `dd`, however, can contain both text-and block-level elements, which effectively allows paragraphs and other nested lists.

Handling Quotations

As its name indicates, a quotation element is used when you are quoting a block of text. Real-world applications of this are common. You might be converting a company employee manual into HTML for the intranet. The section on payroll practices may quote several pieces from your state's labor law. Your sales staff may want to include on your public Web site brief testimonials from satisfied customers. Such examples lend themselves to the use of `blockquote` and `quote` elements.

The *blockquote* Element

The `blockquote` element is a container, and it therefore requires a closing tag, `</blockquote>`. As a block-level element, it can contain other block-level elements, such as paragraphs, lists, and text-level formatting.

Most major browsers display the contents of the `blockquote` element by indenting from both left and right margins. Because `blockquote` is a block element, paragraph breaks are inserted both before and after the container contents.

Warning Though most browsers present blockquote with a double-sided indent, you cannot rely upon that for presentation, as the HTML and XHTML recommendations have never specified it. It is a tradition taken from the print world and programmed into the first browsers, and it has remained. A particularly creative browser programmer could instruct her program to present blockquote in a green font three times larger than the standard, and it would still be valid.

The Quote Element

For short, inline quotes, a separate element is available: `q`. Quote is also a container and has a corresponding closing tag of `</q>`. Quote is for quoting short passages, which can only contain other inline elements.

Controlling Preformatted Text

From the beginning, one of the most basic tenets of HTML, and now XHTML, has been flexibility in display. Text lines wrap based on the end user's screen resolution and browser window size rather than on premeasured page widths. Even with the advent of layout-control measures now available to the Web developer in style sheets (covered in Chapter 7), inherent flexibility should only occasionally be interrupted when the lack of absolute spacing will distort or destroy the informational content.
In order to preserve absolute text placement, XHTML allows a preformatted text area through the opening tag `<pre>` and the corresponding closing tag `</pre>`.
The data contained within the tags is displayed *exactly* as it is laid out in the source XHTML file. Line breaks and multiple-space characters (to produce white space) are preserved. In order to keep the absolute formatting of the data, browsers will use a fixed-width font. The default font used in both Internet Explorer and Navigator on a Windows platform is Courier New.

Modifying Text Display

For some time now, HTML has provided for basic text styling—such as italic, boldface, and underline. XHTML 1 carries that forward in the Transitional version. HTML, and now XHTML, has also allowed the Web developer to mark passages for emphasis and strong emphasis, letting the browser determine how to present that. Directing for emphasis without stipulating specific rendering instructions is known as providing the text's *structure* versus its display.
The argument about whether to use structural markup over display markup has become moot. The traditional means of rendering emphasis with the container `<em>` and `</em>` was with italic text. Strong emphasis, using the container `<strong>` and `</strong>`, generally produced bold text. Now, though, emphasis could be displayed using a different font color or by increasing the font size. Strong emphasis might be displayed by turning the text into purple italics, indented, and three font sizes larger than the original. Some browsers let the user determine such visual cues, while others are preprogrammed. The bottom line is that you can't count on an `em` element to produce italicized text. It probably will for 95% or more of your visitors, but it's not guaranteed.

In order to counteract display uncertainty, tags were developed that specifically produced *font styling*—rendering of the text with a visual enhancement such as italic, boldface, or underline, without changing the color, font size, or font face.

Tip Though this next section discusses how to be very specific in your presentation, XHTML is moving away from this habit in the markup and relegating it to style sheets. The structural form of the element is preferred, and is far more accessible in today's world of alternative Web-browsing devices (see *Chapter 20* for more information on alternative devices and accessibility for people with disabilities).

Font-Style Elements

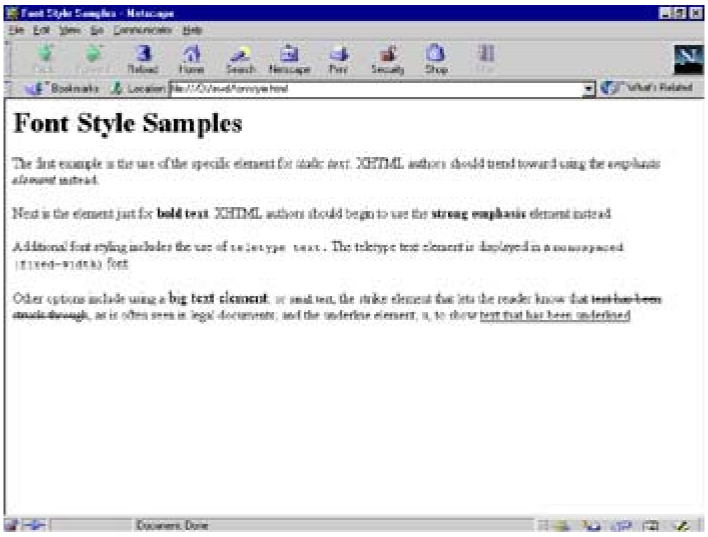
Font-style elements are text-level elements. That means they can be used inline without creating paragraph breaks around them. A single word or phrase could be rendered in *italic* or **bold** type just as easily as it was here. Each element is a container and requires a closing tag.
The XHTML markup in Listing 4.4 provides examples of the most commonly used font-style elements.

Listing 4.4: fontstyle.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Font Style Samples</title>

</head>

<body bgcolor="#FFFFFF" text="#000000" link="#0000FF"

alink="#FF00FF" vlink="#800080">
```

```
<h1>Font Style Samples</h1>


<p>The first example is the use of the specific element for
  <i>italic text</i>. XHTML authors should trend toward using the
  <em>emphasis element</em> instead.</p>


<p>Next is the element just for <b>bold text</b>. XHTML authors
should begin to use the <strong>strong emphasis</strong> element
instead.</p>


<p>Additional font styling includes the use of <tt>teletype
text.</tt>
The teletype text element is displayed in a <tt>monospaced
(fixed-width)</tt> font.</p>


<p>Other options include using a <big>big text element</big> or
  <small>small text</small>; the strike element that lets the
reader know that
  <strike>text has been struck through</strike>, as is often seen
in legal documents; and the underline element, u, to show
  <u>text that has been underlined</u>.</p>
</body>
</html>
```

Figure 4.5 shows `fontstyle.html` displayed in Netscape Navigator 4.73 on a Windows 98 machine.


Font Style Samples - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location What's Related
Document: Done

Font Style Samples

The first example is the use of the specific element for *italic text*. XHTML authors should tend toward using the *emphasis element* instead.

Next is the element just for **bold text**. XHTML authors should begin to use the **strong emphasis** element instead.

Additional font styling includes the use of `teletype text`. The teletype text element is displayed in a monospaced `(fixed-width)` font.

Other options include using a **big text element**, or small text, the strike element that lets the reader know that ~~text has been struck through~~, as is often seen in legal documents; and the underline element, u, to show text that has been underlined.

Figure 4.5: How Navigator 4.73 displays the file `fontstyle.html`

Deprecation versus Style Sheets

The distinction between deprecation and a suggestion by the W3C to use a style sheet is an important one. Deprecation is a strong signal that says, "Don't use this anymore. It's been done away with, or we've provided this other means of accomplishing the same thing." The W3C has deprecated many text-styling elements, including strikethrough and underline tags.
For those text-styling elements that have not been deprecated, the consortium has expressed its confidence in style sheets as a better means of managing those styling issues. Even so, you still must keep in mind the audience that will be visiting the Web site and whether they'll likely use a browser that supports not only style sheets but also the style properties you want to use. See Chapter 7 for more information on style-property support across today's most popular browsers.

Special Character Entities

As its name implies, the World Wide Web is a huge entity, consisting of information in many languages. I've already talked about how the Web is intended to convey information across many types of machines and operating systems. How then do you ensure that people who speak different languages, and whose languages use symbols that are not found on standard Western keyboards, can use the Web?

Character Sets

Character sets have been around for a long time, long before the Web. You might be familiar with term *ASCII*. ASCII, invented by IBM Corporation in the 1960s, is one set of characters and can be used fairly commonly. If you're using a standard English keyboard, you might want to think of ASCII as what you can type without holding down any special keys (except the Shift key for capital letters). If you type characters by holding down the Option, Command, Control, or Alt keys, the characters probably are not ASCII.
XHTML uses a particular *Document Character Set*, which simply means a set of certain characters that are usable in HTML documents. The Document Character Set for HTML is ISO10646, also known as the *Universal Character Set*, or UCS.

Tip ISO is the International Standards Organization. It is responsible for determining international standards for many types of things, not just for the computer industry. ISO sets all types of measurements and other industrial types of standards, such as those for the metal industry.

As in ASCII, certain characters are part of the UCS and others are not. Chances are, most characters you would be interested in using are in the UCS, but the situation is a bit more complicated than what it seems at first glance.

Browser Support of Character Sets

As you learned from the discussion of browser capabilities in Chapter 2, just because a feature exists does not mean a browser supports it. In fact, the default on most modern browsers is to support only a subset of the Universal Character Set—the ISO 1889-1 subset, also known as Latin-1. So your standard, Western version of Netscape Navigator, without any configuration other than defaults, will only correctly display the characters in this subset. However, browsers can recognize other subsets, or character encodings, of the UCS.

Why doesn't every browser automatically work with the entire Universal Character Set? The answer is pretty simple—the UCS is very large. Because most documents are written in a single language (although, of course, that language may vary), it is easiest to let the author of each document choose the appropriate subset.

Tip If you are familiar with character sets in general, you might be interested to know that the Universal Character Set is the same as Unicode2. Both are 16-bit character sets. If you set the upper byte of the UCS to zero, the bottom eight bits compose the ASCII character set.

Characters Not in the Latin-1 Subset

What if you want to use characters not in the Latin-1 subset? You have two choices. One option is to specify in the `meta` element which type of encoding you would like to use. The second option is to specify *character entities*, which are codes for the characters you wish to use. I'll take a look at both options.

Using the *meta* Element

Remember the `meta` element? In case you've forgotten, it goes in the document's `head` area and allows the author of the XHTML document to include extra information about the document. One of these extra pieces of information can be the document encoding (a.k.a. character set, a.k.a. character encoding), which you set by assigning a code.

Here's an example:

```
<meta http-equiv="Content-Type" Content="text/html, charset=fr-FR">
```

The document encoding here is French for France (as opposed to French for Canada or French for Senegal). Just to give you an idea of some of the encoding options available, Table 4.3 lists some of the more frequently spoken languages around the world.

Table 4.3: Common Character Encodings

LANGUAGE	CHARACTER SET
English	En
English/United States	en-US
English/United Kingdom	en-GB
French	Fr
French/France	fr-FR
French/Canada	fr-CA
German	De
Japanese	Ja
Chinese	Zh
Chinese/China	zh-CN
Chinese/Taiwan	zh-TW
Korean	Ko
Italian	It
Spanish	Es
Spanish/Spain	es-ES
Portuguese	Pt
Portuguese/Brazil	pt-BR

Warning If you have non-ASCII characters that appear before you specify the encoding, the browser cannot determine which type of encoding you have specified in the meta element.

Using the Character Entities

Character entities are numeric representations of characters, and they display the desired character regardless of the document encoding. So they come in handy. How to represent them in XHTML can become a bit confusing, however, because they can be represented in three ways:

- By numerical entities that use decimal values
- By numerical entities that use hexadecimal values
- By named entities

I'll take a look at each of these.

Numerical Entities That Use Decimal Values

Numerical entities can be defined by decimal values. With decimal values, you specify the character by using this syntax: `&#Unicodevalue;` where `Unicodevalue` is a unique number assigned to this symbol.

Tip *You can find Unicode values in many places, including at the World Wide Web Consortium at http://www.w3.org. Sometimes text editors, such as BBEdit, also list them. Look for something called an ASCII or Unicode table.*

Here's how you would code the decimal numerical entity for the division sign:

```
<body>
<p>I often use the &#247; sign.</p>
</body>
```

Tip Note the semicolon at the end of the entity. It is required with every entity, not just as a separator between two entities in a row.

Numerical Entities That Use Hexadecimal Values

Numerical representations of entities can also be in hexadecimal values. Hex values can usually be found in the same locations as the decimal values—Web sites and text editors.
To represent entities using hex values, you use the following syntax: `&#xhexadecimalvalue;` where the hexadecimal value of the character replaces `hexadecimalvalue`.

Here is the same example using the hexadecimal representation of the division-sign entity:

```
<body>
<p>I often use the &#xF7; sign.</p>
</body>
```

Referring to Entities by Name

Eventually it was realized that using the decimal and hexadecimal representations was not the most intuitive way to code entities. Slowly, names have evolved for the entities. They are a certainly more intuitive, but they are sometimes a bit odd.
Often, it's easy to think of several abbreviations for an entity. For example, you might wonder whether the named entity for the division sign is `&divide;`, `&div;`, or `&division;`. It's `&divide;`, and its use is demostrated in this example:

```
<body>
<p>I often use the &divide; sign.</p>
</body>
```

Tip Be sure to note that the named entities do not contain # following &. Also note that names do not exist for all the entities, at least not yet. So you may need to use the decimal or hex representations.

When to Use Character Entities

The Universal Character Set covers a lot of characters, including the standard ASCII characters on your keyboard. You may be wondering when to use entities and when it's acceptable to type the character you want. After all, if this book were on a Web page, it could have been written entirely with decimal or hexadecimal entities.

Tip Now that you're an expert on character entities, it might be useful to know that in Web-developer slang, to represent a character with an entity is to *escape* it.

The truth is, many Web browsers will be able to translate many of the characters you type. That is, it's okay to type a comma and a period rather than escaping them. And you can type all of the letters of the alphabet and numbers without worrying.

After that, it gets trickier. If you're not sure about a certain character, it's best to escape it. You should always check to see how the character looks when viewed through the Web browsers you wish to support. You should always replace the following five items with their respective entities:

- Quotation marks (" ")
- The less-than and greater-than signs (`< , >`)
- The ampersand (`&`)
- Any characters not commonly found in English
- Mathematical symbols

I'll look at each of these items in turn.

Quotation Marks

Quotation marks play an important role in XHTML because they are used to surround attributes in an XHTML element—for example: `<img src="foo.gif">`. Of course, quotation marks are also used to quote text on the page.

Generally speaking, it's best to escape all of the quotations in the general text. The most commonly used entity for this is the named one, `&quot;`. Here's an example:

```
<body>
<p>&quot;I love XHTML.&quot;</p>
</body>
```

Tip If you have "curly quotes" in your word processor or whatever text editor you are using to create your XHTML pages, be sure to turn them off. Curly quotes are tricky in that they are character entities.

The Greater-Than and Less-Than Signs

Greater-than and less-than signs should always be escaped when used in text because when unescaped, they represent the beginning and end of XHTML tags. The more recent browsers understand that a space around the unescaped signs means they are less-than and greater-than signs. But that feature can actually be a problem if you use < and > to surround an XHTML tag but accidentally leave a space within the brackets. The older browsers will ignore everything between the < and > tags, thinking that they are XHTML tags.

The most common way to escape these characters is by using the named entities: `&lt;` (less than) and `&gt;` (greater than). Here's an example:

```
<body>
<p>5 &lt; <br />
5 &gt; 3</p>
</body>
```

Other Mathematical Symbols

You should also always use entities to represent other mathematical symbols such as `+, =, x , -`, and `?`. Although the Web was initially invented as a way for scientists to easily share information, browsers vary quite a bit in their interpretations of mathematical symbols. It's best to use entities.

The Ampersand

Because the ampersand (`&`) indicates the beginning of an entity, it should also be escaped. Of all the named entities, `&amp;` is used most often. Here's an example of the ampersand named entity:

```
<body>
<p>The &amp; symbol is frequently escaped by using the &amp;amp;
entity.</p>
</body>
```

This example is a bit trickier than others you've seen. Note that the second ampersand is escaped and then immediately followed by `amp;`. Viewed in a browser, `&amp;amp;` will render "`&amp;`." Try it out for yourself.

Characters Not Commonly Found in English

The early Internet was primarily built by the U.S. government as a defense-agency project. Although many browsers exist, the primary ones used throughout the world were built to support ASCII automatically, other characters not so well. ASCII was invented by IBM, a U.S.-based company, and represents North American and European letterforms and punctuation.

So the Web supports certain letters and punctuation most easily and U.S.-based language the most. It's not that other languages cannot be supported or represented; but you usually have to escape any non-English characters, even European characters such as *Å*. Remember, if you can't type it directly from your keyboard without using special keys, it probably needs to be escaped.

Working with Images

Text-level elements don't cause paragraph breaks. They can be nested and can contain other text-level elements but not block-level elements. The `img` element—pronounced as *image*—is used to insert images *inline* within your text. Inline refers to a layout that features text and images together; this layout is what you're used to seeing on Web pages and in modern magazines and books.

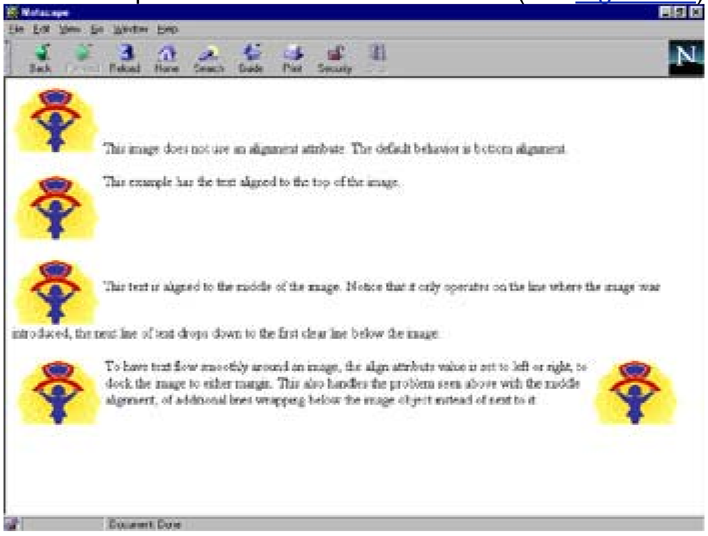
The `img` element is an *empty element*; that is, it is self-contained, so it needs the closing shorthand slash discussed in Chapter 3. At its most elementary syntax, `img` identifies the location (URL) of an image file to be displayed on your Web page. The basic syntax is as follows:

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

The `img` element is an easy one to remember, as saying it out loud can readily identify the purpose of each of its parts. The attribute `src` is the XHTML notation for *source*, which is the location of the file in URL form. Beyond the basic syntax of the `img` element, six significant types of attributes may be used within the element. Each attribute is important in its own right and can significantly change the way the image is displayed.

Alignment

An image file can be displayed in different positions relative to the line where it occurs within the page. The five possible values for this attribute (see Figure 4.6) are:



This image does not use an alignment attribute. The default behavior is bottom alignment.

This example has the text aligned to the top of the image.

This text is aligned to the middle of the image. Notice that it only operates on the line where the image was introduced, the next line of text drops down to the first clear line below the image.

To have text flow smoothly around an image, the align attribute value is set to left or right, to dock the image to either margin. This also handles the problem seen above with the middle alignment, of additional lines wrapping below the image object instead of next to it.

Figure 4.6: The `align` attribute is a powerful layout aid.

- **Top** The top of the image will be lined up with the top of the current line. When that line includes text, the top of the current line is where the top of the capital letters extends.
- **Middle** The baseline—the line that touches the bottom of letters that don't "drop down"—of the current line is aligned with the middle of the image.
- **Bottom** The bottom of the image sits on the baseline of the current line. This is the default alignment value.
- **Left** The image is docked to the left margin; its left border is flush with the margin. The text will flow around the image's right border until the first new line can reach the left-hand margin unimpeded by the bottom of the image.
- **Right** The image is docked to the right margin; its right border is flush with the margin. The text will flow around the image's left border until the first new line can reach the right-hand margin unimpeded by the bottom of the image.

Warning Special attention needs to be taken when using left or right alignment with images. Some older browsers may not "clear" an alignment (that is, they would have the text revert to the default position in relation to the image versus the previously stated right or left alignment) if text lines haven't completely flowed around it. This can result visually in a stair-step effect. To make sure that the next line begins on a completely clear line, insert a line break—<br />—adding an attribute for clear. The syntax appears as <br clear="all" />. Alternative values for clear are left and right, to drop to the first line clear to the left margin or clear to the right margin, respectively.

Images as Links

Even the newest Web surfer has likely seen images that link to other parts of a Web site. The process of creating a link using an image is simple. The markup includes the `img` tag nested within the anchor that provides the link, as shown here:

```
<a href="newfile.html"><img src="widget1.gif" /></a>
```

Images used as links are displayed by default with a single-pixel-wide border around them in the declared link color scheme. You have the option of changing that by adding the border attribute to your `img` tag. A value of 0 removes the border from around the image. The markup would look like this:

```
<a href="newfile.html"><img src="widget1.gif" border="0" /></a>
```

Image Dimensions: *width* and *height*

The `width` and `height` attributes describe the size of the image, measured in pixels. Browsers can display an image without knowing the image's size. But knowing the size allows the browser to reserve the space required for image display and to continue loading the text lines that follow the inline-image tag. As the images are sent from the Web server, they fill in the blocks of space reserved for them. The page will display faster, so your audience isn't kept looking at a blank screen, waiting for the entire page to finish loading. You've probably already seen this behavior while surfing the Web.
The syntax for a 100-pixel-wide × 100-pixel-tall graphic called `widget1.gif` is as follows:

```
<img src="widget1.gif" width="100" height="100" />
```

Warning Avoid the temptation to resize a graphic by changing the width and height values within the img element. Doing so will often greatly distort the image. If the image's physical size needs to be changed, use the resampling or cropping functions of your favorite graphics editor.

Alternative Text

Sometimes a visitor to your Web site can't see your images. A visitor may use a text-only browser, such as Lynx (see Chapter 2); use a speech synthesizer (see Chapter 20); or surf the Web with the browser's option to auto-load images turned off (which can be done in both Internet Explorer and Navigator). Fortunately, XHTML allows you to provide *alternative text* for your images.

Tip IE, version 3 and higher, and Navigator, version 4 and higher, display the alternative text as a tool tip when a visitor's mouse is placed over an image. This isn't required by the XHTML recommendation; it was a programming decision made by the companies. This appearance cannot be controlled with an XHTML element or attribute.

What should be entered as alternative text? You should enter a short, concise description of the image or the destination or action to be taken if the graphic serves as a link. If you had a photo of a widget to be used with a specific gadget, your alternative text could read "Widget 1 for use with Gadget 202." You could create that text by using the following syntax:

```
<img src="widget1.gif" alt="Widget 1 for use with Gadget 202" />
```

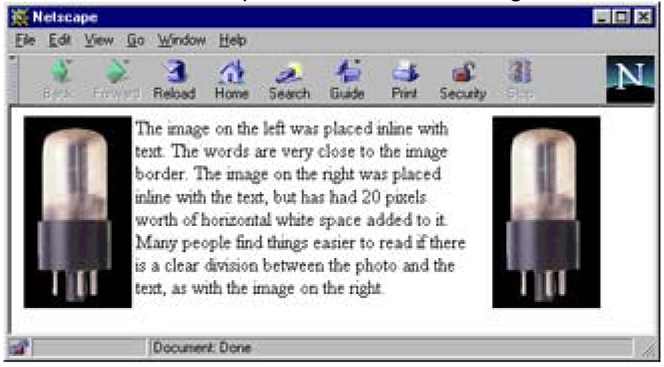
For images of purely decorative treatment, such as small bullets or other accents that don't provide information or navigation, an empty `alt` attribute—`alt=""`—should be used. Speech readers and other browsers that rely on `alt` content then know they can safely ignore those images.

Warning Almost any character can be used within an alternative-text block. One that can't be used is the quotation mark. Because all attribute values are enclosed in quotes, a quote in the middle of your alternative-text value will confuse the browser; the browser will interpret that first extra quote as the *end* of your alternative text. You can get around this problem by using single quotes or by avoiding them altogether.

White Space

Those familiar with the world of desktop publishing or other print processes will understand the concept of *white space*. White space is the absence of printing around an element on a page.

The default spacing around a graphic is a single typed space between the graphic and the next character of text in the same line. The default often produces a crowded look and makes the text near the graphics difficult to read. The screen shot in Figure 4.7 shows an inline image with the default amount of white space and an inline image where I've added a little extra white space.


Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Document: Done

The image on the left was placed inline with text. The words are very close to the image border. The image on the right was placed inline with the text, but has had 20 pixels worth of horizontal white space added to it. Many people find things easier to read if there is a clear division between the photo and the text, as with the image on the right.

Figure 4.7: An image without added white space compared to an image with added white space

On the left, the text crowds up against the image. On the right is more space, or "breathing room," for the image. It doesn't look squished.

The white space was added using the hspace attribute in the `img` element. The `hspace` and `vspace` attributes provide horizontal white space and vertical white space, respectively. The value is measured in pixels, with syntax such as the following:

```
<img src="widget1.gif" hspace="10" />
```

This syntax inserts 10 pixels of horizontal white space on *both sides* of the graphic, in addition to the default single space that is found between all images and the remaining inline text.

If you want white space on all four sides of the image, use both the `hspace` attribute and the `vspace` attribute, as follows:

```
<img src="widget1.gif" hspace="10" vspace="10" />
```

Image Mapping

When an image is going to be used as an image map, that fact can be declared in the `img` tag using either the `usemap` or `ismap` attribute. The `usemap` attribute is a function of client-side image maps, and `ismap` is used with server-side maps (infrequently used today). The construction of image maps, including alternative methods of handling them, such as the `map` element, is discussed in detail in Chapter 9.

Summary

In Chapter 3, you worked primarily with the structural framework, or canvas, of your XHTML documents. In this chapter, you've learned how to apply paint to that canvas, so to speak. Text can be presented in a variety of methods, from traditional paragraphs, lists, and quotations to various inline phrasal treatments.

Images provide an interesting visual component to the document, and incorporating them is quite easy with the `img` element. The alignment, spacing, and use of alternative text for images can be provided with additional attributes, such as `top`, `middle`, `bottom`, `left`, `right`, `alt`, `hspace`, and `vspace`.

Chapter 5: Creating Tables

Overview

Tables have a great impact on the look and feel of Web pages, more so than many features. In this section, I'll look at basic table structures and how they are made. For information about how to make your tables accessible for visually impaired people or for those with browsers that do not support tables, see Chapter 20.

Tables evolved significantly between HTML 3.2 and HTML 4.01. XHTML 1 carried forward the table functionality, yet cut some basic table elements when XHTML Modularization was introduced (see Chapter 21). For example, the basic tables module takes away some of the more expansive table options, such as table heads and footers, which were introduced in HTML 4.01. But these elements are available in the larger tables module. In Modularization, the simplified tables are to accommodate the audience accessing sites from alternative and smaller devices. Most up-to-date, desktop-oriented browsers handle the newer features of tables readily.

This chapter covers the following topics:

- Creating basic tables
- Table captions and borders
- Alignment of tables and cells
- Cell spacing and padding
- Grouping of columns and rows

What Are Tables?

If you've spent even a small amount of time on the Web, you've probably seen tables. HTML tables consist of rows and columns, just like tables in print. The boxes formed by the intersection of the rows and columns are called *cells*. Tables can format data, make separate areas for toolbars, and generally aid in the layout and design of information. Because tables have so many formatting options, they are a tremendous contribution to Web design. Not all browsers support tables, though, so you should read Chapter 20 for alternatives.

Figure 5.1 shows the StarGate SG-1 Web site (`http://www.stargate-sg1.com/index2.cfm`), which is a great example of what can be done with tables. Layout for the various segments of this data screen begins with tables. If you're interested in seeing a longer piece of sample code than this chapter contains, go to this site and view the source code.

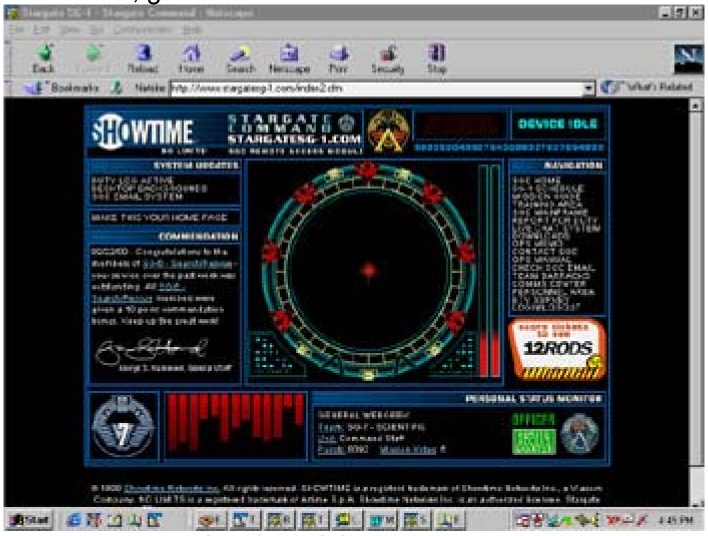


Figure 5.1: The StarGate SG-1 Web site

Tip If you're not familiar with StarGate SG-1, it's a popular science-fiction series that airs on the Showtime cable television network in the United States, in reruns on some traditional broadcast stations, and a series year or two behind in Canada and other countries.

Creating a Basic Table

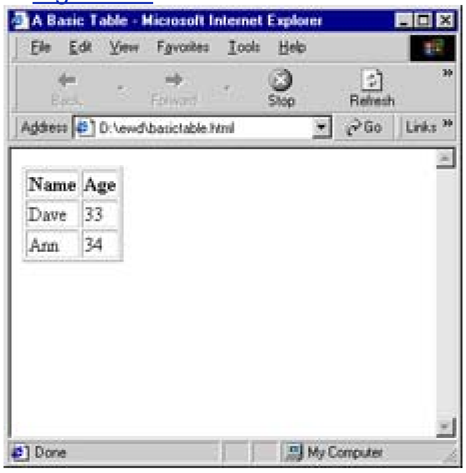
The best way to learn how to build a table is to look at the XHTML and then examine the results in a browser. Listing 5.1, also on the CD, shows the most basic XHTML for making a table.

Listing 5.1: basictable.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>A Basic Table</title>

</head>

<body>

<table border="1">
```

```
<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>

</tr>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

Figure 5.2 shows what the table looks like in IE 5.5 on Windows 98.


A Basic Table - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\basictable.html Go Links
Done My Computer

Name	Age
Dave	33
Ann	34

Figure 5.2: The rendering of `basictable.html`, as viewed in IE 5.5 on Windows 98

The XHTML elements used are shown in Table 5.1. I'll examine them more closely.

Table 5.1: A Closer Look at the XHTML Elements Used in `basictable.html`

XHTML ELEMENT	USE
`table`	This element surrounds the table and delineates where it begins and ends.
`tr`	The element tells the browser there's a table row, and it surrounds the table cells within the row.
`td`	The element tells the browser there's a table cell and surrounds the data of the cell. That is, it says what's inside the defined cell.

Based on this information, you can see that the table in Figure 5.2 has three table rows defined and that each table row contains two table data cells. In the first cell of the first row, the data contains "Name"; in the second cell of the first row, the data is "Age," and so on.

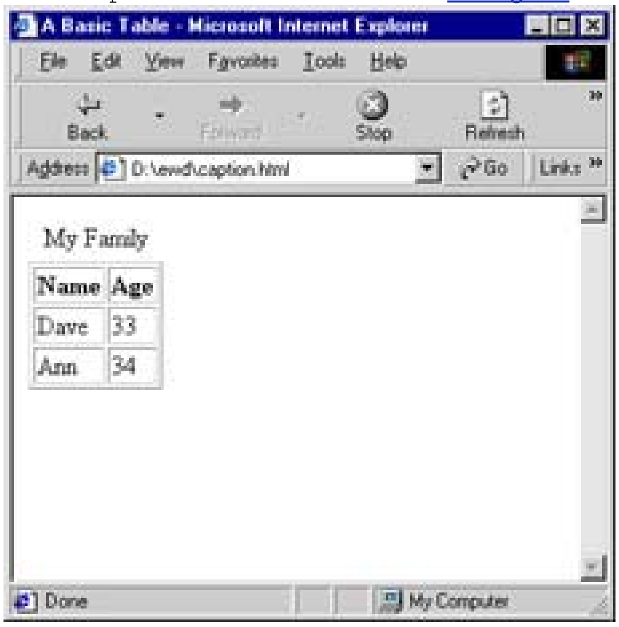
Tip The cells of a table can contain many types of elements, including forms, images, lists, and other tables. They may also contain preformatted text and more than one paragraph.

Table-Formatting Options

Now you've created a simple table. If you want to spice it up, you can try many types of formatting. In this section, I'll add one feature at a time to the sample XHTML and then look at the effect of the code in a browser.

Captions

If you want, you can add a caption to your table, like the caption shown in Figure 5.3. To do so, you use the `caption` element as shown in Listing 5.2 of `caption.html`, which is also found on the CD.


A Basic Table - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\caption.html
Go Links
Done
My Computer

My Family

Name	Age
Dave	33
Ann	34

Figure 5.3: The rendering of `caption.html` in IE 5.5 on Windows 98

Listing 5.2: caption.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>A Basic Table</title>

</head>

<body>

<table border="1">

<caption>My Family</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>
```

```
</tr>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>\
```

You can specify where the caption is located relative to the table by using the `align` attribute. For example, using `<caption align="bottom">Put your caption here</caption>` places the caption beneath the table. Try it and see.

Borders

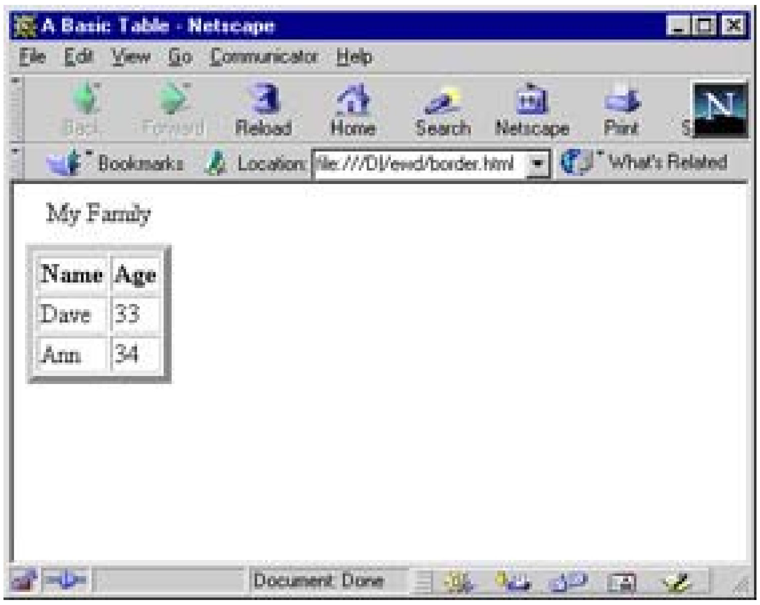
You've probably seen tables with borders around them. There are many options for borders, including options for placement and thickness. I'll take a quick look at a very simple border before moving on to more sophisticated border options.
In Listing 5.3 here and on the CD, I'll adjust the `border` attribute value on the `table` element to `5`, referring to the width of the border measured in pixels.

Listing 5.3: border.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>A Basic Table</title>

</head>

<body>

<table border="5">

<caption>My Family</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>
```

```
</tr>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

Figure 5.4 displays what `border.html` looks like in Navigator 4.73. The raised beveled look on the outer border is a result of the increased width.


A Basic Table - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location: file:///D|/ewd/border.html What's Related
Document: Done

My Family

Name	Age
Dave	33
Ann	34

Figure 5.4: The rendering of `border.html` in Navigator 4.73

For more interesting effects, I can manipulate the presentation of the outer border using the `frame` attribute on the `table` element. Table 5.2 shows the possible values and describes their prescribed results.

Table 5.2: Possible Values for the `frame` Attribute

VALUE	RESULTING TABLE
`void`	A table with no sides. This is the default value.
`above`	A border on the top side only.
`below`	A border on the bottom side only.
`lhs`	A border on the left side only.
`rhs`	A border on the right side only.

Table 5.2: Possible Values for the `frame` Attribute

VALUE	RESULTING TABLE
`hsides`	A border on the top and bottom sides.
`vsides`	A border on the left and right sides.
`box`	A border on all four sides.
`border`	A border on all four sides. The `box` and `border` values are interchangeable.

Using the `void` value for the `frame` attribute, I can create a table that looks like a traditional tic-tac-toe board, as shown in Listing 5.4 and also on the CD-ROM.

Listing 5.4: tictactoe.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>The Outer Frame Border</title>

</head>

<body>

<table frame="void" border="2">

<tr>

<td>X</td>

<td>&nbsp;</td>

<td>O</td>

</tr>

<tr>

<td>O</td>

<td>X</td>

<td>O</td>

</tr>

<tr>

<td>X</td>

<td>&nbsp;</td>

<td>&nbsp;</td>

</tr>

</table>
```

```
</body>

</html>
```

Warning Though the frame attribute for tables has been around since HTML 4, it still is not widely implemented. Only the latest of browsers supports this option. Internet Explorer 5.5 performs as intended, and Navigator 6 removes the outside border but still leaves a distinct impression as to where the cells' edges are (see Figures 5.5 and 5.6).

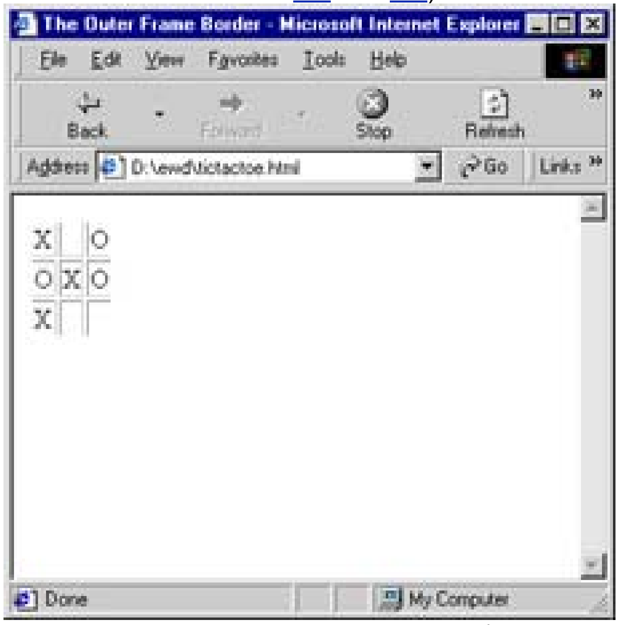

The Outer Frame Border - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\tictactoe.html
Go Links
X O
O X O
X
Done
My Computer

Figure 5.5: The tic-tac-toe grid in IE 5.5 (Windows 98)

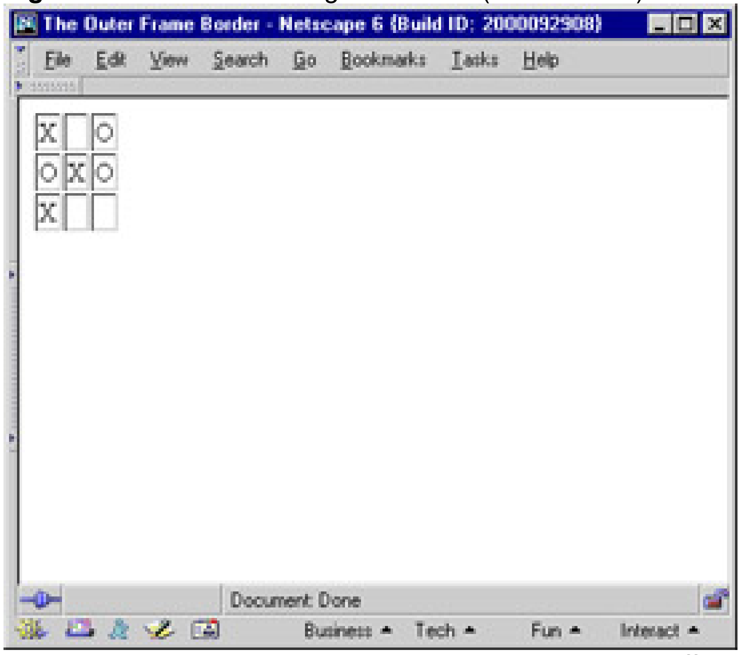

The Outer Frame Border - Netscape 6 (Build ID: 2000092908)
File Edit View Search Go Bookmarks Tasks Help
X O
O X O
X
Document: Done
Business Tech Fun Interact

Figure 5.6: The same grid in Navigator 6 with a slightly different presentation

Rules

The `frame` attribute applies to the outer border of the table. Starting in HTML 4 and again in XHTML, you can also specify the appearance of the internal borders of rows and columns by using the `rules` attribute. Listing 5.5, also on the CD-ROM, shows an example.

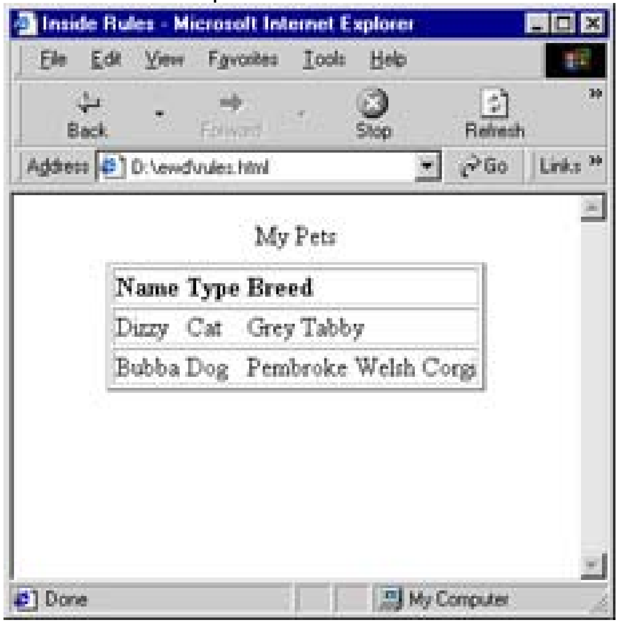
Listing 5.5: rules.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Inside Rules</title>

</head>

<body>

<div align=center>

<table rules="rows" border="2">

<caption>My Pets</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Type</strong></td>

<td><strong>Breed</strong></td>

</tr>

<tr>

<td>Dizzy</td>

<td>Cat</td>

<td>Grey Tabby</td>

</tr>

<tr>

<td>Bubba</td>

<td>Dog</td>

<td>Pembroke Welsh Corgi</td>

</tr>

</table>

</div>

</body>

</html>
```

Warning Like the frame attribute, the rules attribute isn't well supported. So far, only IE 5.5 on the Windows platform produces the intended results.

In Listing 5.5, `rules` would only be applied between rows and not between columns (see Figure 5.7). You could also choose between columns (`rules="cols"`) and between all cells (`rules="all"`). For reference, you can also use rules between cell groups (`rules="groups"`). Cell groups are covered later in this chapter.


Inside Rules - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\rules.html
Go Links
Done
My Computer

My Pets

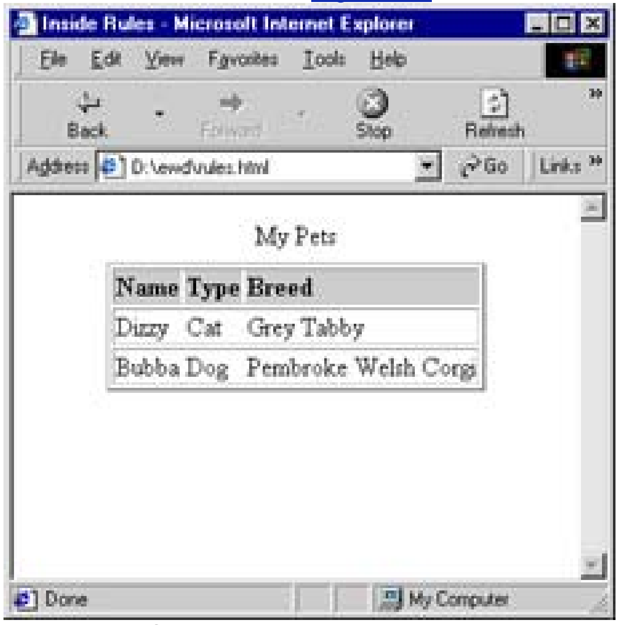
Name	Type	Breed
Dizzy	Cat	Grey Tabby
Bubba	Dog	Pembroke Welsh Corgi

Figure 5.7: The `rules` attribute can be used to remove internal borders between table cells.

Colors

You can add color to a table, which is done by using the `bgcolor` attribute on any of the `table`, `tr`, or `td` elements. This attribute uses the syntax `bgcolor="`*`color`*`"`. The preferred color values are written in hex notation, as covered in Chapter 3 in the section "Background Color." Taking another look at the `rules.html` example in Listing 5.5, I can color the first row, which serves as column headings, by adding `bgcolor` to the `tr` element, as shown:

<tr bgcolor="#CCCCCC">
The results are seen in Figure 5.8.


Inside Rules - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\rules.html
Go Links
Done
My Computer

My Pets

Name	Type	Breed
Dizzy	Cat	Grey Tabby
Bubba	Dog	Pembroke Welsh Corgi

Figure 5.8: Cells, rows, columns, or entire tables can be highlighted using background color.

Alignment

Along with all the other types of formatting, XHTML also allows the user to align text within the cells. You can choose from three different types of alignments: horizontal, vertical, and by character.

Tip Even though the *alignment* refers to alignment of data within cells, you can use the align attribute like the bgcolor attribute—what it affects depends on where you place it. That is, place it in the table element to align the entire table, in the tr element to align the entire row, and in the td element to align the contents of a single cell.

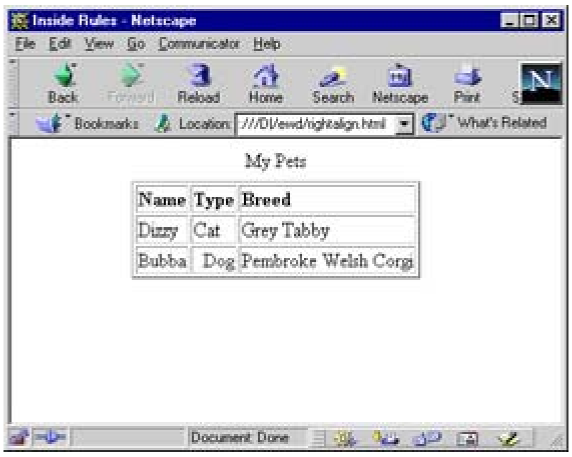
Horizontal Alignment

Revisiting the table of my family pets in Listing 5.6 (and on the CD-ROM), I'll align the contents of the third column to the right (see Figure 5.9). For now, I am going to align each cell in the column individually. Later, when I cover groups of rows, you'll see how attributes can be applied to a group of rows in order to affect an entire column.

Listing 5.6: rightalign.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Inside Rules</title>

</head>

<body>

<div align=center>

<table border="2">

<caption>My Pets</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Type</strong></td>

<td><strong>Breed</strong></td>

</tr>

<tr>

<td>Dizzy</td>

<td>Cat</td>

<td>Grey Tabby</td>

</tr>

<tr>

<td align="right">Bubba</td>

<td align="right">Dog</td>

<td align="right">Pembroke Welsh Corgi</td>

</tr>

</table>
```

```
</div>

</body>

</html>
```


Inside Rules - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location: ///D|/ewd/rightalign.html What's Related
Document: Done

My Pets

Name	Type	Breed
Dizzy	Cat	Grey Tabby
Bubba	Dog	Pembroke Welsh Corgi

Figure 5.9: Right alignment in individual cells using the `align` attribute

Other horizontal-alignment options include `left`, `center`, `justify`, and `char`. The first three options work just like `right`. However, the `char` value still remains to be implemented in any major browser. The intention is to allow alignment on a specific character, such as a decimal point in numbers, which would be highly desirable when presenting a large table full of accounting figures. Properties used in Cascading Style Sheets (see Chapter 7) are supposed to manage this feature, but again, browser implementation is scarce.

Vertical Alignment

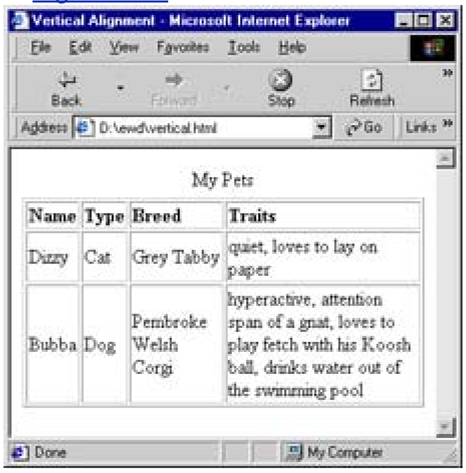
You also have the option of aligning text vertically within a cell. This is particularly helpful to give a uniform presentation when working with cell contents of various sizes. In Listing 5.7, here and on the CD-ROM, I'll create another table describing my pets, with some additional information added to it about their traits.

Listing 5.7: vertical.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Vertical Alignment</title>

</head>

<body>

<div align=center>

<table border="1" >

<caption>My Pets</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Type</strong></td>
```

```
<td><strong>Breed</strong></td>

<td><strong>Traits</strong></td>

</tr>

<tr>

<td>Dizzy</td>

<td>Cat</td>

<td>Grey Tabby</td>

<td>quiet, loves to lay on paper</td>

</tr>

<tr>

<td>Bubba</td>

<td>Dog</td>

<td>Pembroke Welsh Corgi</td>

<td>hyperactive, attention span of a gnat, loves to play fetch with

  his Koosh ball, drinks water out of the swimming pool</td>

</tr>

</table>

</div>

</body>

</html>
```

Figure 5.10 demonstrates how `vertical.html` is rendered by IE 5.5 (Windows 98).


Vertical Alignment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Address D:\ewd\vertical.html
Go
Links
Done
My Computer

My Pets

Name	Type	Breed	Traits
Dizzy	Cat	Grey Tabby	quiet, loves to lay on paper
Bubba	Dog	Pembroke Welsh Corgi	hyperactive, attention span of a gnat, loves to play fetch with his Koosh ball, drinks water out of the swimming pool

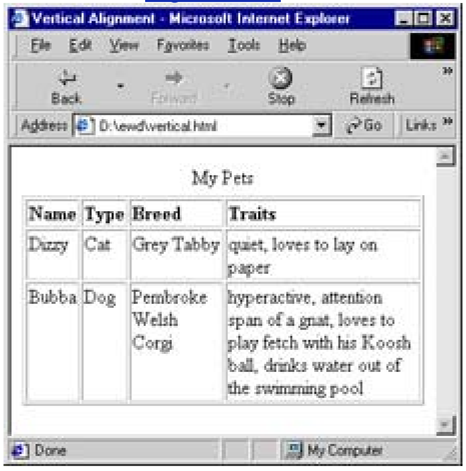
Figure 5.10: The code in `vertical.html` produces cells with the contents vertically aligned differently in each column.

The text in the right column, describing the animal's traits, fills the cells. The other columns are aligned to the default, which is the middle of the cell. It can be an awkward visual presentation. So instead, I want to vertically align each to the top of the cell. To do this, I need to add the `valign` attribute to each `tr` element, as shown:

<tr valign="top">
As with horizontal alignment, the `valign` attribute can be placed on the table, table row, or table cell. The potential values for this attribute are `top`, `middle`, `bottom`, and `baseline`.

Tip The baseline valign attribute value has a very specific definition, one that I think makes its usefulness rather questionable. To quote from the W3C Recommendation, "All cells in the same row as a cell whose valign attribute has this value should have their textual data positioned so that the first text line occurs on a baseline common to all cells in the row. This constraint does not apply to subsequent text lines in these cells." Experimentation is the best way to learn how this value makes a browser behave.

By adding a `valign` attribute to each table row in the `vertical.html` file, I get a new presentation as shown in Figure 5.11.


Vertical Alignment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\vertical.html
Go Links
Done
My Computer

My Pets

Name	Type	Breed	Traits
Dizzy	Cat	Grey Tabby	quiet, loves to lay on paper
Bubba	Dog	Pembroke Welsh Corgi	hyperactive, attention span of a gnat, loves to play fetch with his Koosh ball, drinks water out of the swimming pool

Figure 5.11: Aligning cell content to the top of each cell can enhance readability of table contents.

Cell Margins
Another table-spacing option is the spacing within and between cells. The attributes that allow you to specify these options are called `cellpadding` and `cellspacing`, respectively.

The *cellpadding* Attribute

The `cellpadding` attribute is used to describe how much empty space is around the content in a cell. It can be specified as a number of pixels or as a percentage of the cell. Using a pixel value is certainly the easiest, but a percentage may be desirable in some instances. When you specify percentage, it is divided into two equal parts: half on each horizontal side. The vertical padding is also split 50/50. That is, if you specify that the padding equals 30 percent of the cell, 15 percent of the width of the cell will be above the data, 15 percent below the data, and 15 percent on each side of the data.

For example, if the cell is 100 pixels wide, and you specify a border of 30 percent, the browser will calculate 30 percent of the 100 pixels, divide that number (30 pixels) by two, and place a cell padding of 15 pixels all the way around the cell.

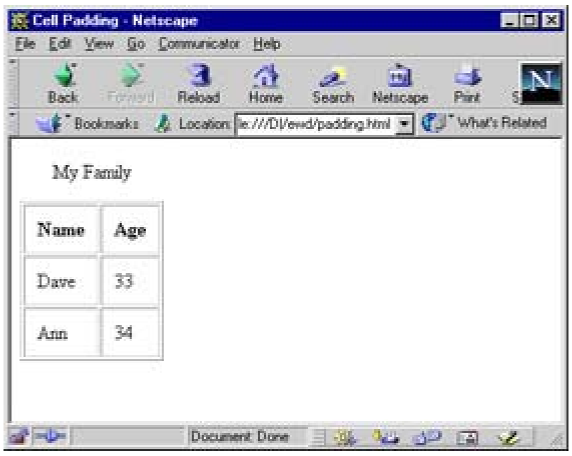
Tip The division of the padding value only occurs when the padding is described as a percentage. A specific pixel value will result in that many pixels placed on all four sides of the cell content.

In Listing 5.8, which is also on the CD-ROM, I'll add some padding to a simple 2 x 2 table (see also Figure 5.12).

Listing 5.8: padding.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Cell Padding</title>
```

```
</head>

<body>

<table border="1" cellpadding="10">

<caption>My Family</caption>

<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>

</tr>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```


Cell Padding - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location: le:///D|/ewd/padding.html
What's Related
Document: Done

My Family

Name	**Age**
Dave	33
Ann	34

Figure 5.12: The `padding.html` file produces a larger table with space surrounding the cell data.

Note the new padding around the text. There are 10 pixels all around, but it does not look even because the text is aligned by default to the left side of the cell.

The *cellspacing* Attribute

The `cellspacing` attribute is similar to `cellpadding`, but it adds space between cells rather than adding space within the cells. Without `cellspacing`, cells are right next to each other—they touch, and content is kept apart only by the border value.

Mixing `cellspacing` values and border values can have interesting results. One of the easiest ways to see the impact of `cellspacing` is to use a background color for the cells, which I'll do in Listing 5.9 here and on the CD-ROM. The portion between the cells that remains white is the `cellspacing` area. Here, the border is specified as a single pixel. See Figure 5.13.

Listing 5.9: spacing.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Cellspacing</title>

</head>

<body>

<table border="1" cellspacing="20">



<caption>My Family</caption>

<tr bgcolor="yellow">

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>

</tr>

<tr bgcolor="yellow">

<td>Dave</td>

<td>33</td>

</tr>

<tr bgcolor="yellow">

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

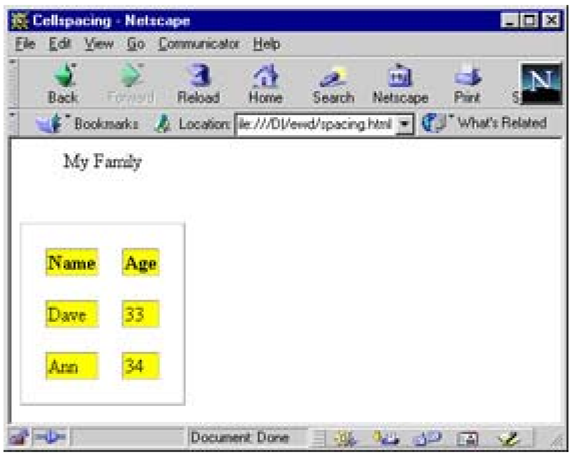

Cellspacing - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks Location: file:///D|/ewd/spacing.html What's Related
My Family
Name Age
Dave 33
Ann 34
Document: Done

Figure 5.13: In `spacing.html`, coloring the cell contents with a background color clearly shows the placement of the extra spacing between cells.

Experiment with border values while using cell spacing to see the effect on the entire table. In this case, the combination of wide spacing and a border renders the table with the border only showing the width on the outer frame.

Grouping Rows and Columns

You've had a chance to see how to make a basic table and to do some fancier formatting. Now you'll see the process of grouping together rows and columns and how that affects their size. You're also going to learn how to specify that a cell span more than one row or column.

Tip You can apply the formatting you've already learned, such as color, borders, and alignment, to groups.

Grouping was new to HTML 4 and is just now being supported in popular Web browsers.

The *colgroup* Element

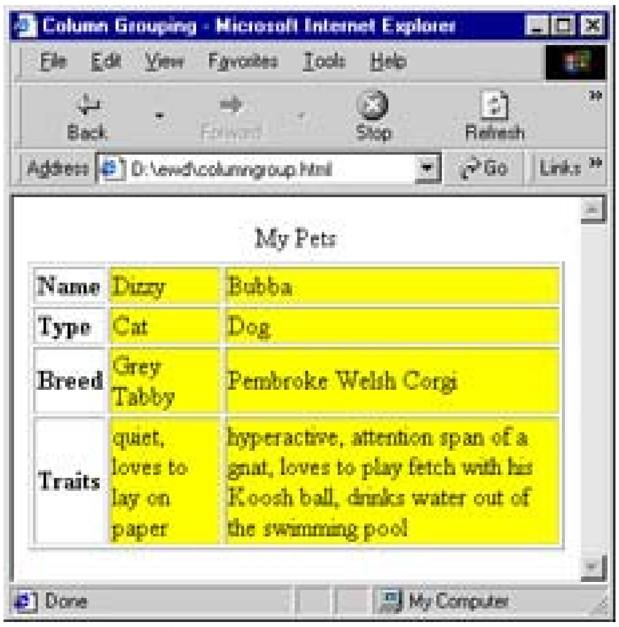
Columns are grouped in order to apply similar presentation properties with a single set of attributes. For instance, a table with header cells across both the top of the table and down the left side of the table might want to group the inner columns together to offset them from the header with a different color, font face, or other visual treatment. In Listing 5.10, I'll use the `columngroup` element to apply the same color to more than one column in a single statement. Listing 5.10 is also on the CD-ROM.

Listing 5.10: columngroup.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Column Grouping</title>

</head>

<body>

<div align=center>

<table border="1" >

<caption>My Pets</caption>

<colgroup span="1">
```

```
<colgroup span="2" bgcolor="yellow">

<tr>

<td><strong>Name</strong></td>

<td>Dizzy</td>

<td>Bubba</td>

</tr>

<tr>

<td><strong>Type</strong></td>

<td>Cat</td>

<td>Dog</td>

</tr>

<tr>

<td><strong>Breed</strong></td>

<td>Grey Tabby</td>

<td>Pembroke Welsh Corgi</td>

</tr>

<tr>

<td><strong>Traits</td>

<td>quiet, loves to lay on paper</td>

<td>hyperactive, attention span of a gnat, loves to play fetch with

  his Koosh ball, drinks water out of the swimming pool</td>

</table>

</div>

</body>

</html>
```

The `colgroup.html` file specifies two column groups. The first group contains just a single column, dictated by the `span` attribute, with a value of `1`. The second column group has a `span` attribute value of `2`. Always work left to right in the table; so the first group is the left-hand column, and the second group is the remaining two columns, which I've colored yellow using the `bgcolor` attribute on the second `colgroup` element (see Figure 5.14).


Column Grouping - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address D:\ewd\columngroup.html
Go Links
Done
My Computer

My Pets

Name	Dizzy	Bubba
Type	Cat	Dog
Breed	Grey Tabby	Pembroke Welsh Corgi
Traits	quiet, loves to lay on paper	hyperactive, attention span of a gnat, loves to play fetch with his Koosh ball, drinks water out of the swimming pool

Figure 5.14: Column grouping is used to impose attributes on more than one column at a time.

The *col* Element

Columns can also be manipulated with the `col` element, which can either be a child element of `colgroup` or operate on columns by itself. The key difference is that `colgroup` defines a structure within the table, whereas the `col` element is empty of structure, used only as a placeholder for attributes.

Tip Why the distinction between structure and an attribute placeholder when the results look the same? Remember, not every user agent (often the browser, but it can be any other program rendering the display) presents tables in the same fashion as you're used to on desktop PCs. When putting columns together into a defined structure, the user agent is told to present them as a whole before moving on to other structures that follow. This has its good and bad points when working with smaller devices such as cell phones and personal desktop assistants. The display may look a little odd, but using the structural method, all of the information in the structure would be displayed together (or sequentially if there's not enough screen space), before moving on to the next structure.

Grouping Rows

You've already seen the benefits of grouping columns, and it follows that you can group rows in much the same manner. In fact, the W3C has defined three special row groups: the table head, table body, and table foot.

The Table Head

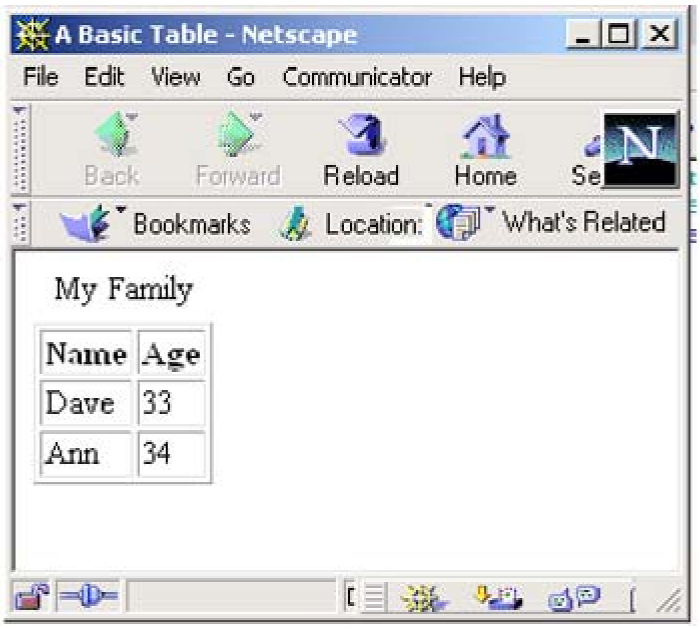
As its name implies, the table-head row grouping places headings at the top of each column. This is something you've seen done in earlier examples, except I did it with a traditional table row and strong elements inside each cell to visually set off the heading. The `thead` element does this for you automatically. It contains one or more table rows and the cells within them. In Listing 5.11 (see also the CD-ROM), I'll use an earlier example, `caption.html`, but this time with the thead element to create the table head.

Listing 5.11: thead.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>A Basic Table</title>
```

```
</head>

<body>

<table border="1">

<caption>My Family</caption>

<thead>

<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>

</tr>

</thead>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

Figure 5.15 shows the results, which are visually the same as Figure 5.3. Today's Web browsers have traditionally been programmed to give emphasis to a heading, which is why the heading cell content shows up in boldface type.


A Basic Table - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Se
Bookmarks Location: What's Related

My Family

Name	Age
Dave	33
Ann	34

Figure 5.15: Cell headers created using a `thead` element

Table heads define whichever row you consider to be the head of the table, which is usually the first row.

The Table Foot

The foot of the table can be defined using the `tfoot` element. Like `thead`, it contains one or more table rows. However, it differs in that the proper placement of this row *by the browser* is at the very bottom of the table. Interestingly enough, when used in the document, `tfoot` occurs immediately after `thead`. In Listing 5.12, which is also on the CD-ROM, I'll put a footer row on `thead.html` to demonstrate.

Listing 5.12: tfoot.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Table Head and Foot</title>

</head>

<body>

<table border="1">

<caption>My Family</caption>

<thead>

<tr>

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>



</tr>
```

```
</thead>

<tfoot>

<tr>

<td colspan="2">August 17, 1996</td>

</tr>

</tfoot>

<tr>

<td>Dave</td>

<td>33</td>

</tr>

<tr>

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

Figure 5.16 shows how this code looks in a browser. In this figure, the code is rendered in Navigator 6.

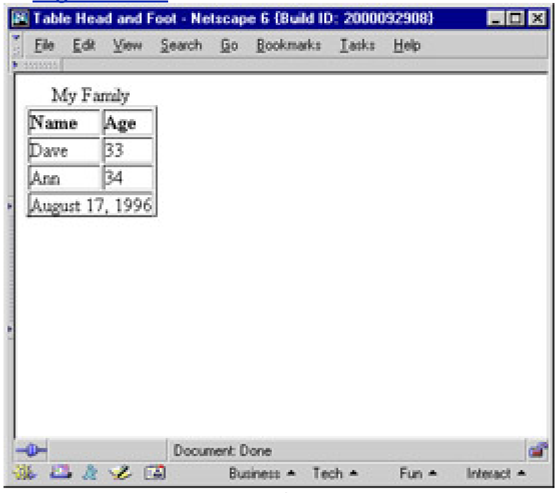


Table Head and Foot - Netscape 6 (Build ID: 2000092908)
File Edit View Search Go Bookmarks Tasks Help
Document: Done
Business Tech Fun Interact

My Family

Name	Age
Dave	33
Ann	34
August 17, 1996	

Figure 5.16: The table foot can complement the dat inside the table.

Warning The tfoot element has only begun to be properly rendered by the browsers. Its required placement immediately after thead in the markup causes many browsers, even Navigator 4.x versions, to render it in the same place, just below the thead content. Today, Internet Explorer 5.5 and Navigator 6 both render tfoot as XHTML intended it to be displayed.

Finally, when working with the `thead` and `tfoot` structures, a `tbody` structure is also available to complete the logical divisions of the table. The `tbody` element contains all remaining rows and cells, and is the last element to be closed before closing the table itself, as shown here (and on the CD-ROM) in Listing 5.13.

Listing 5.13: tbody.html

```
<!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" xml:lang="en" lang="en">
<head>
<title>Table Head and Foot</title>
</head>
<body>
<table border="1">
<caption>My Family</caption>
<thead>
<tr>
<td><strong>Name</strong></td>
<td><strong>Age</strong></td>
</tr>
</thead>
<tfoot>
<tr>
<td colspan="2">August 17, 1996</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Dave</td>
<td>33</td>
</tr>
<tr>
<td>Ann</td>
<td>34</td>
</tr>
</tbody>
```

```
</table>

</body>

</html>
```

Cells That Span More than One Column or Row

You may have noticed in the `tfoot` section of the last couple examples that a new attribute appeared in the `td` element within the `tfoot` element: `colspan`. The `colspan` attribute dictates how many columns an *individual cell* should occupy. The same can be done within a row as well, using the `rowspan` attribute. The table in Listing 5.14 (see also the CD-ROM) shows examples of column spanning, row spanning, and spanning of both columns and rows. I'll color the background of each cell in a different hue to allow you to easily see the divisions (see Figure 5.17).

Listing 5.14: spanning.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Spanning Rows and Columns</title>

</head>

<body>

<table border="1" cellpadding="20">

<tr>

<td bgcolor="white">&nbsp;</td>

<td bgcolor="yellow">&nbsp;</td>

<td bgcolor="pink">&nbsp;</td>

</tr>

<tr>

<td colspan="2" bgcolor="green">&nbsp;</td>

<td bgcolor="purple"></td>

</tr>

<tr>

<td rowspan="2" bgcolor="blue">&nbsp;</td>

<td bgcolor="gray">&nbsp;</td>

<td bgcolor="teal">&nbsp;</td>

</tr>
```

```
<tr>

<td bgcolor="fuschia">&nbsp;</td>

<td bgcolor="olive">&nbsp;</td>

</tr>

</table>


</body>

</html>
```

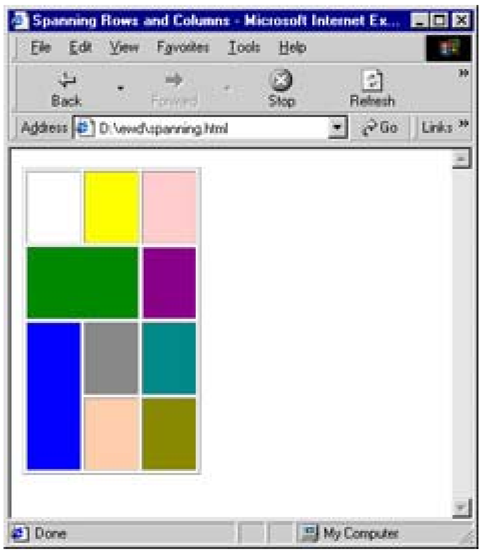

Spanning Rows and Columns - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Address D:\ewd\spanning.html
Go
Links
Done
My Computer

Figure 5.17: Cells can span both columns and rows.

Summary

In this chapter, I covered the ins and outs of tables: the basics of managing captions, borders, and alignment. Delving deeper into the structure of the table, you learned how to manipulate the outer frame and the inner rules. Columns and rows can be grouped into logical structures or simply spanned with presentational attributes. Have fun in your design playing with these attributes!

Chapter 6: Working with Frames

Overview

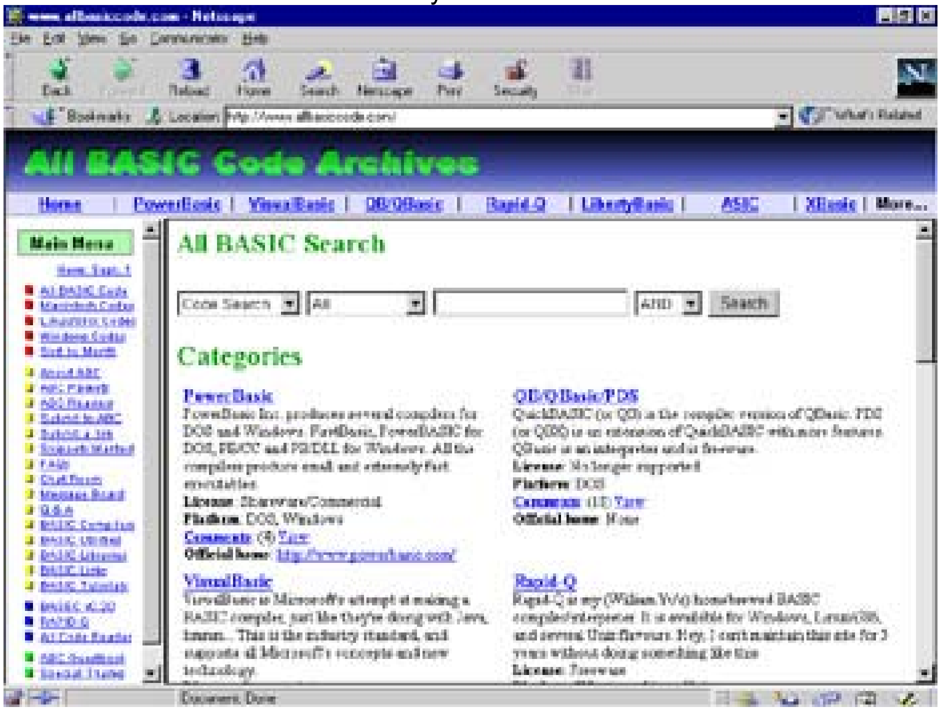
Sometimes it's desirable to have some content always remain visible within a Web page. Most software programs have menu bars and icons that always remain in view, regardless of what else changes on the screen. Similarly, Web designers will likely wish to provide navigation content or branding information, such as a logo, in a static area. *Frames* were developed to fill this need. In this chapter, you will learn how to implement frames.

This chapter covers the following topics:

- Using frames to set off content
- Creating the frame set
- Design issues for frames
- Providing an alternative to frames for older browsers
- Inserting inline frames

Using Frames

Frames divide the browser window into two or more individual sections, which will operate as if they were independent browser windows. XHTML documents that make use of frames have a unique structure that may seem to violate many of the rules set down in the previous chapters. Figure 6.1 shows a site with a three-frame layout.



PowerBasic Inc. produces several compilers for DOS and Windows: FirstBasic, PowerBASIC for DOS, PB/CC and PB/DLL for Windows. All the compilers produce small and extremely fast executables.
License: Shareware/Commercial
Platform: DOS, Windows
Comments: (4) View
Official home: http://www.powerbasic.com/

QuickBASIC (or QB) is the compiler version of QBasic. PDS (or QBX) is an extension of QuickBASIC with more features. QBasic is an interpreter and is freeware.
License: No longer supported
Platform: DOS
Comments: (11) View
Official home: None

VisualBasic is Microsoft's attempt at making a BASIC compiler, just like they're doing with Java, [illegible]... This is the industry standard, and supports all Microsoft's concepts and new technology.

Rapid-Q is my (William Yu's) homebrewed BASIC compiler/interpreter. It is available for Windows, Linux/i386, and several Unix flavours. Hey, I can't maintain this site for 3 years without doing something like this.
License: Freeware

Figure 6.1: A Web site layout that makes use of frames

The *frameset* Element

The `frameset` element is a container that defines the borders to be laid out for content presentation within the main browser window.

The individual frames within the browser window are defined in two dimensions: rows and columns. The *row count* is the number of horizontal sections to be created within the browser window. The number of vertical sections is the *column count*. Rows are described by the `rows` attribute and columns by the `cols` attribute.

Each attribute has an implied value of `1`. So if the attribute is not set within the `frameset` element, a frames-capable browser will assume a single horizontal or vertical space that results in a display across the full width or height of the browser window.

The following is the basic XHTML for a `frameset` element:

<frameset rows="*value*, *value*" COLS="*value*, *value*">

</frameset>

The format of the value for the `row` and `cols` attributes is a bit different than what you've seen so far. Each of these attributes contains a comma-delimited list of values. Imagine you want to divide a browser window into four equal quadrants. To do that, you need two rows and two columns. The size of each row or column is declared in the list of values for each of those attributes, as shown in the example frameset container just presented. What values should be used for each attribute? The answer can be found in the upcoming section "Using a Percentage of the Browser Window."

Allocating Space between Frames

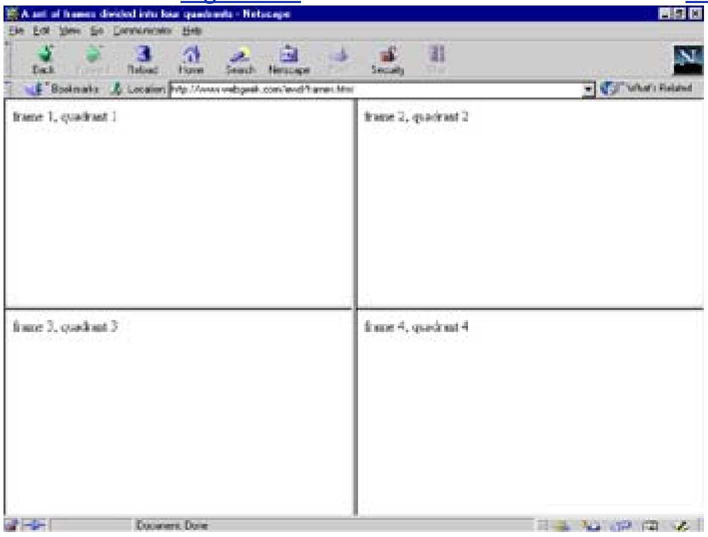
The values used for row widths and column lengths can be set in three different ways: in fixed-pixel values, in a percentage of the browser window, or by using *relative spacing*.

Setting Fixed-Pixel-Sized Frames

When you want to present an exact-size frame, you use fixed-pixel values. Remember that a pixel remains a pixel, regardless of the visitor's screen resolution. A 200-pixel-wide frame will take up just under one third of the browser window when viewed at full screen on a monitor set to 640 × 480 resolution. But that same 200-pixel-wide frame will take up just one quarter of the browser window when viewed at full screen on a monitor set to 800 × 600 resolution. Frames with fixed-pixel values often hold logos in a horizontal frame across the top of a site or hold navigational information in a fixed frame.

Using a Percentage of the Browser Window

The browser window can be divided up using percentages only if the dimensions of all frames can be expressed in percentages. If you wish to divide a browser window into four equal quadrants, the use of percentages would be the most obvious choice. The rows attribute would hold a value of "50%, 50%" and the COLS attribute would hold the same value of "50%, 50%". The browser would divide the available window space in half both horizontally and vertically, resulting in four evenly sized quadrants, as shown in Figure 6.2 and in the file frames.html in Listing 6.1.



A set of frames divided into four quadrants - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location
What's Related
frame 1, quadrant 1
frame 2, quadrant 2
frame 3, quadrant 3
frame 4, quadrant 4
Document: Done

Figure 6.2: The browser window divided into four evenly sized quadrants by a frameset element

Listing 6.1: frames.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>A set of frames divided into four quadtrants</title>

</head>

<frameset rows="50%, 50%" cols="50%, 50%">



<frame src="frame1.html" frameborder=1 name="q1" />
```

```
<frame src="frame2.html" frameborder=1 name="q2" />

<frame src="frame3.html" frameborder=1 name="q3" />

<frame src="frame4.html" frameborder=1 name="q4" />

</frameset>

</html>
```

Two items are noteworthy in this example. The first is the change from the XHTML 1 Transitional `DOCTYPE` declaration to the XHTML 1 Frameset `DOCTYPE` declaration. The second is a reminder that formerly "empty" elements, such as `frame`, need the closing slash (`/`) added in order to be well formed.

The attributes for the frame tags used in this example file are discussed in the upcoming section "Adding Content to Your Frames." The four files used to fill each quadrant appear in Listings 6.2–6.5.

Listing 6.2: frame1.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>The first of four quadrants </title>

</head>



<body bgcolor="#FFFFFF" text="#000000">



<p>frame 1, quadrant 1</p>



</body>

</html>
```

Listing 6.3: frame2.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>The second of four quadrants </title>

</head>
```

```
<body bgcolor="#FFFFFF" text="#000000">


<p>frame 2, quadrant 2</p>


</body>
</html>
```

Listing 6.4: frame3.html

```
<!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" xml:lang="en" lang="en">
<head>
<title>The third of four quadrants </title>
</head>


<body bgcolor="#FFFFFF" text="#000000">


<p>frame 3, quadrant 3</p>


</body>
</html>
```

Listing 6.5: frame4.html

```
<!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" xml:lang="en" lang="en">
<head>
<title>The fourth of four quadrants </title>
</head>


<body bgcolor="#FFFFFF" text="#000000">
```

```
<p>frame 4, quadrant 4</p>


</body>

</html>
```

Warning Notice that the DOCTYPE declaration for the frame-content documents in Listings 6.2–6.5 returns to XHTML 1 Transitional. The Frameset DOCTYPE declaration is only used in documents that contain the frameset element.

Using percentages, your row or column values will always add up to 100 percent. But what happens when you want to use a combination of frames that includes one fixed-pixel frame? How do you tell the browser to divide the remainder of the available space? Percentages won't work because they are percentages of the *whole* window. This is where relative spacing comes in.

Handling Relative Spacing

The concept of relative spacing may on the surface appear to be similar to spacing by percentages. However, if you look closely, you'll see it's really very different.
Suppose that with the 200-pixel-wide frame created previously in "Setting Fixed-Pixel-Sized Frames" you want the *remaining* space divided into two frames, one taking up three quarters of the remaining space, the other taking up one quarter of the remaining space. The values for `rows` would be defined as follows:

```
<frameset rows="200,3*,1*">
```

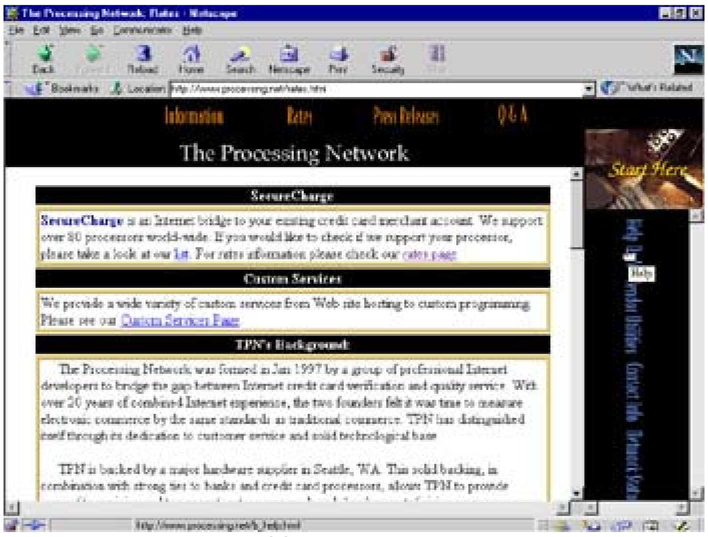
The asterisks tell the browser to make the second frame three times bigger than the third frame. Thus, the second frame is sized *in relation* to the third frame.

If the available width of the browser window is 600 pixels, the first row takes up its predefined width of 200 pixels, the second is given 300 pixels, and the third is 100 pixels. If this page is viewed on a monitor set to 800 × 600 resolution with 760 pixels in available browser-window width, the first row would take up 200 pixels, the second row would be set to 420 pixels (three quarters of 560), and the third would be 140 pixels.

Adding Content to Your Frames

Now that you've allocated space for your frames, it's time to define the content that will go into each of them. The content is derived from individual XHTML files that will be loaded into each frame. Each frame that was created by the rows and columns declared in the `frameset` must contain content, even if that content is an XHTML file that simply has an empty body container.

WWW It is possible to get really creative with frames by making use of the attributes available to you. You can challenge traditional Web design by placing elements, such as navigational tools, in unexpected places. The creative use of frames is now finding its way out of personal pages and onto corporate sites. An interesting example can be seen on The Processing Network's Web site at `http://www.processing.net/info.html` (see Figure 6.3).


The Processing Network: Rates - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.processing.net/rates.html What's Related
Information Rates Press Releases Q & A
Start Here
Help Vendor Matters Contact Info Network Status
Help
http://www.processing.net/h_help.html

The Processing Network

SecureCharge

SecureCharge is an Internet bridge to your existing credit card merchant account. We support over 80 processors world-wide. If you would like to check if we support your processor, please take a look at our list. For rates information please check our rates page.

Custom Services

We provide a wide variety of custom services from Web site hosting to custom programming. Please see our Custom Services Page.

TPN's Background

The Processing Network was formed in Jan 1997 by a group of professional Internet developers to bridge the gap between Internet credit card verification and quality service. With over 20 years of combined Internet experience, the two founders felt it was time to measure electronic commerce by the same standards as traditional commerce. TPN has distinguished itself through its dedication to customer service and solid technological base.

TPN is backed by a major hardware supplier in Seattle, WA. This solid backing, in combination with strong ties to banks and credit card processors, allows TPN to provide

Figure 6.3: The use of frames allows this site's creator to use navigation tools both on the top, in a traditional manner, and on the right, something difficult to do without frames.

The site designer combined several effects here. The first is a fixed-size frame across the top without borders, which holds the site title and four major navigational topics. The frame on the right contains additional navigational tools and is set to scroll because the contents of that frame may be longer than the available space within the browser. (On my 800 × 600 display, it all just barely fits.) The main frame in the middle will also scroll when necessary.

The *frame* Element

The `frame` tag defines the individual frame contents. It has seven primary attributes:

- name
- `src`
- `noresize`
- `scrolling`
- `frameborder`
- `marginwidth`
- `marginheight`

Some of these attributes will be familiar to you from other tags you've learned.

Naming Frames

The `name` attribute is used to assign a name to your frame. Each frame should be named in a way that's meaningful. When links are used to load new HTML files into the various frames, the frame name will target that process appropriately. Names such as main, NavWindow, header, or footer can be envisioned easily.

Frames do have special naming conventions you'll need to remember. Frames that you create must have names that begin with an alphabetical character—that is, any letter from *A* to *Z* in either lowercase or uppercase. Four reserved frame names have special syntax and have been set aside to describe special behaviors: `_blank`, `_self`, `_parent`, and `_top`. Their special properties are described in the upcoming section "Targeting a Frame."

Frame Source

The source (`src`) attribute should be a familiar one to you because you saw it in the section on handling images. It is the source URL of the document to be contained within that frame. Following the concepts of relative path statements, you might use only a filename if the source file resides in the same directory as the file defining the `frameset`, or you might include path information or a fully qualified URL.

No Resizing Allowed

The attribute `noresize` is somewhat unique in that it's a binary attribute. In HTML, the effect of the attribute is turned on simply by its presence (without the need for a value). XHTML doesn't allow that; all attributes must have values in order to be well formed. Therefore, `noresize` can have one value, which is `"noresize"` if that property is desired for the frame. In full, it would be written as `noresize="noresize"`. If the `noresize` attribute is not present, the user is free to resize the frame as desired.

Independent Scrolling

XHTML recognizes that the content of frames will often exceed their allotted viewing area. Just as a browser provides scroll bars for long or wide Web pages, such scroll bars can be provided for individual frames. The `scrolling` attribute has three possible values:

- **Auto** This is the default value. The browser will insert scroll bars when the length or width of the document loaded into the frame exceeds the available viewing space.
- **Yes** When the attribute is set to a value of `"yes"`, the browser is instructed to supply scroll bars, whether or not the document loaded into that frame exceeds the available viewing space.
- **No** The browser is instructed to *not* provide scroll bars. Any content of that frame that exceeds the available viewing space will not be displayed. This value should be used with care, and the display should be tested for acceptable results on a variety of browser window sizes and screen resolutions.

You may be wondering why you would ever want to use `yes` or `no` instead of `auto`. In certain situations, the design of the page can be compromised when scrollbars are present. By setting scroll bars to `no`, you ensure that minor inconsistencies on the display don't put unsightly scrollbars in your frames; however, you do run the risk of some data loss. Similarly, from a design standpoint you may always want scroll bars to appear. It's always a design decision, not a usability decision, to not use `auto`.

Creating Bordered Frames

Figure 6.2 showed frames dividing a browser window into four quadrants. The gray bars you see in the screen capture are the frame borders. By default, the browser includes them. To proactively set the use of a border, set the `frameborder` attribute to a value of `1`. To instruct the browser not to use them, set the value to `0`.

Tip You might recognize frameborder values from the binary logic used in programming languages. A value of 1 represents *true*, and a value of 0 represents *false*. The same concept has been carried over into XHTML conventions. The value of 1 for a frameborder attribute could be said to be true for the instruction to include a border and false (a value of 0) for the instruction to not include a border.

Defining Margins

The attributes `marginwidth` and `marginheight` are used to determine the size of the left and right, and top and bottom margins, respectively. The values for these attributes are measured in pixels and *must* be greater than `1`.

Tip If either attribute is not included within the frame element, the default placement is left up to the browser. XHTML does not provide a set default margin values for browsers to use, so the result varies from browser to browser.

WWW A noted San Francisco Web Designer, Derek Powazek, created a site that combines many of the attributes discussed here in a very unusual way. A story on his site The Fray (`http://www.fray.com/hope/meeting`) is actually "hidden" behind two other frames (See Figure 6.4). To get to the story, the reader resizes the two frames according to the arrows shown.

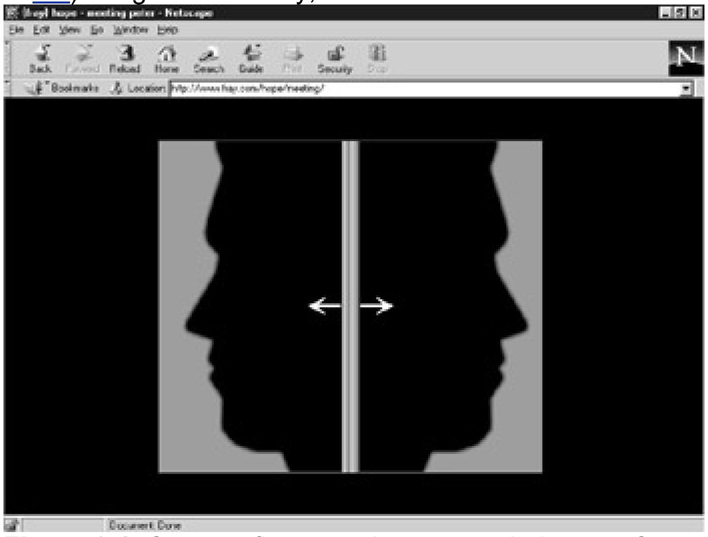

File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://www.fray.com/hope/meeting/
Document: Done

Figure 6.4: Can you figure out how to reach the story?

Is Imitation *Really* the Sincerest Form of Flattery?

One issue that is most difficult for many new Web designers to understand is the application of copyright to XHTML markup. How could simple `body`, `p`, and `ul` tags be copyrighted? Wouldn't other Web designers quickly control all possible variations of tags? Not really.

In practice, copyright attaches to a work—traditionally thought of as drawings and paintings, or books like this one—at the moment of creation. When applied to XHTML and the Web, copyright applies to the Web-page content. When the tags used to create that content become the content itself, as with the design for the page captured in Figure 6.4, a very new and uncharted place in copyright law is entered.

It's become almost a right of passage among Web designers to view the XHTML source code of sites around the Web and to learn from the techniques used to create them. The key has always been the idea of learning by example: You take the skills learned and apply them *in your own way* on your own Web sites.

The extremely unique and creative methods used by Derek in creating the page pictured in Figure 6.4 have unfortunately been copied wholesale—someone simply cut and paste—and used on other sites. It's certainly flattering, but it is also treading very shaky legal ground. Derek was able to resolve his dispute with the other party when they agreed to include a short statement crediting him with the concept.

Avoid situations like this one. Some developers might not have been as accommodating as Derek was or may have even been "flattered" enough to call their intellectual-property attorney.

The bottom line? Adhere to the tradition of learning by example. Take what you've seen and apply it to your work in a new and different way. If you find something you think is "just perfect" for your site, contact the original author and ask for permission to duplicate it. Above all, if you have concerns about whether incorporating ideas into your efforts is proper, consult an attorney.

Targeting New Frame Content

Now that the browser window has been divided into a set of frames with behavior and appearances defined, what happens when you need to create links in one frame that will display new content in another frame? The *target* for the link handles those instructions. A target is an attribute within a link. It defines the name of the frame in which the new content should be loaded. In Listing 6.6, I'll take another look at the `frameset` created in the `frames.html` file in Listing 6.1:

Listing 6.6: frames.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>A set of frames divided into four quadrants</title>

</head>

<frameset rows="50%, 50%" cols="50%, 50%">



<frame src="frame1.html" frameborder=1 name="q1" />

<frame src="frame2.html" frameborder=1 name="q2" />

<frame src="frame3.html" frameborder=1 name="q3" />

<frame src="frame4.html" frameborder=1 name="q4" />

</frameset>
```

```
</html>
```

Now, I'll add a link in the `frame1.html` file to the `frame1a.html` file, as shown in Listing 6.7.

Listing 6.7: frame1.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>The first of four quadrants </title>

</head>

<body bgcolor="#FFFFFF" text="#000000">



<p>frame 1, quadrant 1</p>



<p>This link will insert new content into this frame by loading the

file

  <a href="frame1a.html" target="q1">frame1a.html</a>.



</body>

</html>
```

The new content file that is being linked to is `frame1a.html`. It is shown in Listing 6.8.

Listing 6.8: frame1a.html

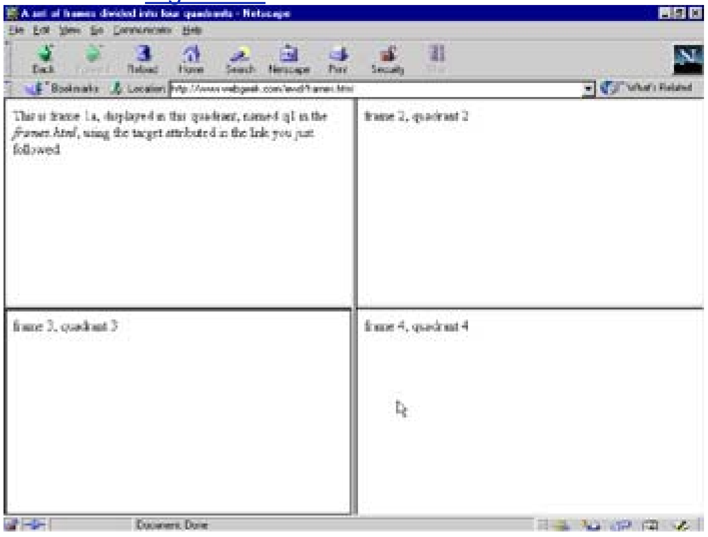
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-Transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>Frame 1a, new content in the 1st quadrant </title>

</head>



<body bgcolor="#FFFFFF" text="#000000">

<p>This is frame 1a, displayed in this quadrant, named q1 in the

  <em>frames.html</em>, using the target attributed in the link
```

you just followed.

</body>

</html>

Notice that after following the link, the only frame with new content is the upper-left quadrant, q1, as shown in Figure 6.5.


A set of frames divided into four quadrants - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.webgeek.com/ewd/frames.html
What's Related
Document: Done

This is frame 1a, displayed in this quadrant, named q1 in the *frames.html*, using the target attributed in the link you just followed

frame 2, quadrant 2

frame 3, quadrant 3

frame 4, quadrant 4

Figure 6.5: The browser screen after `frame1a.html` has been loaded

Reserved Targets

In the earlier section "Naming Frames," four names were introduced as reserved: `_blank`, `_self`, `_parent`, and `_top`. These names are values and have special meaning to the Web browser. They initiate specific behavior when loading new content. See Table 6.1.

Table 6.1: Reserved Frame Names

NAME	ACTION
`_blank`	The linked file loads into a new and unnamed browser window.
`_self`	The document loads into the frame where the file containing the link currently resides.
`_parent`	The browser loads the file into the closest outer or parent frameset from the current frame, if framesets have been nested. The value will default to the behavior of `_self` if no parent frame exists.
`_top`	The document is displayed into the full original browser window, removing any other frame structure.

Tip Remember, framesets are containers. A frame can hold another frameset, like nesting Tupperware bowls. If you have a stack of three, a file being loaded from inner bowl 3 (if _parent is used) would load in parent bowl 2, not into the outermost bowl 1.

A _blank Target

Table 6.1 says that a target value of `_blank` will result in the XHTML file being loaded into a new, unnamed browser window. That means the user's computer system will launch the browser in a second instance to receive the new content and move the focus there from the first instance, where the link was clicked.

You must consider two major issues when using this approach. First, the user may not have recognized that a new browser has been launched. They may try to use their browser's Back button to return to previous pages and may be confused when they find they can't. The new browser window does not inherit the page-access history from the original browser window. Your design should take this into consideration by at a minimum including links back to the original framed content. Better yet, you should avoid opening any new browser windows unless you tell the user that you are doing so and give them an option between opening a new window or loading the page into their current browser window.

Second, by launching a new browser, you're making an imposition on your visitor's computer resources. Those with less powerful computers may run into performance problems—and potentially even system crashes—if their computers are overtaxed by running the additional instance of the browser program. Therefore, don't take the launching of new windows lightly. Consider all other design options before choosing to go forward with launching a new instance of the browser.

A _self Target

The `_self` target value tells the browser to load the new content into the current frame. You can consider this normal link behavior if you think of the frame as a full window. The previous example using a link from `frame1.html` to `frame1a.html` mimics this behavior. Instead of using the target value `q1`, the value `_self` could have replaced it with identical results.

The _parent Value

The `_parent` target value is one that can be handy when you have a layout that is several `frameset` elements deep. However, it does require a thorough understanding of the "family tree" of each `frameset`. Rather than using `_parent`, it's often easiest to use the specific name of the frame you wish to target.

Climbing Out to the _top Target

When the target value of `_top` is used, all `frameset` elements are ignored, and the new content is displayed in the original full browser window. The `_top` value is greatly favored over `_blank` because it doesn't tax the user's system by launching an entire new instance of the browser.

Warning If you don't use the _top value when linking to a Web page outside of your own site, that new site will be enclosed within a frame on your site. Such browser behavior, although the default for a link without a target value, can land you in a pretty murky legal setting. Many a Web-site owner has contended that when another Web developer displays their content in that Web developer's frames, the developer has repurposed it or is intimating that the content is the developer's own by enclosing it in branded frames. As a result, some feel such developers have crossed the line into copyright or trademark infringement. Besides the possible legal implications, a more immediate issue is the degradation of your users' experience of the framed content by shoehorning it into a smaller space than what was intended. All things considered: Use the target value of _top when linking to any other sites.

Design Issues in Frames

Two major design pitfalls arise when using frames on a Web site. The primary challenge is what happens when someone using a browser that can't display or interpret frames visits your site. Such a scenario is more common than you might imagine (see Chapter 20 about alternative-access devices that "suffer" from this problem), so it's important to provide for this segment of your potential audience. I'll describe a solution in detail in the next section.

Note In several of the earliest frames-supporting browsers, primarily Navigator 2, the browser's Back button took the user all the way back out of the framed content, instead of just back one link. Though Navigator 2 is now quite old, a small percentage of visitors may use that version. By providing navigation within your pages, you can circumvent this inconvenience to your visitors.

The second problem tends to vex your users. Consider this: You have a beautiful site, gigabytes of high-quality reference materials or the most popular toys in town just before Christmas. Your visitors want to revisit specific pages, so they bookmark it (add it to their Favorites list, in Internet Explorer lingo). Normally, that lets the visitor get right back to the page the next time they log on. But wait! What happens when your site is framed? Does the bookmark capture the material in your "main" or "content" frame? Does it capture the entire state of the frameset? How would it know which frame(s) are the important ones?

Unfortunately, the browser has no idea what to do with a framed site, so it bookmarks the "shell," or the basic frameset. Next time the site is revisited, the user will see the default presentation, normally back at the "beginning" of their experience on your site—an inevitably frustrating experience.

Providing for Browsers That Can't Display Frames

The concept of frames originated with an implementation by Netscape in its Navigator 2 browser. Browser versions predating that implementation aren't able to display frames, and some less-used current browsers can't handle them, either. How can a Web designer provide for both types of visitors in one site?

Some designers provide a second complete set of Web pages that visitors can access separately. That's a plausible route when the framed layout makes use of nested frames and a significant number of screen divisions. However, the XHTML specification for frames includes a provision for displaying content to nonframes browsers—the `noframes` element.

A common layout using frames provides a navigation menu in one small frame; the XHTML file, perhaps called `nav.html`; and the content itself in a main frame with a filename such as `main.html`. The `frameset` itself is defined in the default file, in this case `index.html`. The markup in `index.html` could appear as simply as the following:

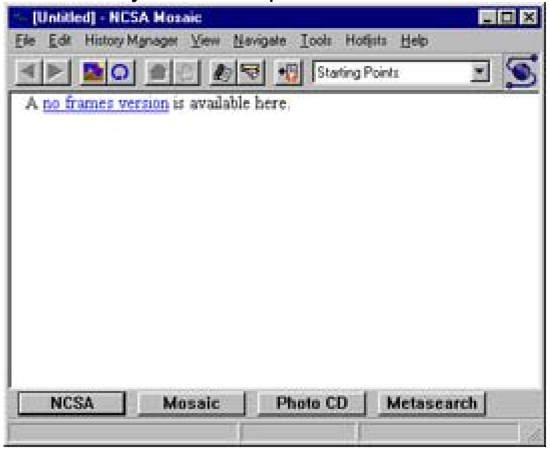
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Frames for Navigation Menus</title>
</head>
<framset cols="30%, 70%">
   <frame name="nav" src="nav.html" />
   <frame name ="content" src="main.html" /
</frameset>
</html>
```

If a user visited this site with a browser that didn't support frames, they'd be greeted by only a blank page. A short addition to the file can take care of this, as shown in Listing 6.9.

Listing 6.9: index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<title>Frames for Navigation Menus</title>

</head>

<framset cols="30%, 70%">

   <frame name="nav" src="nav.html" />

   <frame name ="content" src="main.html" />

</frameset>

<body>

<P>A <a href="main.html">no-frames version</a> is available here.

</body>

</html>
```

The body container presents the user with the single sentence and link shown in Figure 6.6.



A no frames version is available here.

Figure 6.6: `Index.html` as seen in a nonframes browser (NCSA Mosaic 3 for Windows 95)

You'll notice that the link provided within `index.html` (Listing 6.9) is to `main.html` (listing 6.10), the same file used as the original `src` value of the second frame. The `main.html` file is where the `noframes` content will be presented, as shown here in Listing 6.10.

Listing 6.10: main.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Frame 1a, new content in the 1st quadrant </title>

</head>

<body bgcolor="#FFFFFF" text="#000000">

<noframes>

<P>...the navigational content from nav.html here...</p>

</noframes>

<P>...the original content found in the main.html document here...

</body>

</html>
```

Although I haven't yet created the `main.html` file content, you should know that the addition of the `noframes` element to hold the navigational content from the `nav.html` frame (in case the user is unable to use frames) will not affect anything else in the document.

The `noframes` container is simply inserted immediately below the opening `body` tag and before any other XHTML markup. Figure 6.6 displays this short file in a nonframes browser. Looks like a regular XHTML page, doesn't it? That's the idea. The move from the framed content to a nonframes display should be as transparent as possible for the user without a frames-capable browser.

Tip The noframes example used in Listing 6.10 provides a simple transition between framed and unframed content. Designers who choose a more complex framed presentation—such as nested framesets or more than two or three frames in a single frameset—may be required to provide a full second set of XHTML pages for nonframed presentation. At that point, a designer should conduct a cost-benefit analysis to evaluate how worthwhile such a presentation is compared to the cost of providing two complete versions of the site. Depending on the type of site being created, accessibility issues may also come into play—whether as the right thing to do to meet federal standards about Web-site accessibility. For more information on accessible design methods, see Chapter 20.

The Inline Frame

An inline frame—the `iframe` element—can be thought of as an image or other object that is inserted into an XHTML document. The object is presented in the specific space that you specify, and text and other elements can flow around it.

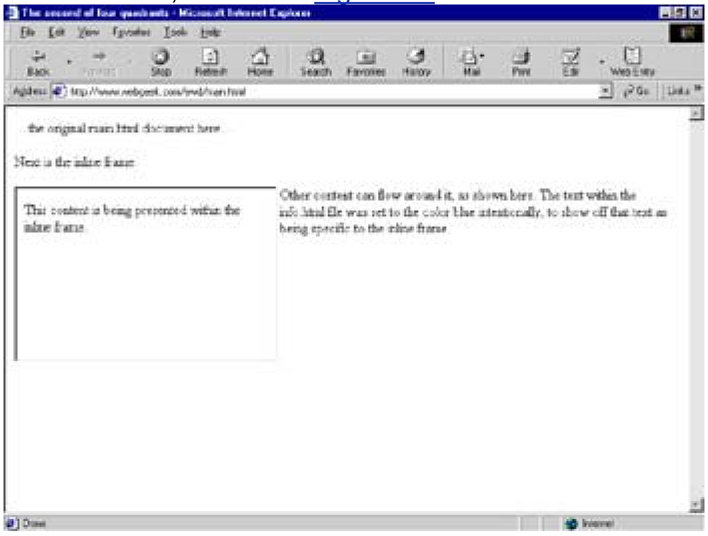
The attributes for the `iframe` element include most of those for the `frame` element. (See the section "The `frame` Element" earlier in this chapter.) The only exception is the `noresize` attribute, which cannot be used in `iframe`. Additional attributes include `width` and `height`, which are used the same way as they are for setting aside space for images. (See the section "Image Dimensions: `width` and `height`" in Chapter 4.) You can use `width` and `height` values instead of `noresize`; the object can't be modified later.

You can name the inline frame as well as determine its border, margin, and scrolling properties. As with `frame`, the information to be displayed in the inline frame is assigned in the `src` attribute. It can be an XHTML file or any other file that would ordinarily be viewed through a link. Here's a short example in Listing 6.11.

Listing 6.11: Displaying Text in an Inline Frame

```
<iframe src="info.html" width="300" height="200" frameborder="1"

  alignt="right">

<P>Your browser apparently does not support frames. You may

  see the content intended to be displayed here by viewing

  <a href="info.html">this link</a>.

</iframe>
```

The XHTML markup in Listing 6.11 displays the contents of `info.html`, shown in Listing 6.12, in an inline frame, as seen in Figure 6.7.



the original main html document here

Here is the inline frame

This content is being presented within the inline frame.

Other content can flow around it, as shown here. The text within the info.html file was set to the color blue intentionally, to show off that text as being specific to the inline frame

Figure 6.7: An inline frame as seen in IE 5

Listing 6.12: info.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Using Inline Frames</title>

</head>

<html>

<body bgcolor="#FFFFFF" text="#0000FF">

<p>This content is being presented within the inline frame.</p>

</body>

</html>
```

The newly modified `main.html` file is shown in Listing 6.13. It was modified to accept an inline frame; `info.html` is inserted into the frame.

Listing 6.13: main.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Main Page</title>

</head>

<html>

<body bgcolor="#FFFFFF">



<p>...the original main.html document here...</p>



<p>Next is the inline frame.</p>



<iframe src="info.html" width="300" height="200" frameborder="1"

  align="LEFT">

<p>Your browser apparently does not support frames. You may

  see the content intended to be displayed here by viewing

  <a href="info.html">this link</a>.</p>

</iframe>
```

```
<p>Other content can flow around it, as shown here. The text within

  the info.html file was set to the color blue intentionally,

  to show off that text as being

  specific to the inline frame.</p>



</body>

</html>
```

Tip At press time, the only major browser to support the iframe element was IE 4 or later. As other browsers produce updated versions to support many of the new options included in XHTML 1, additional support for inline frames can be expected.

Summary

Frames provide the Web designer with the ability to set off portions of the available browser window into independent smaller windows—frames. Frame support began to be available in IE 3 and Navigator 3 and higher.
Frames can be drawn by using specific pixel values for dimensions, by percentages, or by relative spacing. The properties for margins, borders, scrolling, and resizing can all be set within the `frames` tag. The `noframes` element is included to allow for browsers that cannot display framed content.
The `iframe` element allows frames to be drawn anywhere in the body of an XHTML document. Attributes of `iframe` are similar to those of `frame`, with the addition of `width` and `height` and the exclusion of `noresize`. Other XHTML content can be set to flow around inline frames that are smaller than the available browser width or height. Browsers that don't support inline frames will display the content within the `iframe` container, as if the tag had never appeared in the document.

Two design pitfalls can arise when working with frames: the necessity of providing content for browsers that don't support them and how to accommodate the user's desire to bookmark valuable information on your site, thus encouraging return visits.

Chapter 7: Style Sheets

Overview

When people gained interest in the Web and began developing their own Web pages, many became frustrated at what they saw as the lack of design control in HTML. The style sheet was developed as a means for giving the designer more precise control over design elements.

This chapter covers the following topics:

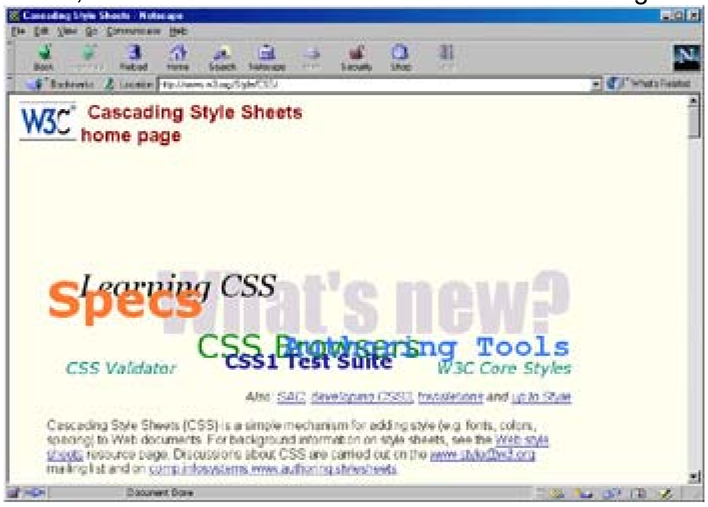
- Selecting a style language
- The application of style
- Document specific styles
- Inline styles
- Referencing an external style sheet
- Online resources for style sheets

Designing with Precision

Word-processor users, desktop publishers, and employees in traditional print media are all used to controlling the entire presentation of their work. Judgements can be made for the proper placement of elements within the page because the "canvas," as it were, has finite borders—the edges of the paper.

In traditional print media, for instance, a graphic could be placed exactly 1.1875 inches from the left margin and exactly 2.0 inches from the top margin. Transferred to the computer screen, that precise placement often became a designer's nightmare. How can they move an image up by a half a line in order to have a more pleasing flow of text around it? Could the spacing between lines be increased in order to accommodate a few lines that had superscript footnote markers? On paper, that is easily done. Within a browser, no mechanisms existed for nudging objects about the screen in very small increments. That absolute control is sometimes called *pixel-level control* over presentation.
Quite a few "solutions" were created for dealing with traditional print concerns such as white space, setting margins, and many more presentational elements. But the solutions weren't reliable. Most either were browser specific or used HTML tags in ways that weren't intended and therefore created inconsistencies or flat-out failures in display (see Chapter 2).
Style sheets have been developed as a means to define these visual instructions without resorting to misuse of HTML tags or solutions that can be implemented only by one browser. A style sheet can be a set of instructions that apply to very specific portions of your document—to highlight *this text* in *that color*, for example. Or a style sheet can be universal guidelines for document presentation, controlling font choices, colors, margins, heading styles, and almost any other visual detail you can imagine. Generally, a style sheet is a document or document fragment that defines rules for the stylistic presentation of content.

Style sheets can be developed in any *style-sheet language*—they are not produced in XHTML. Each style-sheet language has its own syntax and rules, much like programming languages and XHTML have their own rule sets. As of now, the only language in use with XHTML across multiple browsers is known as *Cascading Style Sheets*, commonly abbreviated as CSS. (It is certainly possible, though, that extensions or other such languages could be developed in the near future.) Cascading describes a behavior of precedence and order, as discussed in detail later in this chapter. CSS has been implemented in most of today's visual browsers, including Netscape and Internet Explorer.
One of the most visually identifiable features of Cascading Style Sheets is the appearance of layered text, as seen on the W3C's Web-site section dealing with style sheets, shown in Figure 7.1.



Also: SAC, developing CSS3, translations and up to Style

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g. fonts, colors, spacing) to Web documents. For background information on style sheets, see the Web style sheets resource page. Discussions about CSS are carried out on the www-style@w3.org mailing list and on comp.infosystems.www.authoring.stylesheets

Figure 7.1: A layered look is achieved through margin, heading, and font manipulation.
The terms *style sheets, style-sheet language*, and *Cascading Style Sheets* are used nearly interchangeably in many Web-development circles, which can lead to some confusion. It is important to understand the distinctions.
The popularity of style sheets may seem like much hoopla over nothing. If you want some text to be blue, with style sheets you need to declare a style to make it blue rather than the "old-fashioned" way of using a `font` element. So where's the improvement?

The Origin of Style Sheets

Microsoft's Internet Explorer 3 for Windows 95/NT was the first browser to provide limited support for style sheets. Style sheets actually began as the browser-specific solution of Cascading Style Sheets. The concept was forwarded to the W3C, and style sheets as a generic entity made their way into HTML 4. With XHTML 1's return to structured document design, style sheets are becoming more and more important in the journey toward XML.

As noted, style sheets can allow Web designers to exert much visual control over their Web pages—as much control as they can over documents produced with tools like Microsoft Publisher or Adobe Acrobat. The truly creative among the Web-designer population are rejoicing! Even those who prefer rather stark, text-based pages can benefit from style sheets. Regardless of design philosophy, style sheets allow all of the visual issues, such as font selection, colors, and emphasis treatments, to be defined in one central location so they can be referenced again and again from hundreds of different documents, saving the designer significant amounts of development time.

Selecting a Style-Sheet Language

When style sheets were integrated into HTML, the W3C didn't prescribe any specific style-sheet language. Instead, they anticipated that multiple languages would evolve for different purposes, and most would likely come from the software firms that produce browsers, as the browser must be programmed to understand each new language. As it turned out, a W3C offering known as Cascading Style Sheets (CSS) was the only offering to come into public view.

Extensible Style for XML

If you already know a bit about XML, you've also probably heard about XSL, the Extensible Stylesheet Language used hand in hand with XML. XSL is considered by many to be a breakthrough in style languages in that it's written in the very syntax that it intends to style; that is, XSL is written in XML syntax.

XSL has two major parts: XSL Formatting Objects (XSL:FO) and XSLT, or XSL Transformations.

XSLT is a language for transforming XML documents, and most Web developers first come in contact with XSL through this language. XSLT has the power to transform XML structures into other XML structures, including those found in XHTML—which then allows them to be presented in today's browsers using the techniques found in this book.
XSL Formatting Objects begins with simple formatting, the process of turning the results of an XSL Transformation into something that's useful for an end-reader or listener. The process of formatting results in the creation of an *area tree*, which is defined in the XSL Working Draft as "an ordered tree containing geometric information for the placement of every glyph, shape, and image in the document, together with information embodying spacing constraints and other rendering information; this information is referred to under the rubric of traits, which are to areas what properties are to formatting objects and attributes are to XML elements."
I'll take a very brief look at XSL in Chapter 22 as I make use of a custom DTD module in XHTML.

To get started using CSS with XHTML 1, the Web designer needs to declare in the document's `head` element, using a `meta` element, which style-sheet language they'll be using. (For information about `meta`-element formatting, see Chapter 3.) The following code is a declaration for the use of CSS:

```
<meta name="Content-Style-Type" content="text/css" />
```

Warning This meta element is equivalent to the HTTP header: Content-Style-Type: text/css. HTTP headers are bits of information that are passed back and forth between the server and the browser. The system was developed in such a way that the data is case sensitive. Accordingly, be careful to maintain the exact capitalization and punctuation as shown in the code before this Warning.

The Application of Style

Now that you've chosen a style-sheet language and included a `meta` element defining that choice, it's time to start applying styles to your content. You have several ways of doing so. The two most flexible and time-saving are as follows:

- Insert a style element container into the document's head element.
- Create a separate document, the style sheet, that defines the styles to be globally applied to all files that reference it.

Within the `style` element are the specific instructions for handling the appearance of content within the document. The `style` element has one attribute, `type`. The value defines what type of style you want to apply—Cascading Style Sheets—so the familiar `text/css` is entered here, as follows:

```
<style type="text/css">
```

The instructions that follow are all enclosed in a comment so that older browsers that don't support style sheets and the `style` comment don't misinterpret them. An XHTML comment begins with a left-angle bracket, an exclamation point, and two dashes. It ends with two dashes and a right-angle bracket. Any amount of text may be enclosed in a comment, but XHTML tags may not. At least a single space must be left after the comment opens and before it closes:

```
<!-- This is a comment. -->
```

What Is a Style?

A style can be as simple as rendering some text in **boldface** or in *italic*. Or it can grow very complex with the adoption of print-based ideas such as line heights (double-spacing, etc.) font weights, leading, kerning, and other practices common to desktop publishing.

In a Web document, you might decide that you want to help your headings stand out a bit more by having all of them use a red font color. Without style sheets, you'd need to make that instruction with each heading tag in your document. With style sheets, that only needs to be declared once.
Style syntax is different than traditional XHTML. Instead of tags or elements, the instructions are called *rules*. The basic syntax for a rule includes a *selector* and a *declaration*. A sample follows:

```
H1 { color:red }
```

The selector is the HTML element that is being operated on, in this case, the `H1` element. The declaration is what the browser is supposed to do with it. The declaration will always be enclosed in braces, as shown.
Inside the declaration, you have a *property* and a *value*, which are separated by a colon. A property is the style sheet version of an XHTML attribute. A value carries the same meaning here. The `color` (property) of the `H1` element (selector) should be `red` (value).

You can save yourself some typing work by grouping selectors into a single rule. Let's say you wanted to have all of your headings rendered in red. You could type out six rules in a row, like so:

```
H1 { color:red }
H2 { color:red }
H3 { color:red }
H4 { color:red }
H5 { color:red }
H6 { color:red }
...
```

Or instead, you can group them like this:

```
H1, H2, H3, H4, H5, H6 { color: red }
```

This rule will apply red to all instances of headings 1 through 6.

Tip For clarity, I've used color names rather than hexadecimal notation for the rules in the examples. It is still best to use hex for broader browser support, especially as more browsers that are not enabled for color-name support adopt CSS support. See Chapter 3 for more information on colors and color names.

Remember the old "structural markup versus display markup" argument for emphasizing and strongly emphasizing content? (See Chapter 4.) Style sheets provide the Web designer with the tools to change the rendering of emphasis rather than relying on the vagaries of a specific browser's rendering of them.

Suppose you want to emphasize your text by using a purple-colored Courier font type. The rule would be written as follows:

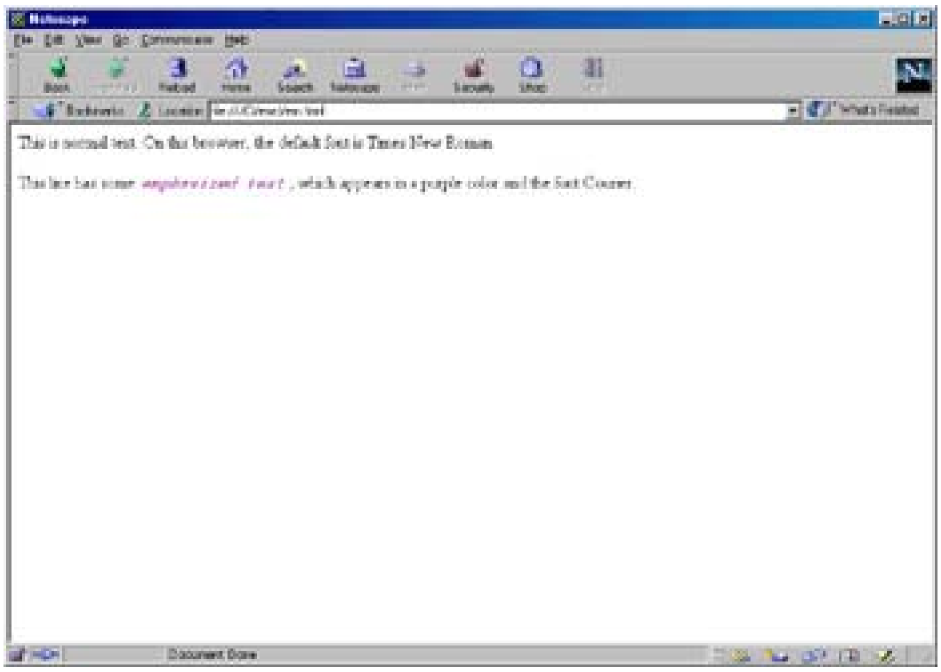
```
EM { color:purple; font-family:Courier }
```

The file `em.html` (Listing 7.1 and on the CD-ROM) is the complete XHTML source for applying this style, and Figure 7.2 shows how it would appear in Netscape 4.76.

Listing 7.1: em.html

```
<!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" xml:lang="en" lang="en">

<head>

<style type="text/css">

<!-- EM { color:purple; font-family:Courier; } -->

</style>

</head>

<body bgcolor="#FFFFFF" text="#000000">



<p>This is normal text. On this browser, the default font is Times

  New Roman.</p>



<p>This line has some <em>emphasized text</em>, which appears in a

  purple color and the font Courier.</p>



</body>

</html>
```


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location What's Related
Document Done

This is normal text. On this browser, the default font is Times New Roman.

This line has some *emphasized text*, which appears in a purple color and the font Courier.

Figure 7.2: Text emphasized with a color and font type change

In Figure 7.2, the emphasized section is also italicized because the `EM` tag traditionally causes text to appear in italic. In the style sheet, I added a color and a font change but did not take away the traditional change to italic. Unless you contradict the traditional rendering of a tag, the browser will apply that rendering in addition to the styles you've outlined. The rendering results from the concept called the *cascade*. You'll learn more about the cascade later in this chapter.

Some Web designers find rules easier to read if only one `property:value` pair is written on each line. That sort of coding would look like so:

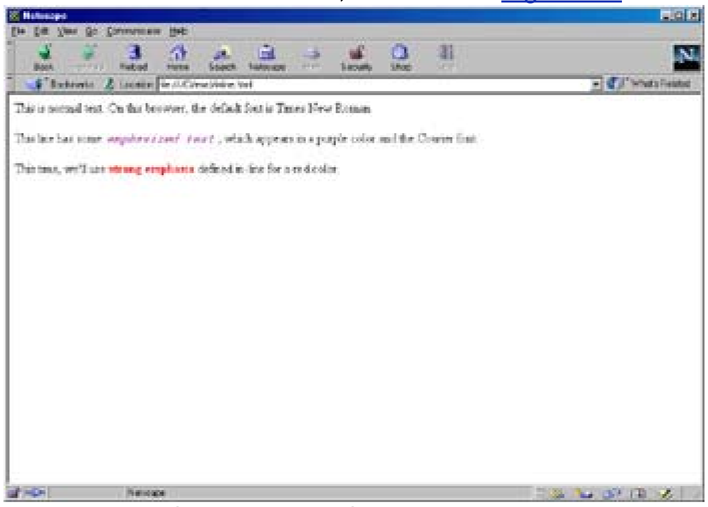
```
EM { color:purple
      font-family:Courier }
```

As long as all your rules are contained within the appropriate comment markers, it's up to you how to format them.

Inline Style Applications

Styles can be applied not only globally to a document but also to specific instances of an element. This is known as *inline* style application.

Taking the example shown in Figure 7.2 one step further, suppose you wanted to use just one instance of strong emphasis (the `strong` tag in HTML) to be red as well as the traditional bold rendering, but you didn't define that ahead of time. You can do so by adding an additional `property:value` pair where the element occurs, as shown in Figure 7.3 and in the markup in Listing 7.2 of `inline.html`.



This is normal text. On this browser, the default font is Times New Roman.

This line has some `emphasized text`, which appears in a purple color and the Courier font.

This time, we'll use **strong emphasis** defined in-line for a red color.

Figure 7.3: A font color modification made using inline styles

Listing 7.2: inline.html

```
<!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" xml:lang="en" lang="en">

<head>

<style type="text/css">

<!-- EM { color:purple; font-family:Courier } -->

</style>

</head>



<body bgcolor="#FFFFFF" text="#000000">



<p>This is normal text. On this browser, the default font is Times

  New Roman.</p>
```

```
<p>This line has some <em>emphasized text</em>, which appears in a
  purple color and the Courier font.</p>

<p>This time, we'll use <strong style="color:red">strong
  emphasis</strong> defined in-line for a red color.</p>

</body>
</html>
```

In this example it would be easier not to use the style-sheet treatment. But, because non-style-sheet treatments are being deprecated, it's important that you learn the style-sheet way to handle the situation.

Defining a Style Sheet

A style sheet, as opposed to an inline application of styles, is a master document that defines the properties to be used in any individual document that references it. To reference it from your XHTML documents, a new element is used inside the `head` element, as follows:

```
<link rel="stylesheet" href="mystyle.css" />
```

The `link` element links the style sheet to the current document. The `rel` attribute defines the relationship between the current document and the document that it's being linked to—hence the value `stylesheet`. In this instance then, the link concept applies to the relationship rather than being a reference to a hyperlink. The two shouldn't be confused. The `href` attribute is familiar to you from hyperlinks, and its value is the URL of the external style sheet.

Tip You know that HTML and XHTML files are conventionally named with the file extension .HTM *or* .HTML. This standard practice allows browser and authoring-tool software—as well as humans—to recognize documents as HTML files. With style sheets, the extension .CSS has become the traditional extension, though it is not required by the style-sheet specification or by the Cascading Style Sheet language. However, it does provide a very helpful visual clue to the file's contents.

The file `styles.html` shown here in Listing 7.3 (and on the CD-ROM) links to the external style sheet `mystyle.css`.

Listing 7.3: styles.html.

```
<!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" xml:lang="en" lang="en">
<head>
<title>External Stylesheet</title>
<link rel="stylesheet" href="mystyle.css" />
</head>

<body bgcolor="#FFFFFF">
```

```
<h1>External Style Sheets</h1>


<p>This file was created using an external style sheet named
  <em>mystyle.css</em>.
The style sheet determined the color and justification of
  the heading, the font face,
the alignment of this paragraph, and the application of
  color to the emphasis tag.</p>


</body>
</html>
```

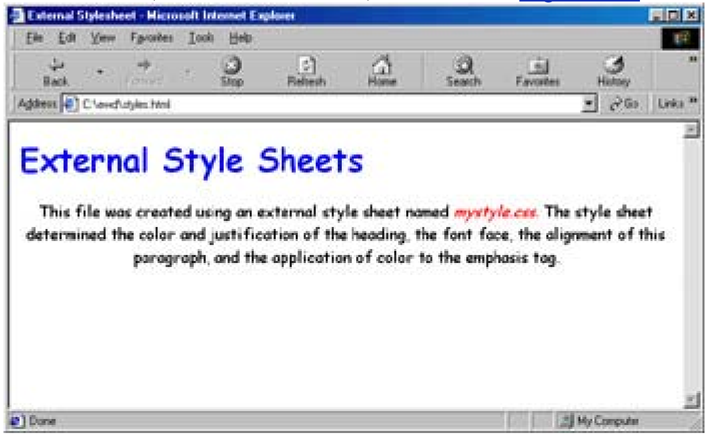
The file `mystyle.css`, as shown in Listing 7.4 (and on the CD-ROM), contains four rules that will be applied to the XHTML document:

Listing 7.4: mystyle.css

```
BODY { font-family: "Comic Sans MS" }
H1 { color: blue; align:center }
P { text-align:center }
EM { color:red }
```

- Level 1 headings that are blue and centered
- A font face of Comic Sans MS (a fun font that is part of the Windows 95 standard set of fonts)
- Paragraph text that is center justified
- The addition of red coloring to text contained within the `EM` tag.

The result, as seen in IE 5.5, is shown in Figure 7.4.


External Stylesheet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Address C:\...\styles.html
Go Links
Done
My Computer

External Style Sheets

This file was created using an external style sheet named *mystyle.css*. The style sheet determined the color and justification of the heading, the font face, the alignment of this paragraph, and the application of color to the emphasis tag.

Figure 7.4: The file `styles.html` rendered with a link to an external style sheet

The Cascading Concept

As noted earlier, it's common among Web designers to use the terms *style sheets* and *Cascading Style Sheets* interchangeably, especially because Cascading Style Sheets is the only readily available style-sheet language. You'll understand from the following discussion, though, why it is important to know the distinction.

Style sheets, as a generic entity, is an adjunct to a markup language that provides for stylistic matters to be defined by an external document, in the `head` of an individual XHTML document, or inline as needed. Cascading Style Sheets is a language—or perhaps better defined as a syntax—for defining those stylistic matters. However, these are only the options available to the designer. Users can also create a style sheet of their own, and browsers have default style sheets.

Cascading pertains to the order of importance in style rules. What takes priority and when? If all three types of style sheets are present—designer defined, user defined, and browser default—then the style rules are collected from each, *cascading* from the designer's style sheet, to the user's, and then to the browser default. When a contradiction is found in the rules, then the cascading mechanism must make a choice. If the contradictory rules are of the same importance level, the mechanism will choose the designer's rule before the user's rule and the user's rule before the browser's rule.

It is possible for the designer, user, or browser programmer to mark one of their rules as more important than others, using the CSS keyword `!important`. For instance, many users find it much easier to read white text on a black background than to read black text on a white background. They may choose to write a style sheet that included the following rule:

```
body {color:white !important;
  background:black !important;}
```

Tip When applying color rules to the body element, it's important to remember that background is the rule designers are used to working with and is equivalent to the XHTML bgcolor attribute. The CSS color property, however, applies to the text within the body. In the sample code shown here, then, the background is black, while the text, unless modified by another rule, will be white.

Inheritance

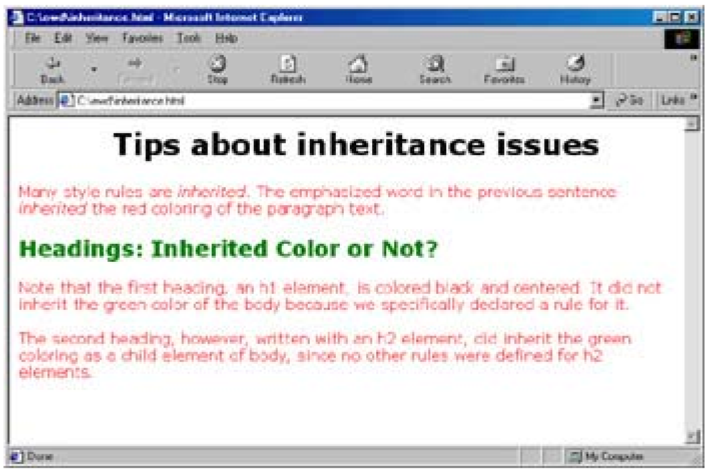
Inheritance, unlike the cascade, refers to properties handed down from *parent* element to *child* element. Consider the basic tree structure. The `body` element is at the top of the tree, acting as the parent element. All elements that branch off of (are contained within) the `body` element are *children* of the `body` element. Note that child elements can have their own children.

The file `inheritance.html` in Listing 7.5 defines several styles in the `head` element. Figure 7.5 shows the rendering.

Listing 7.5: inheritance.html

```
<!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" xml:lang="en" lang="en">

<head>

<style>

<!-- body { color:green }

    h1 { color:black; align:center }

    p {color:red}

    -->

</style>

</head>

<body>

<h1>Tips about inheritance issues</h1>
```

```
<p>Many style rules are <em>inherited</em>. The emphasized

  word in the previous sentence <em>inherited</em> the red

  coloring of the paragraph text.</p>


<h2>Headings: Inherited Color or Not?</h2>

<p>Note that the first heading, an h1 element, is colored

  black and centered. It did not inherit the green color of

  the body because we specifically declared a rule for it.</p>

<p>The second heading, however, written with an h2 element,

  did inherit the green coloring as a child element of

  body, since no other rules were defined for h2

  elements.</p>


</body>

</html>
```


C:\ewd\inheritance.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Address C:\ewd\inheritance.html
Go Links
Done
My Computer

Tips about inheritance issues

Many style rules are *inherited*. The emphasized word in the previous sentence *inherited* the red coloring of the paragraph text.

Headings: Inherited Color or Not?

Note that the first heading, an h1 element, is colored black and centered. It did not inherit the green color of the body because we specifically declared a rule for it.

The second heading, however, written with an h2 element, did inherit the green coloring as a child element of body, since no other rules were defined for h2 elements.

Figure 7.5: The `em` element inherits the blue text from the style defined for the `p` element, and the `h2` element inherits the green color set for the `body` element.

Style Properties and Values

WWW You may choose from a huge array of individual style rules and properties for Cascading Style Sheets—far more rules and properties than would be possible to cover in this chapter. In this section, you'll learn about styles that have been put into use on the Web, using XHTML elements, and a few additions that Web designers have been yearning for. The full text of both Cascading Style Sheets recommendations (Level 1 and Level 2) are available online at the W3C Web site, at `http://www.w3.org/TR/REC-CSS1/` and `http://www.w3.org/TR/REC-CSS2/`, respectively.

Styles for Fonts

Cascading Style Sheets provide five style properties for fonts: `font-family`, `font-style`, `font-size`, `font-weight`, and `font-variant`.

Font Family

A font family is a group of fonts derived from the same parent font. The font family *Arial* has several members; Arial Narrow and Arial Black are two of them. Each looks much like the others, as children from the same two parents look alike. When you specify a font family in CSS, you want to use the family group name (in this case, Arial) rather than a specific family member name.

When setting a font-family choice in a style rule, listing several similar font families will increase your chances of the text being rendered as you intended if your first-choice font isn't installed on the visitor's system. Using the Arial example, a rule with several font-family choices could look like the following:

{ font-family: Arial, Helvetica, AvantGarde, "Century Gothic",

sans-serif }

The last font family is the generic font family of `sans-serif`, loosely translated as "without decoration." Serifs are the small flourishes on the tips and edges of letters, such as the foot on each leg of an *n* or *m* and the small vertical line at the top of a capital *S*. A sans-serif font doesn't have those decorations. Should a visitor to the page using this style rule not have any of the four mentioned fonts installed, the inclusion of `sans-serif` will tell the visitor's computer to use an available sans-serif font rather than just the default font family, which might be serif or some other generic type.

Warning If a font family name has spaces in it, as does the font Century Gothic, the name must be enclosed in quotation marks. Otherwise, the browser will condense what it sees as extraneous white space and look for a font called CenturyGothic, which it won't be able to locate.

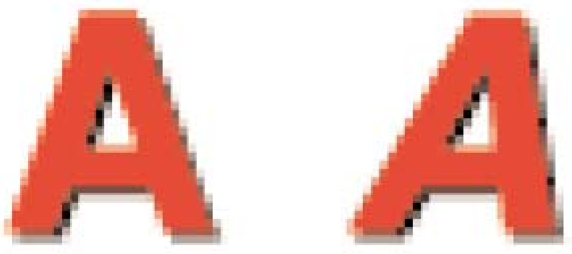
Five generic font families are recognized by CSS, as shown in Table 7.1.

Table 7.1: The five generic font families supported by CSS

GENERIC FAMILY	FONT EXAMPLES
serif	Times, Courier, Garamond, Palatino
sans-serif	Arial, Helvetica, Folio, Century Gothic
cursive	ShellyAllegro, Vivaldi, Nuptial
monospace	Courier, Lucida Console, QuickType Mono
fantasy	Western, Keyboard, Crayon

Font Style

Three basic font-face variations exist: normal, italic, and oblique. (Boldface, by the way, is not a font style; it's a font weight.) A normal font style is sometimes referred to as a Roman style. The type you are reading now is a normal font styling. You are probably familiar with italic, but you might not be familiar with oblique. An oblique style is a normal style, slanted to the right. The two *A*s, shown here, illustrate the difference between normal and oblique.



Font Size

You're probably familiar with the print-world concept of font sizes if you've ever used a word processor. A typical default might be set to 10- or 12-point type. Using the font-size property, you can change the font size from the default size.

Tip Your Web browser also uses a default font-size setting. Take a look now to see what it is. In Navigator 4.x, select Edit → Preferences → Appearances → Fonts. Other browsers will have different menu steps; the option will generally be available through a Preferences or Options menu.

You can describe font sizes three common ways: absolute size, relative size, and percentages. Each of these descriptive methods depends on the browser for interpretation. Not every browser will render the description in exactly the same manner as the others.

Warning The possible values outlined in the next four sections should be thought of as a guide, understanding that values moving toward the extremes have fewer chances of being interpreted as you intended.

Absolute Size

Absolute sizing uses the keywords small, medium, and large. At either extreme, nomenclature that may be familiar to you from clothing sizes is used: xx-small, x-small, x-large, and xx-large. But how large is a large and how small is a small?

Each browser has its internal table of font point sizes that correspond to these absolute sizes. The suggested difference between levels is 1.5 times the original. So, a medium is 1.5 times the size of a small. A large is 1.5 times the size of a medium and, as you can deduce, 3.0 times the size of a small. So, if the medium font size were 12 point, then large would be 18 point. An xx-small would be down near 5 point, which would be awfully hard to read on a computer monitor!

Relative Size

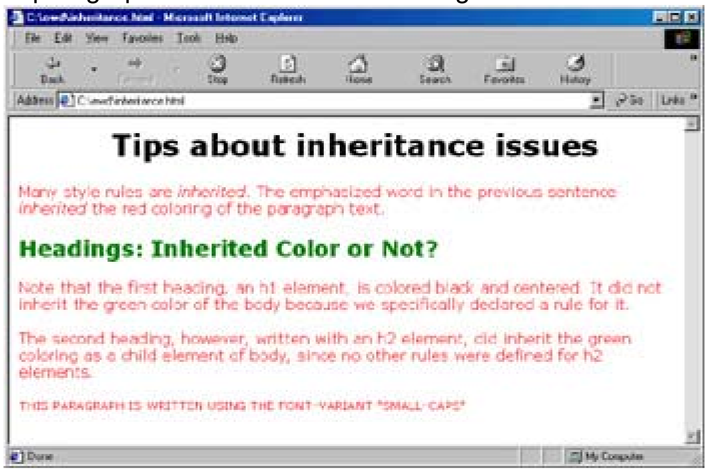
A relative size has only two possible values, larger and smaller. Those are both set *relative* to the current font size. If the suggested incrementing value of 1.5 times the original is kept, then an existing font size of 24 point, when styled to a property of `larger`, would become a size of 36 point.

Percentage

This method is probably the easiest for you to visualize. If you want a font to be changed to be half as big as the current size, you'd increase it by 150 percent. The syntax used is `{ font size:150% }`.

Font Variant

A font variant can be considered a further extension of font styles. The one currently supported by CSS is called `small-caps`. As you might imagine, in `small-caps` the text is rendered in capital letters smaller than the height normal for caps in that font, as seen in Figure 7.6 (applied to an added third paragraph under the second heading in the `inheritance.html` file).



Tips about inheritance issues

Many style rules are *inherited*. The emphasized word in the previous sentence *inherited* the red coloring of the paragraph text.

Headings: Inherited Color or Not?

Note that the first heading, an h1 element, is colored black and centered. It did not inherit the green color of the body because we specifically declared a rule for it.

The second heading, however, written with an h2 element, did inherit the green coloring as a child element of body, since no other rules were defined for h2 elements.

THIS PARAGRAPH IS WRITTEN USING THE FONT-VARIANT "SMALL-CAPS"

Figure 7.6: The style `font-variant` applied with `small-caps`

Font Weights

The font weight relates to how dark or light a font would appear. For the most part, font weight applies best to the print world; however, the **bold** weight is one exception.

Text Styles

Text styles are modifications made to the text as a whole, rather than to individual characters, as is done with font styles. Text styles you are likely to be familiar with include *justification*—the aligning of each line of text to a given margin—and indenting. Table 7.2 lists four common text styles that can be manipulated through the use of style sheets.

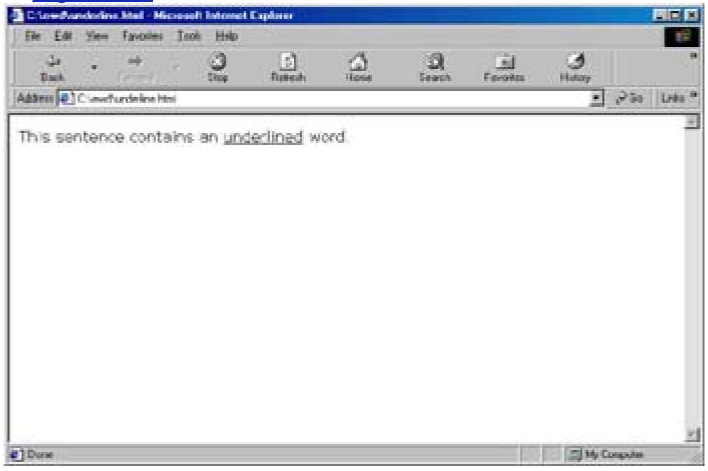
Table 7.2: Four common text styles

PROPERTY	EXAMPLE VALUES/EFFECTS
`text-decoration`	underline, strikethrough
`vertical-align`	baseline, top, middle
`text-indent`	indentation by specified unit
`text-align`	left, right, center, justify

Text Decoration

The text-decoration property in style sheets is now the preferred method of handling the underlining and strikethrough. So instead of using the underline `u` element, as you saw in Chapter 4, you use the `span` element along with the inline declaration of style. For example, without style sheets an underlined word would be marked up like this:

This sentence contains an <u>underlined</u> word.
Using CSS, the `span` element spans the block of text you want to apply a style to, as seen here and in Figure 7.7:



This sentence contains an underlined word.

Figure 7.7: An underline applied using the **`span`** element and inline styles

```
<p>This sentence contains an <span style=
"text-decoration:underline">underlined</span> word. </p>
```

This implementation necessitates a bit more typing for you when applying styles to single words or phrases, as was done here. However, the W3C's decision to deprecate the underline and strikethrough containers keeps style elements consistently within the style-sheet realm instead of within XHTML.

Vertical Alignment

Vertical alignment is used to align text in relationship to other objects, such as images. Should the text align with the top of an image and flow down the side? Or should the text begin in the vertical middle of the image or even aligned with the bottom? In XHTML, these issues were defined with the `align` attribute within the `img` (or other object) element.

Indenting Text

Among all the tricks and work-arounds that Web designers have developed over the past few years, almost none have been developed more than for finding a technically valid and visually acceptable form of indenting. Many designers have accepted the fact that the Web doesn't make use of indenting, but others aren't willing to give indents up yet.

The text-indent property takes a value that is a specific number of *em spaces*. An em is the height of the current font. If you want to indent three of those units measured horizontally, the style would be expressed within the rule as `{ text-indent:3em }`.

Text Alignment

Text alignment is one of the most common text-positioning properties already in use on the Web. You'll very frequently encounter text or headings that have been aligned to the center or to one of the margins. An alignment method that many designers have been waiting for is `justify`. This value will justify text to both margins rather than leaving the typical ragged-right margin seen in Web documents. The `justify` alignment method has been supported by Netscape and IE version 4 and later browsers, as well as with Opera version 3.5 and later.

Style-Sheet Resources Online

WWW The Web has several very thorough tutorials and reference works regarding style sheets:

- Level 1 and 2 W3C recommendations on Cascading Style Sheets can be found at `http://www.w3.org/TR/REC-CSS1` and `http://www.w3.org/TR/REC-CSS2`, respectively.
- CNet's `Builder.com` area has a tutorial, a consolidated property reference table, and even a helpful tool called the Style-o-Matic at `http://home.cnet.com/webbuilding/0-7258.html?tag=st.bl.3880.dir.7258`.
- WebReview.com has compiled a comprehensive Style Sheet Reference Guide with articles, tables and more. It's located at `http://www.webreview.com/style/index.shtml`.

Tip The topic of CSS has filled a number of books all by itself. My personal favorite is *Cascading Style Sheets: The Definitive Guide*, by Eric A. Meyer (O'Reilly & Associates, May 2000).

Summary

Style sheets are becoming increasingly important to today's Web designer. XHTML moves back to a structured-document approach to authoring where the style sheet provides the visual treatment and presentation. Style sheets provide the Web designer with a means of defining visual stylistic issues in a single location, either in the head of an XHTML document or in an external style sheet. The concept is flexible enough that the application of styles inline is possible for single-instance usage.

A style sheet is a method of handling instructions and must be defined in a style-sheet language. Cascading Style Sheets is the most prevalent style-sheet language in use at this writing, though XSL is gaining fast in the world of XML vocabularies.

Chapter 8: Communicating through Forms

Overview

Communicating with the owners of a Web site has become an integral part of the World Wide Web experience. It occurs in dozens of formats—from a simple page that requests visitor feedback to complex order forms or even Web-based bulletin-board messaging systems. The possibilities are limited only by your imagination, your visitor's interest in the product or service, and their willingness to spend the time required to complete the form.

This chapter covers the following topics:

- The `form` container
- Ten commonly used input controls
- Alignment and layout within the form
- Sending the data directly to e-mail
- Processing the response using CGI scripts

The Form in Action

The best forms not only perform the function of collecting data but also retain the look and feel of the rest of the site. Members of Biztravel.com, an online travel-reservations service located at `http://www.biztravel.com`, use the form shown in Figure 8.1. In a very small space, the Express Flight Planner form collects the following information:

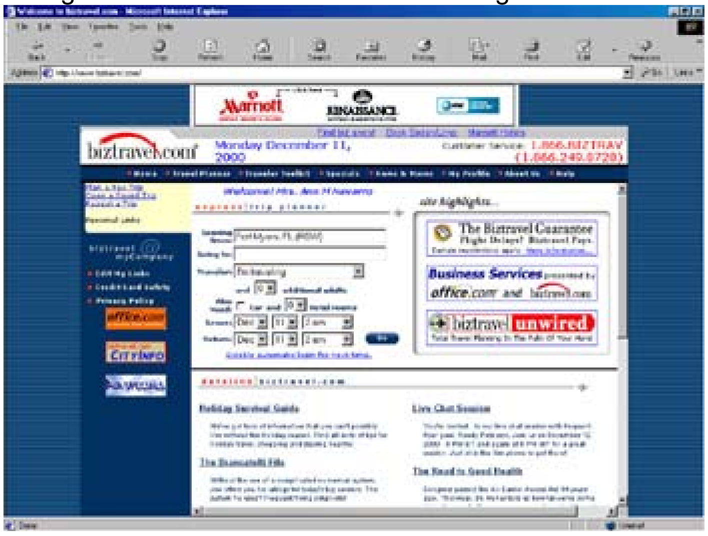


Figure 8.1: The Biztravel.com form makes reservations easy for frequent business travelers.

- Where the trip is originating
- Where the flight will flight go
- Who is travelling and how many others will go along
- Whether a car rental or hotel is needed upon arrival
- The departure and return date

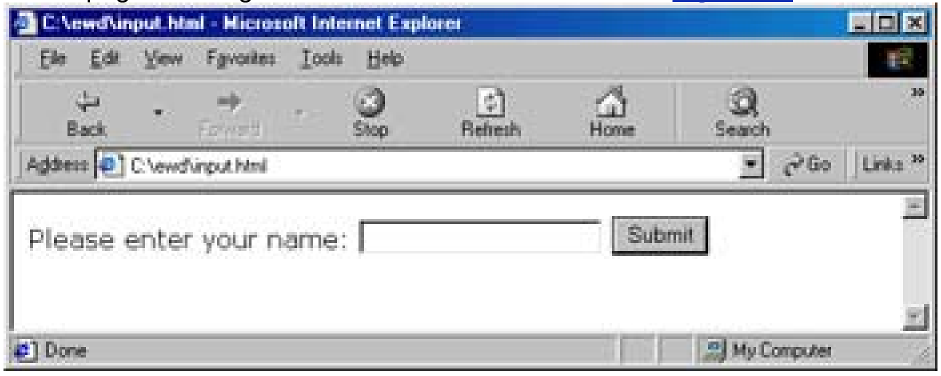
The communication between the visitor and the Web site is successful on several fronts. All the information is collected in a very small space, in a layout that blends beautifully with the rest of the site. The site returns information to the user in a useful and meaningful format.

Preparing the Form

A basic form consists of the `form` container and two or more elements called *controls* that are placed within the container. A control is an XHTML element that provides a means for the user to enter or select data. That data can be a check mark in a box, or it can be text-based information like a name or e-mail address. Regular text and image-based content can also be included within the form container. Here's a very basic form:

```
<form>
Please enter your name: <input type="text" name="name" size="20" />
<input type="submit" value="Submit" />
</form>
```

The page resulting from this bit of XHTML is shown in Figure 8.2.


C:\ewd\input.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\input.html
Go Links
Done
My Computer

Please enter your name: [   ] Submit

Figure 8.2: A basic form containing a text input and a submit control

The `input` element is used for each of the controls. The `type` attribute defines what kind of control it will be. The first example is a `text` control, which creates a box for the user to type in their name. The second control provides the button labeled Submit, which activates the form. Table 8.1 contains a list of 10 controls available to you.

Table 8.1: Ten Input Control Types

INPUT CONTROL TYPE	CREATES
`text`	A text input box. It may only be a single line wide.
`password`	Also a text input box. The characters being typed are hidden from view to maintain security.
`checkbox`	A box that holds a check mark or `x` when selected. It toggles between `on` and `off`.
`radio`	Similar to a `checkbox` but used in groups. Only one option per group may be active.
`hidden`	Allows inclusion of predetermined *`key=value`* pairs (see the following "Naming Controls" section) to be submitted with the form. The browser does not display the pairs.
`submit`	Creates a standard button used for form submission.
`image`	Allows use of an image in place of the standard button for form submission.
`reset`	Inserts a standard button used to return the values of all input fields to their original state.
`select`	Provides a list of options for the user to choose from. The list may be in drop-down-menu or list-box form.
`textarea`	A text input area larger than a single line.

Tip The XHTML 1 recommendation provides for other control types that require additional markup or programming techniques. The other control types are beyond the scope of this text; some aren't widely used.

Both of the `input` elements in the example had additional attributes that I'll look at in turn.

The Text Box

The first control in the form was a text box, which is probably the most common control you'll see on the Web because it's so versatile. Any information you want to collect from a Web-site visitor—a word, phrase, sentence, or even just a character or two—can be placed into a text box. Visually, a text box

can only span a single line of text. That is, the control box rendered by the browser will sit on only one line—text boxes don't wrap as text lines do.
The `input` element used for the following control has two new attributes, `name` and `size`:

```
<input type="text" name="name" size="20" />
```

In addition to these two, attributes of `value` and `maxlength` are possible, which would set an initial text string to appear within the control, and the maximum length of the input, respectively.

Naming Controls

Every control must have a name. The browser collects form data in *`name=value`* pairs and sends it to the server in URL-encoded form. The name half of the *`name=value`* pair is taken from the `name` attribute.
For clarity's sake, it's often easiest to think of a control's `name` as its label. If you're asking the user to input their first name, you might use a `name` value of `first-name` or simply `name` as the example did.

Tip Because CGI programs (which I'll look into in more detail near the end of this chapter) that receive the value of the name attribute often operate on that data, it's a good idea to be brief (while also maintaining a balance with readability) when choosing the value. Common practices include using mixed case lettering instead of spaces—e.g., FirstName instead of first name—or even abbreviating it down to a single word such as fname. Whatever method you choose, be sure you can "decode" it later!

Defining a Control's Size

The `size` attribute, as it applies to text boxes, defines how wide the box will be in terms of a number of character spaces. You might think of that as the number of characters that will fit *in view* within the text box rendered on the user's screen. If the user has a default font size of 36 points, the text box in the example would be 20 36-point spaces wide. If instead the default font size were a more customary 12 points, the text box would be 20 12-point spaces wide.
You might guess that, if `size` defines the number of characters in view, the box could accept more input than the number of characters it is wide. That's true. If more characters are entered than fit in the viewable area, the text should scroll sideways as the user types. Even so, when choosing a text-box size, be sure to use a value that would meet one of the longest possible entries you could imagine. A control asking for the user's last name might be adequate for most respondents at a size of 10 characters but would serve almost all respondents at 15 or 20.

Limiting Response Length

At times, you may want to limit the amount of information a user can enter in a text box. The `maxlength` attribute provides that. If the attribute is not present, the default is an unlimited length. The `maxlength` value should never be set smaller than the `size` of a text box; otherwise, the user may understandably get confused.

Predefined Values

The attribute `value` can be used to insert a predetermined string of text into the text box. Although it's most often used in conjunction with other controls, you may use it alone. An addition could be inserted into the first control in the example, such as:

```
<input type="text" name="name" size="20" value="Enter your name" />
```

The result is shown in Figure 8.3.

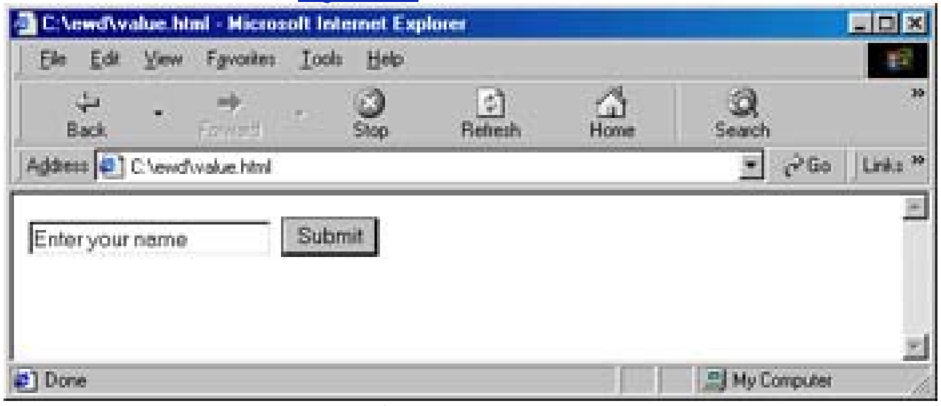

C:\ewd\value.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\value.html
Go Links
Enter your name
Submit
Done
My Computer

Figure 8.3: A text box with a defined value

Password Control

A special form of the `text` control exists for the entry of sensitive information: the `password` control. It accepts the same attributes as the `text` control: `name`, `size`, `maxlength`, and `value`. The difference is in the display. When the user types in the password text box, the input is obscured from view by a series of asterisks or other characters, as shown in Figure 8.4. Others will therefore not be able to read the password as it's typed.

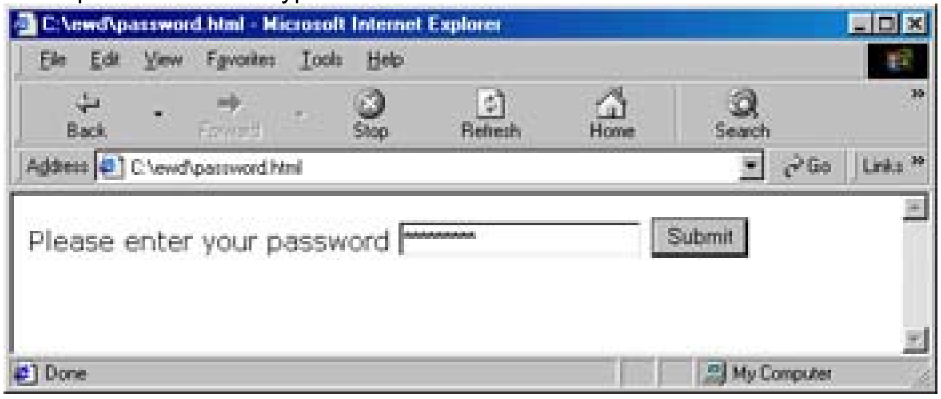

C:\ewd\password.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\password.html
Go Links
Please enter your password
Submit
Done
My Computer

Figure 8.4: Asterisks within a `password` control obscure user input.

Tip Though the user can't view the input into a password control, the data captured by the form is the actual entry. No decoding is necessary for processing.

Check Boxes: Letting the User Choose as Many as Apply

Forms allow a Web designer to collect very detailed information from their site visitors. In order to extract meaningful results from the responses, questions are often stated with a predefined set of answers to choose from. The visitor then checks each answer that applies.

In its most basic form, the `checkbox` control uses only the `type` and `name` attributes. You can make a label by placing text immediately to the right of the control, as shown:

<input type="checkbox" NAME="label" /> label

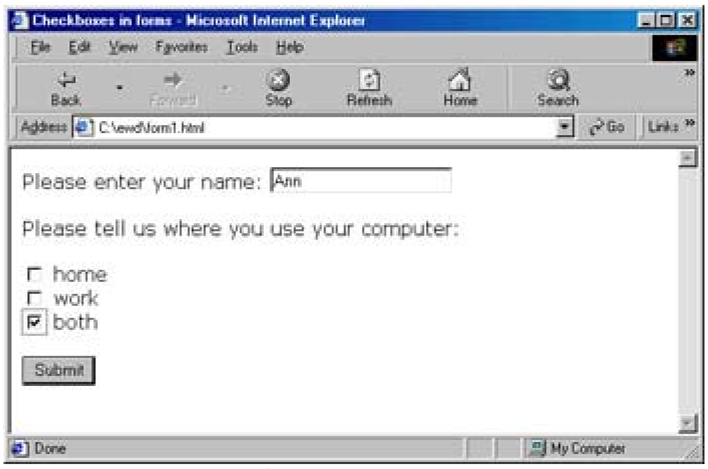
As you saw in Table 8.1, `checkbox` toggles between `on` and `off`. The default state is `off`, which means no check mark is present. This state is stored in the `value` attribute when it is not otherwise defined. When the user places the mark, the state switches to `on`. If the sample here were to be used on a real form and the form submitted with the box checked, the *`name=value`* pair submitted would be `name=on`.

The file `form1.html` shown in Listing 8.1 and on the CD-ROM takes the original form example and adds a question that uses check boxes. The resulting view in the browser is shown in Figure 8.5.

Listing 8.1: form1.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Checkboxes in forms</title>

<body bgcolor="#FFFFFF" text="#000000">



<form>



<p>Please enter your name: <input type="text" name="name"     width="20" /></p>
```

```
<p>Please tell us where you use your computer:</p>


<p><input type="checkbox" name="location" value="home" /> home<br />

<input type="checkbox" name="location" value="work" /> work<br />

<input type="checkbox" name="location" value="both" /> both<br />


<p><input type="submit" value="Submit" /></p>


</form>

</body>

</html>
```


Checkboxes in forms - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\form1.html
Go Links
Done
My Computer

Please enter your name: Ann

Please tell us where you use your computer:

☐ home
☐ work
☑ both

Submit

Figure 8.5: A group of check boxes with one item selected

When reviewing Listing 8.1, you may immediately notice that all three `input` elements use the same `name` attribute of `location` and that the `value` attribute is then set to a specific value. How is the on/off state handled if a `value` is already present? The `value` attribute, when used with check boxes, defines the value to be assumed only during the `on` state. It remains empty if the user leaves the box unchecked. By using the same `name` definition, the checked responses can be easily grouped together when the recipient reviews all the information that the form collected.

Radio Buttons: Letting the User Choose One and Only One

What about situations when a single choice needs to be made out of several available options? An attribute doesn't exist that prevents a person from checking more than one check box. Another control type, however, was made to handle that problem: the `radio` control—often referred to as a *radio button*.

The `radio` control is presented visually as an empty circle. When marked, the circle contains a black dot much like a bullet.

The following snippet can be added to `form1.html`:

```
<p>What type of Internet connection do you use?</p>

<input type="radio" name="connection" value="modem" /> modem<br />

<input type="radio" name="connection" value="ISDN" /> ISDN<br />

<input type="radio" name="connection" value="DSL" /> DSL<br />

<input type="radio" name="connection" value="cable" /> cable<br />
```

```
<input type="radio" name="connection" value="LAN" /> LAN<br />
```

The `value` attribute for the `radio` control is *required* and the `name` values *must* be identical in order for the control set to function properly. The *`name=value`* pair associated with the control inherits the value of the dot that's been selected, such as `connection=DSL`.

Drop-Down Menus and List Boxes

Sometimes a question calls for too many possible responses to arrange neatly using radio buttons or check boxes. The `select` control addresses this scenario beautifully. A `select` control can take two familiar forms:

- **Drop-Down Menu** A list of choices "drop down" when the user clicks on the arrow provided.
- **List Box** A list presents choices in view bordered by a box.

Providing a Default Selection

One nifty option for check boxes and radio buttons is the ability to preselect choices for your users. Preselecting can save the user some time by setting the selections to the most commonly used choices, or it can subtly guide a user toward the choice you prefer.

For example, many companies would prefer to send customers e-mail notices regarding new products and services rather than incur the expense of a paper mailing. A radio control could be added to the form asking for permission to send the respondent occasional news items via e-mail. A predefined selection of Yes means that the user must proactively opt out of such an arrangement.

E-mail traffic is a very sensitive issue on the Net these days. Always be very clear with your visitors as to what you will or won't do with their information. Assure them that you won't sell or disclose their data to any third parties if that is, in fact, the case, and you'll likely get more positive responses.
You can preselect choices by adding the `checked` attributed, which is considered a *Boolean* attribute in XHTML specifications. Boolean here refers to the logical state of being true or false (you can also think of it as on or off). If the attribute is present, the value is inherently true. So in HTML, the `checked` attribute could be present without a value. Today in XHTML, well-formedness constraints require that a value be declared—and so it is here:

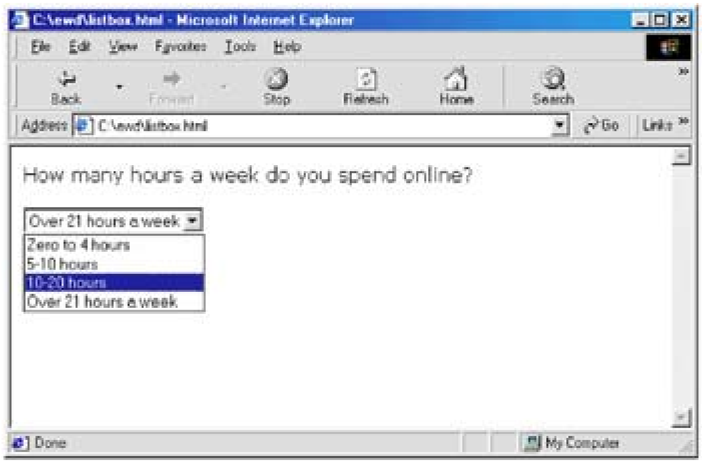
```
<input type="radio" name="connection" value="modem" checked="checked" /> modem
```

The user sees the modem button already marked. A button will be clicked only if the user is lucky enough to surf with a faster connection.

The construction begins with the `select` element, which serves as a container for the list. Each individual list item is then entered using the `option` element, as follows:

```
<p>How many hours a week do you spend online?</p>
<select name="time">
<option value="0-4"> Zero to 4 hours</option>
<option value="5-10"> 5-10 hours</option>
<option value="10-20"> 10-20 hours</option>
<option value="21+"> Over 21 hours a week</option>
</select>
```

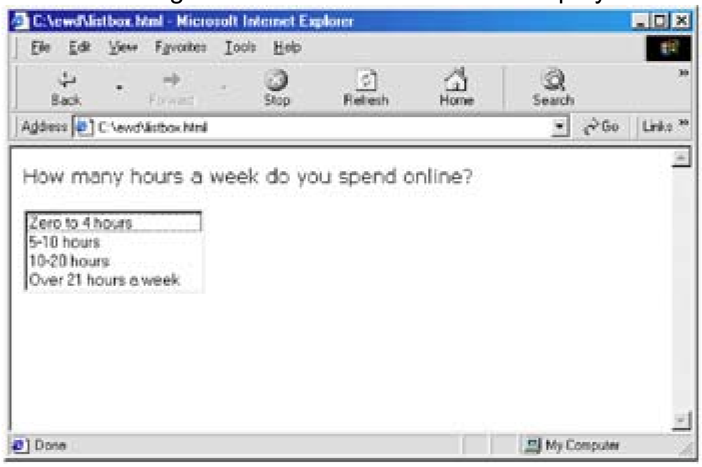
Because the `select` container and its contents are all part of the control, the `select` element contains the `name` attribute. The value of the control is determined by which `option` is selected when the form is processed. The foregoing simple `select` control is displayed as a drop-down menu, as shown here.


C:\ewd\listbox.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\listbox.html
Go Links
Done
My Computer

How many hours a week do you spend online?

Over 21 hours a week
Zero to 4 hours
5-10 hours
10-20 hours
Over 21 hours a week

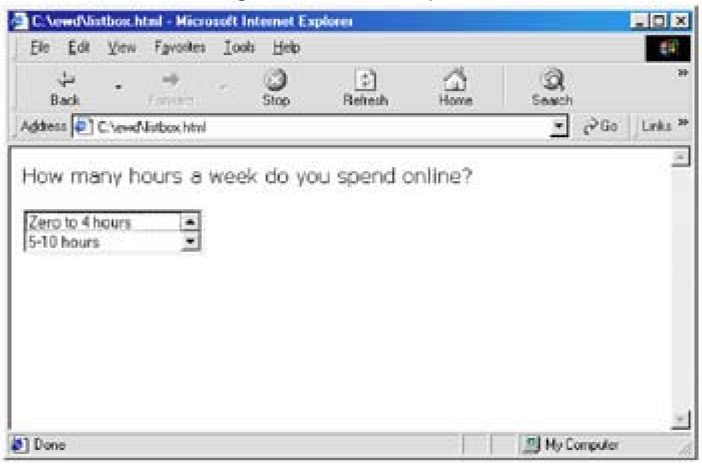
To turn this `select` control into a list box, the `size` attribute needs to be added; `size` determines how many rows will be displayed at a given time. The default value is `1`, which creates the drop-down list. A size of `2` or greater will create the list-box display. The sample seen here has been set to `4`.


C:\ewd\listbox.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\listbox.html
Go Links
Done
My Computer

How many hours a week do you spend online?

Zero to 4 hours
5-10 hours
10-20 hours
Over 21 hours a week

You aren't required to set the size of the list box to the number of options available. If the value is smaller, a scrolling mechanism is provided to let the user see all the choices, as shown here.


C:\ewd\listbox.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\listbox.html
Go Links
Done
My Computer

How many hours a week do you spend online?

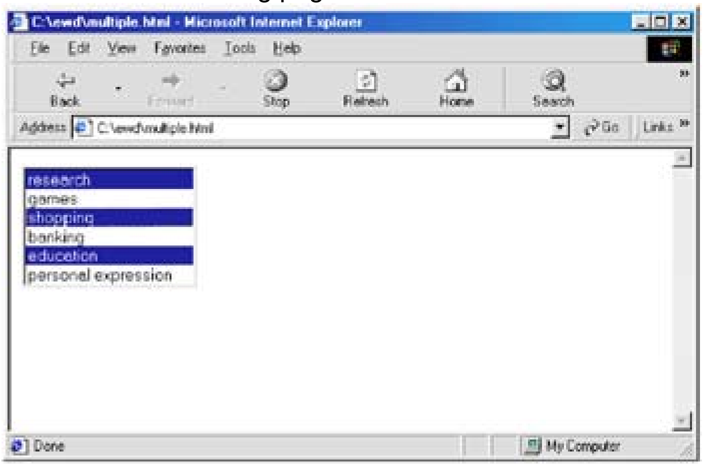
Zero to 4 hours
5-10 hours

Users can make multiple choices from a `select` control with the addition of the Boolean `multiple` attribute:

```
<select name="usage-type" size="6" multiple="multiple">
<option value="research"> research</option>
<option value="games"> games</option>
<option value="shopping"> shopping</option>
<option value="banking"> banking</option>
<option value="education"> education</option>
<option value="expression"> personal expression</option>
```

</select>

By holding down the Ctrl key (Option key on the Mac), the user can highlight more than one choice, as seen in the following page. Each value is then sent when the form is processed.


C:\ewd\multiple.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\multiple.html
Go Links
research
games
shopping
banking
education
personal expression
Done
My Computer

Extended User Input with *textarea*

A `textarea` control is defined as a multiline field. Any time you want your visitors to be able to enter comments, ask questions, or provide information that may take up more than one line, `textarea` is the control to be used.

The syntax for the text area is a bit unusual. It takes the form of a container and has two new attributes: `rows` and `cols`. The following code demonstrates the use of `textarea`:

<p>Please tell us which Web sites are your favorites:</p>

<textarea name="comments" rows="5" cols="60"></textarea>

As with all form controls, the `textarea` must be named. The attributes `rows` and `cols` define the borders of the input area in terms of the number of lines high—rows—and the number of single-spaces wide—columns.

The `textarea` control also allows you to prompt the user by placing some text into the field. Anything contained between the opening and closing `textarea` tag will be displayed inside the field, as seen in Figure 8.6 and in the lines that follow:

<textarea name="comments" rows="5" cols="60">My favorite sites are:

</textarea>

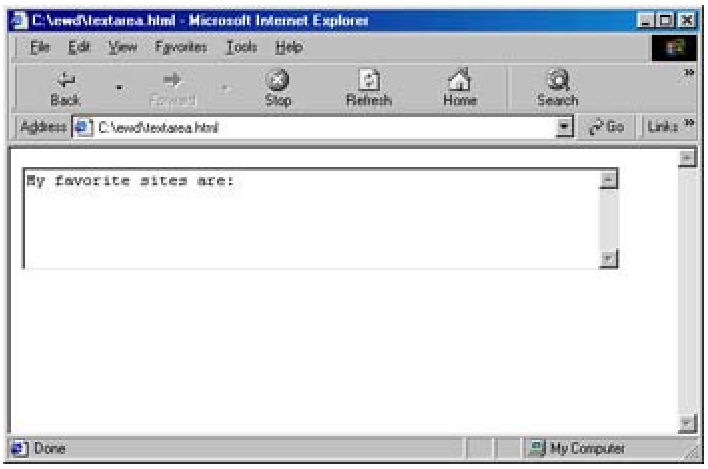

C:\ewd\textarea.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\textarea.html
Go Links
My favorite sites are:
Done
My Computer

Figure 8.6: A `textarea` control with predefined text included

Supplying Hidden Details

In almost any informational exchange between you and your site visitors, you will need static information. For instance, your site might have three forms on it: a survey, an order form, and a complaint form. When you receive responses, each one needs to be labeled properly with the form name. Instead of relying on the form-processing method, you can simply insert the form name using a hidden field. The syntax is very much like a text box with a preset value, as shown here:

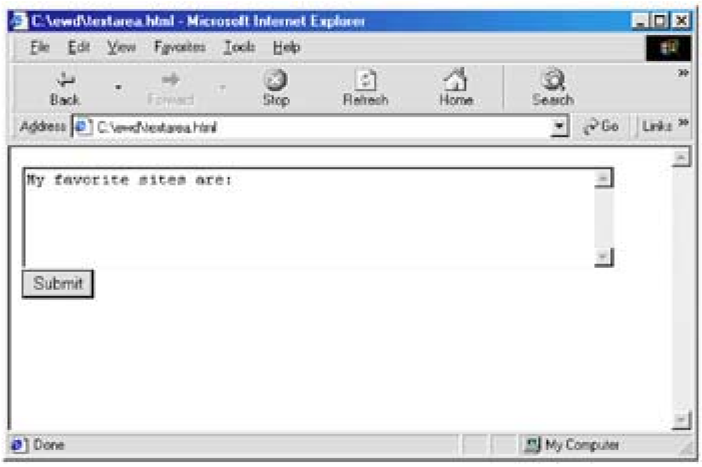
<input type="hidden" name="form" value="WebUsageSurvey" />
This `input` element will generate the *`name=value`* pair of `form=WebUsageSurvey` and send it along with the data that was input directly by your visitor. If you place this tag before any other input fields, you'll be able to sort your responses with just a quick glance at the first *`name=value`* pair.

Submitting the Form

After your users fill out a form on your Web site, they need to be able to send it to you. They can do so by clicking a button that sends the form. Or, if they want to start over and fill out the form from scratch, they can click on a Reset button.

Creating a Submit Button

Almost any Web surfer will recognize the Submit button seen here. Its function is fairly intuitive. It looks like a button, and when a user clicks it, it submits information.


C:\ewd\textarea.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address C:\ewd\textarea.html
Go Links
My favorite sites are:
Submit
Done
My Computer

The `form1.html` file, shown in Listing 8.1, contains the control for a submit button:

<input type="submit" value="Submit" />
The control can function by itself, without any attributes beyond the `type`. Most browsers will default to insert the text "Submit" or some other relatively self-evident text on the button if the value has not been set. To make sure all of your visitors see the same text or to provide something else entirely, use the `value` attribute to define the button's label.

Tip With Navigator 4, Netscape broke with the tradition of the default value for a Submit button, moving from simply "submit" to "submit query." To accommodate an audience that may not understand why the term "query" would be attached to an order form or other response that isn't asking a question, consider making a habit of defining your own values for all submit controls.

Replacing Submit with an Image

The ubiquitous nature of the Submit button has prompted many a Web designer to wish for an alternative. With the `image` input control type, you now have a choice. To replace the ordinary gray button with a custom image, replace the standard markup `<input type="submit" value="Submit" />` with the following:

<input type="image" src="*mybutton.gif*" ALT="Submit Button" />
This control includes attributes you're familiar with from the `img` element: source location and alternative text. It's the input type that makes this function as a submit button.

Tip Notably, the image input type does not take the height *and* width attributes used on the img element, which has confused many designers who've learned the correct practice of using height *and* width with their images. But the image input type specifies a form control, not an image; therefore, height *and* width do not apply.

Resetting Form Defaults

If you've seen a few Submit buttons, you've undoubtedly seen them paired with a Reset button. When clicked, this button will reset all form fields to their original values. If no predetermined values were set, the form will be wiped clean. If fields did have predetermined values, the form will once again show just those values.

Putting It All Together

When the time comes to build a form, it's helpful to visualize it or even sketch it on paper. Forms can be hard to make look good if you just make them up as you go. Ask yourself the following questions:

- Are there questions that do best in a true/false format?
- Must the response be chosen from a finite set of options?
- Can more than one choice be made from a set of answers?
- Should the user be able to write in a choice if an appropriate one isn't present?
- Does the user need to have room to explain or comment at length?
- Do you need to limit the length of responses because of constraints in processing or field-size limits in databases?

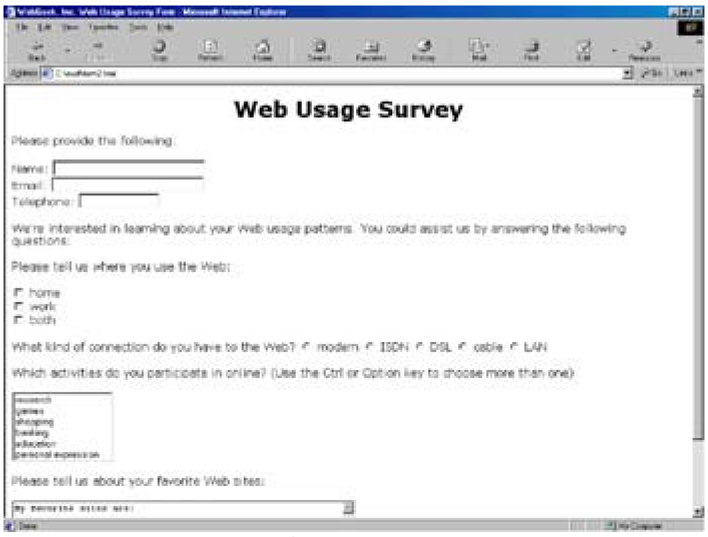
The answers to these questions will help you choose what input controls to use in each area of the form. The file `form2.html` shown in Listing 8.2 and on the CD-ROM makes use of seven different control types: `text`, `radio`, `checkbox`, `select (list box)`, `textarea`, `submit`, and `reset`. See also the rendering in Figure 8.7.

Listing 8.2: form2.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>WebGeek, Inc. Web Usage Survey Form</title>

</head>

<body bgcolor="#FFFFFF" text="#000000">



<h1 align="center">Web Usage Survey</h1>



<form method="post" action="http://www.webgeek.com/cgi-bin/forms.cgi">



<input type="hidden" name="Form" value="Web Usage Survey Form" />



<p>Please provide the following:</p>



<p>Name: <input type="text" name="name" size="30" /><br />

Email: <input type="text" name="email" SIZE="30" /><br />

Telephone: <input type="text" name="phone" size="15" /></p>



<p>We're interested in learning about your Web usage patterns. You

  could assist us by answering the following questions:</p>
```

```
<p>Please tell us where you use the Web:</p>


<p><input type="checkbox" name="location" value="home" /> home<br />
<input type="checkbox" name="location" value="work" /> work<br />
<input type="checkbox" name="location" value="both" /> both<br
/></p>


<p>What kind of connection do you have to the Web?
<input type="radio" name="connection" value="modem" /> modem
<input type="radio" name="connection" value="ISDN" /> ISDN
<input type="radio" name="connection" value="DSL" /> DSL
<input type="radio" name="connection" value="cable" /> cable
<input type="radio" name="connection" value="LAN" /> LAN</p>


<p>Which activities do you participate in online? (Use the Ctrl or
   Option key to choose more than one)</p>
<select name="usage-type" size="6" multiple="multiple">
<option value="research"> research</option>
<option value="games"> games</option>
<option value="shopping"> shopping</option>
<option value="banking"> banking</option>
<option value="education"> education</option>
<option value="expression"> personal expression</option>
</select>


<p>Please tell us about your favorite Web sites:</p>
<textarea name="comments" rows="5" cols="60">My favorite sites are:
   </textarea>
<p><input type="submit" value="Submit Survey" />
<input type="reset" /></p>
```

```
</form>

</body>

</html>
```



Web Usage Survey

Please provide the following:

Name:
Email:
Telephone:

We're interested in learning about your Web usage patterns. You could assist us by answering the following questions:

Please tell us where you use the Web:

home
work
both

What kind of connection do you have to the Web? modem ISDN DSL cable LAN

Which activities do you participate in online? (Use the Ctrl or Option key to choose more than one)

research
games
shopping
banking
education
personal expression

Please tell us about your favorite Web sites:

My favorite sites are:

Figure 8.7: A complete form

Though Figure 8.7 shows a fully functional form, it could use some improvement in layout:

- Notice that the top three text boxes are unevenly aligned.
- Having the connection choices spread out across a single line makes them difficult to scan quickly.

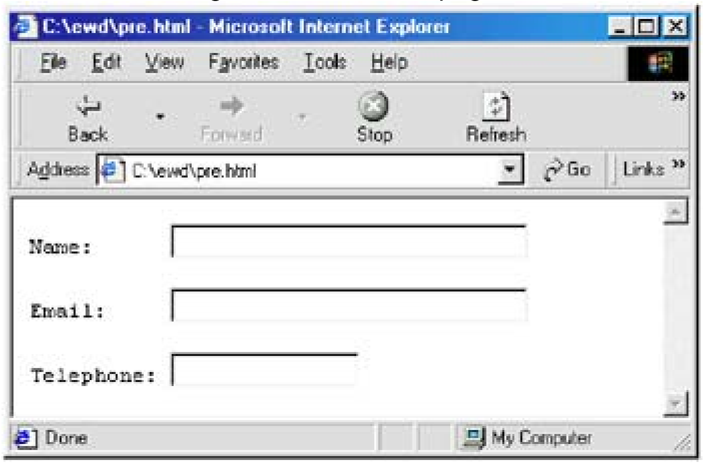
Layout and presentation within forms can be improved upon with additional XHTML markup or style-sheet properties.

Aligning Form Fields

The reason the three text boxes appear unaligned should be pretty obvious: The text that immediately precedes them contains a different number of characters on each line, as shown:

```
<p>Name: <input type="text" name="name" size="30"><br />

Email: <input type="text" name="email" size="30"><br />

Telephone: <input type="text" name="phone" size="15"></p>
```

A basic solution would be to enclose the fields within a preformatted text area using the `pre` element container. Unfortunately, though that solves the spacing problem, it also generates a problem of it's own—the change from the default page font to the monospaced font for the field labels, as seen here.



C:\ewd\pre.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh
Address C:\ewd\pre.html Go Links

Name:

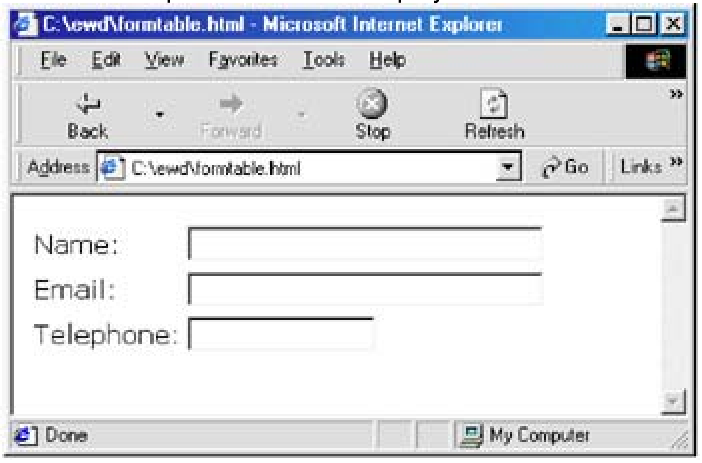
Email:

Telephone:

A better solution would be to create a two-column table, as follows:

```
<table border="0">
<tr>
<td>Name:</td>
<td><input type="text" name="name" size="30"></td>
</tr>
<tr>
<td>Email:</td>
<td><input type="text" name="email" size="30"></td>
</tr>
<tr>
<td>Telephone:</td>
<td><input type="text" name="phone" size="15"></td>
</tr>
</table>
```

As you can see, the alignment looks the same as with the preformatted text solution, but the table doesn't tamper with the font display.


C:\ewd\formtable.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Address C:\ewd\formtable.html
Go
Links
Done
My Computer

Name:
Email:
Telephone:

The next area that could use some work is the group of check boxes and radio buttons. The current presentation has the check boxes in a vertical column, and the radio buttons in a horizontal row:

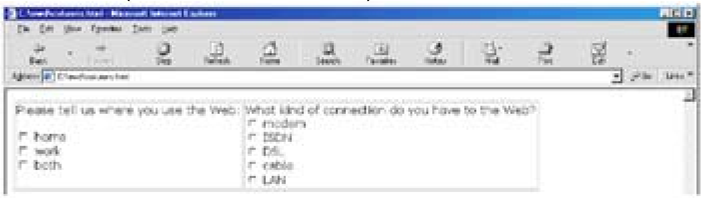
```
<p>Please tell us where you use the Web:</p>

<p><input type="checkbox" name="location" value="home" /> home<br />
<input type="checkbox" name="location" value="work" /> work<br />
<input type="checkbox" name="location" value="both" /> both<br /></p>

<p>What kind of connection do you have to the Web?
<input type="radio" name="connection" value="modem" /> modem
<input type="radio" name="connection" value="ISDN" /> ISDN
<input type="radio" name="connection" value="DSL" /> DSL
<input type="radio" name="connection" value="cable" /> cable
<input type="radio" name="connection" value="LAN" /> LAN</p>
```

The radio buttons display on a single line because the `input` element does not inherently generate a line or paragraph break. Until the browser reaches a new `p` or `br` element (or a block-level element that implies one), it will continue to place the elements inline until it's forced to wrap at the right edge of the available window.

Two columns would present a more traditional grouping of items. A table works best in this instance also, as shown:

```
<table border="0">
<tr>
<td valign="top">
<p>Please tell us where you use the Web:</p>

<p><input type="checkbox" name="location" value="home" /> home
<input type="checkbox" name="location" value="work" /> work
<input type="checkbox" name="location" value="both" /> both</p>

</td>
<tr>
<td valign="top">
<p>What kind of connection do you have to the Web?
<input type="radio" name="connection" value="modem" /> modem
<input type="radio" name="connection" value="ISDN" /> ISDN
<input type="radio" name="connection" value="DSL" /> DSL
<input type="radio" name="connection" value="cable" /> cable
<input type="radio" name="connection" value="LAN" /> LAN</p>
</td>
</tr>
</table>
```

The output will be two columns of choices, three on the left and five on the right, as you can see. Notice that the attribute `valign="top"` was used in the `td` cell. That assured that the top of the two columns will be aligned, rather than leaving the smaller left-hand column aligned to the middle of the right-hand column (the default behavior).



Please tell us where you use the Web:	What kind of connection do you have to the Web?
☐ home	☐ modem
☐ work	☐ ISDN
☐ both	☐ DSL
	☐ cable
	☐ LAN

Processing the Responses

So far, one key instruction has been missing from the form examples—how to actually process the form. You have two basic options: process through the user's mail client or through a server. These two options are commonly known as *mailto forms* and *CGI scripts*.

In either case, when the Submit button is pressed, the browser gathers the information entered by the user, joins it into `key=value` pairs, and then converts it into *URL-encoded format*. URL encoding takes the `key=value` pairs and *concatenates* them—joins them together—into one long string of characters. Spaces, carriage returns, and line feeds are replaced by their ASCII character codes. The long string is then transferred to the form processor.

Using Mailto Forms

The first process, a mailto form, is easier for the novice Web designer to implement, but it's the poorer of the two choices. A mailto form takes the URL-encoded data string and sends it to a designated e-mail address through the visitor's e-mail program. The catch is that the visitor's Web browser must be "aware" of its e-mail program. While Navigator has long had an e-mail client integrated right into the product, not all Web browsers have one—not even IE, though it has become more tightly integrated with Outlook or Outlook Express. Visitors using browsers without an e-mail client may have difficulty completing this type of form. An alternative contact method such as a telephone number or e-mail address should be plainly available.

Adding the Form Action

To initiate mailto handling, two new attributes need to be inserted in the opening `form` element: `method` and `action`. The `method` attribute tells the browser how to pass off the URL-encoded string of data. For most purposes, you will use the value `post` here. The `action` attribute defines what process will "act" upon the string. As discussed here, that's an e-mail link, as shown:

<form method="post" action="mailto:*you@yourcompany*.com">

The instruction `mailto` tells the browser to activate the e-mail client and send the data (the URL-encoded string) to the address specified.

Warning It is difficult to predict what will happen when a Web browser doesn't understand how to handle a mailto form. One of the most common scenarios is that the data simply falls into the "bit bucket" and is never sent to you. Or, the e-mail client will come up with a blank message addressed to the e-mail address noted in the action, with no form data. Not a good thing!

What Arrives in the Mail

Adding the processing instructions to the form was the easy part. The downside to mailto forms, outside of the spotty handling capability among browsers, is that the responses arrive in e-mail—often as an attachment instead of in the body of the message—in the same URL-encoded format that the browser passed them off in.

Filling out the form pictured in Figure 8.7 with mailto processing, the following data came back:

```
Form=Web+Usage+Survey+Form&name=Ann+Navarro&email=ann@webgeek.com
↪&phone=5551234&location=both&connection=DSL&usagetype=banking&
↪usagetype=games&usagetype=shopping&comments=My+favorite+web+
↪site+is+ http://www.bingozone.com
```

Yikes! If you look closely you can see the `name=value` pairs: *`form=Web Usage Survey Form`*, *`location=both`*, etc. But it's far from easy reading in this state.

A far better solution is to employ a few lines of code in one of many programming languages that can handle information passed in the URL-encoded state.

Processing with CGI Scripts

CGI stands for *Common Gateway Interface*. Ordinarily, the Web browser communicates directly with the Web server. Requests for HTML and graphic files go from browser to server, and the files are passed back in response. When the browser needs something to be processed, rather than having a file handed back, the request passes through the CGI.

A CGI script is a small program—a script—that runs on the server based on a request that's come through the gateway. For Web purposes, they're most often written in Perl or C, though with the rise in popularity of Windows NT-based servers, more are cropping up in other languages. At the most basic level, all that's required is that the language be able to create an executable file that runs on the same platform as the server.

In order to run CGI scripts on your Web site, you need to have access to a *cgi-bin*, a special directory on the site that holds scripts and executable programs. If you are working in a virtual Web-hosting environment—that is, you pay an Internet Service Provider for Web-site space—you'll need to find out whether you have access to your own `cgi-bin`. Some ISPs won't allow them or will allow use of only the special scripts that they've made available to all their customers.

The opening `form` element on the HTML file takes the `method` and `action` attributes, as it did with mailto forms. This time, the `action` is the URL that leads to the Perl script, as seen here:

```
<form method="post" action="http://www.yourcompany.com/
  cgi-bin/forms.cgi">
```

The file `forms.cgi` shown in Listing 8.3 and on the CD-ROM is one such script. The numbers to the right of many lines are the line numbers, which I will refer to in a moment.

Listing 8.3: forms.cgi

```
#!/usr/bin/perl

   #0

# enter the email address the data should be sent to

# don't forget to use the format name\@company.com


# Next edit the subject line appropriately


# Edit the return variable to hold the URL for the page you want


# your visitor to go to next


# Title is the name of the page they'll go to


# Ask your system administrator for the 'the path to sendmail' on

   your system

# edit below if necessary


# ------ edit only these variables. Do not remove quotes -----------#

$email = "you\@yourcompany.com"; #1

$subject = "Form Responses";    #2

$return = "http://www.yourcompany.com"; #3

$title = "My Company Home Page";  #4

$SENDMAIL = "/usr/sbin/sendmail -t";  #5

# ---------------- do not edit below this line --------------------#

read(STDIN, $data, $ENV{'CONTENT_LENGTH'});  #6

open (MAILOUT, "| $SENDMAIL") || die "Error, where's your sendmail?";
    #7

@response = split(/&/,$data);  #8
foreach (@response) {          #9
  tr/+/ /;                  #10
  s/=/ = /;                  #11
  s/%(..)/pack("C",hex($1))/ge;  #12
  print MAILOUT "$_\n";         #13
}                              #14
```

```
close(MAILOUT);                 #15

print "Content-type: text/html\n\n";      #16
print "<html><head>\n";                 #17
print "<title>Form Sent!</title>\n";      #18
print "</head><body>\n";                #19
print "<h3>Form Sent!</h3>\n";            #20
print "<p>Follow this link back to
  <a href=\"$return\"$title;</a>.\n";     #21
print "</body>\n</html>\n";              #22
# end                                 #23
```

Tip The Perl script shown in Listing 8.3 outputs a very minimal HTML document. You may have noticed that it's in fact HTML and not XHTML, with the required DOCTYPE declaration, xmlns attributes, etc. I've chosen to leave this script as it was originally developed in the pre-XHTML world in order to retain as much readability for you as possible. The amount of character escaping and other manipulation required to output valid XHTML, I believe, would obscure the lesson at hand: that you can deliver a basic response page generated by your script.

Although this book isn't intended to be a comprehensive resource for CGI scripting, understanding a script that's used as frequently as this one can greatly add to your marketable skills.

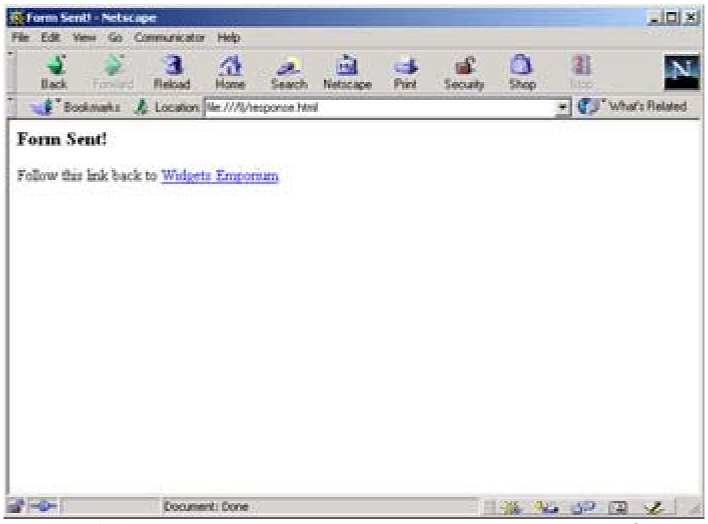
To use this script, you need only edit the six lines marked #0 through #5, as follows:

- **Line 0: Path to Perl** The first of two bits of information you'll need to obtain from your system administrator. The *path* is the route that a computer would travel through the directory structure to get to the exact location of the Perl language interpreter on the server. This line will always begin with the `#!` symbols immediately preceding the path statement and should always be the first line in any Perl script.
- **Line 1: E-Mail Address** This line holds the e-mail address that you want the responses sent to. The @ symbol is always *escaped* by inserting a backslash (\ ) immediately preceding it in the e-mail address. An @ symbol is a special character in the Perl language; by using the backslash you tell the server to interpret the @ as text rather than with its assigned programmatic behavior.
- **Line 2: Subject** This line holds the text that will show up as the subject line in the e-mail message being generated.
- **Line 3: Return URL** After the form is processed, your visitor is taken to an HTML page that informs them of the successful submission. This line holds a URL that will be provided as a link. Enter the URL of the page you want them to move to next.
- **Line 4: URL Title** What do you want to call the URL you're sending them to? The text will tell them "Follow this link back to: *your title here*."
- **Line 5: Path to Sendmail** This is the second bit of information you'll need from your system administrator. Enter the information exactly as they give it to you.

That's it! The rest of the script doesn't change. Upload the file to your `cgi-bin` directory and set any required file permissions before trying to use it. Unix users will want to set permissions to `755`. Others will want to consult their system administrator for any special instructions.

Tip Not every Web server will require you to set file permissions. File permissions are found within Unix and Unix-like servers; they grant specific permission for various events to occur—including the execution of a CGI script. The process of setting file permissions goes beyond the scope of this text. Your system administrator is the best source of assistance for ISP-specific information.

The differences in how the browser handles the form are immediately apparent. With a mailto form, it was hard to tell that anything actually happened. With this script, the user is taken to a whole new page that confirms the success of the process, as seen in Figure 8.8.


Form Sent! - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:///I|/response.html
What's Related
Document: Done

Form Sent!

Follow this link back to Widgets Emporium

Figure 8.8: The HTML page displayed upon successful submission of the form processed by the script `forms.cgi`

There's a marked difference as well in the e-mail you receive from a form submission processed with the foregoing CGI script. The following is the entire contents of the message created by the script:

From: httpd <httpd@*mail.yourcompany.com*>

Date: Mon 23 Oct 2000 15:39:03 -0500

To: you@yourcompany.com

Subject: Web Usage Survey Form

Form = Web Usage Survey Form

name = Maggie Thomas

email = mt@widgets.com

phone = 555-1234

location = both

connection = DSL

usage = banking

usage = shopping

usage = games

comments = My favorite Web site is http://www.bingozone.com

Much easier to read, isn't it? The lines of Perl code between lines 8 and 14 are responsible for changing the information back into a much more legible format.

Tip You'll notice that using this particular script, the From header on this message is the name of the Web server process—httpd. More advanced forms can capture the e-mail address that the user enters and insert it into the headers so you can use your e-mail program's reply function normally. If you were to respond to this message from Ms. Thomas, you'd need to replace the httpd daemon address—the funny-looking From address in the response (httpd httpd@*mail.yourcompany*.com)—with her e-mail address.

Summary

Forms are invaluable for communicating with your Web-site visitors. Information is collected from the user by means of input controls. Common controls include text boxes, radio buttons, check boxes, select lists, and text areas. The `form` container can hold XHTML markup that adjusts layout and presentation as well as the input controls.

Once the user completes the form, two methods of data retrieval are available: direct to e-mail and the use of a Common Gateway Interface (CGI) script. Forms processed through e-mail—mailto forms—are not compatible with all popular Web browsers. CGI scripts are the preferred method for form handling.

Part II: The Site Design Process

Chapter List

Chapter 9: Navigating Your Site

Overview

One site element that novice Web designers can easily overlook is the navigation tools. How does the user move around your site? Can they jump to unrelated sections of the site from any page? Must they use the browser's Back button to trace their steps through half a dozen pages before they can move off in another direction?

Imagine you are walking down an avenue window-shopping and decide to stop in a store that has a few interesting items in the display window. You step through the door and are immediately faced with five different pathways. No signs and no friendly sales staff are available to give directions. You set off down one of the most promising aisles and instead of finding the lamp you wanted to look at up close, you find yourself in men's clothing. How many times will you trace your steps back to the beginning and try again before you give up and walk out the door?

Providing easily accessible and intuitive means of navigating your site will exponentially increase your chances of having your visitors find what they want and stick around to explore some more. How the user moves around the site should be a topic of discussion from the earliest planning stages of your site's development.

This chapter covers the following topics:

- Preproduction planning
- Basic site organization
- Navigational menus
- Site maps
- Navigating with image maps

Preproduction Planning

In Chapter 13, you'll step through the processes of identifying your site's purpose, deciding what response you want your users to have (e.g., learn, buy, or be persuaded), determining design constraints placed on you by an existing trade format, and gathering existing materials to be converted for use on your Web site. It's important to keep navigation in mind while you work through those processes. A solid plan for site construction will go a long way toward saving time in the XHTML production process.

Every Web-design studio and individual Web developer seems to have different methods for laying out and keeping track of how each document fits into the grand scheme of a new Web site. They range from the "crayon-on-napkin" crowd all the way up to those who use powerful database tools or professional flow-charting systems. It is most important to choose what works best for *you*. If you're constantly digging for Post-It notes or searching your hard drive for that image file you worked on last week, you'll lose valuable productivity time.

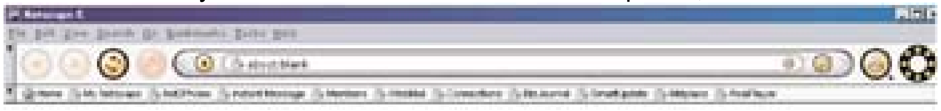
Ideas that can help you plan a Web site more efficiently include:

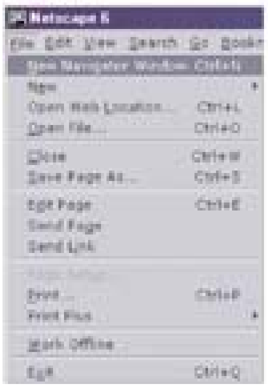
- **Make a basic site map.** A site map can be as simple as an outline or hand-drawn flowchart. Visualizing the site helps determine how many subdirectories you'll need and where each file should go.
- **Keep images in one place.** If you have a single image directory, graphics can be readily located and easily called from any directory within the site.
- **Keep a local mirror image of the site.** When producing your documents, save them in a directory structure that's a duplicate of what will be found on the live Web site. The mirror image makes uploading and maintenance a breeze.

Once you have your basic layout determined, it's time to think about how your visitors will get around your site.

Navigational Menus

Most every computer user is familiar with the concept of a *menu*. Just under the title bar of most programs you'll find the menu bar, a collection of words or phrases that when chosen with a pointing device or a keyboard combination reveal a list of options, as seen here.





Web sites also need menus. Because the look and feel varies so much from site to site, the visitor doesn't always find a familiar navigational model. It's your job as the Web designer to make sure the visitor recognizes the tools you provide as navigation aides and to present them in a logical and attractive manner.

Visual Cues

Figure 9.1 is a screen shot of the Basic Guru Web site. The designer has chosen to provide navigation links in a column down the left side of the page. Such treatment has become very popular. It allows you to set off a relatively small portion of the available screen for hyperlinks, while making the "nav bar," as it's often called, immediately visible to the site users.

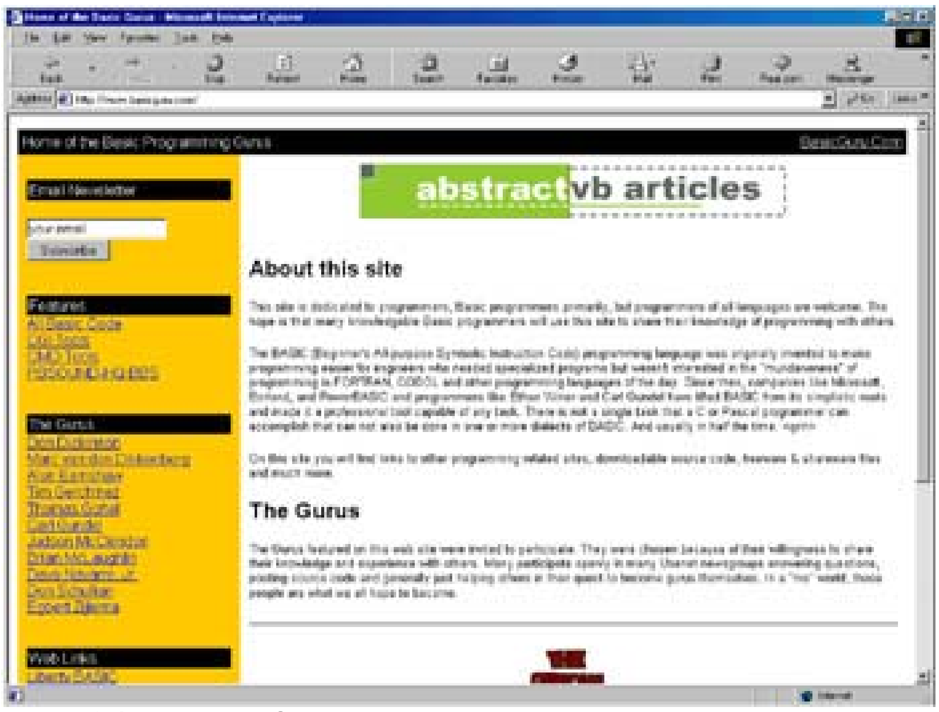
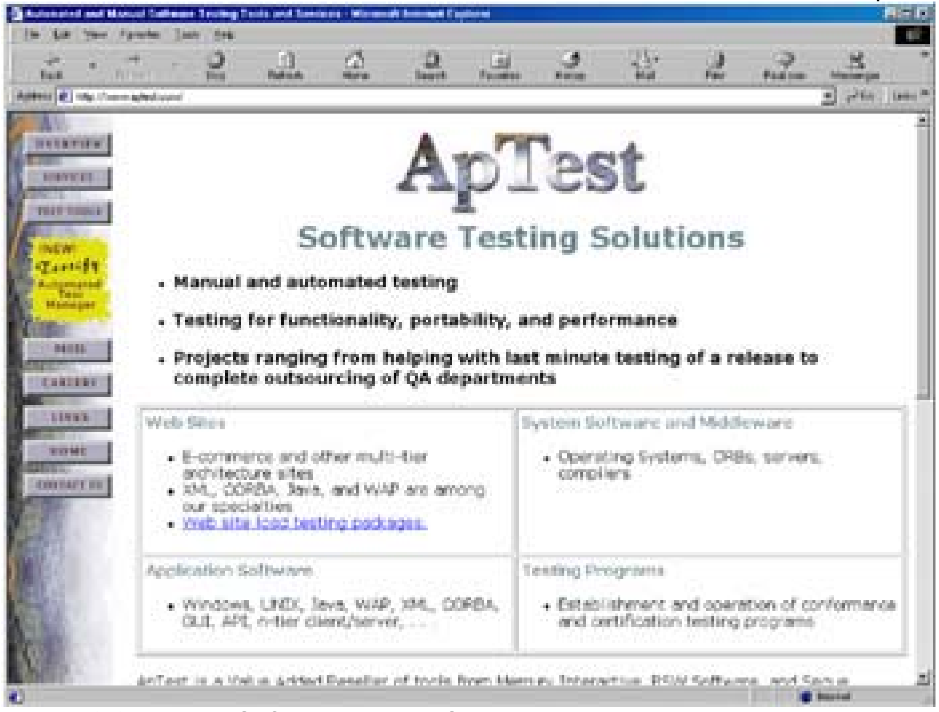


Figure 9.1: The Basic Guru Web site sets navigation links all in one location but also divides them logically by purpose.

On this site, the hyperlinks are not only arranged in a left-hand column but also separated into four groups set apart by black section dividers. Each group of links is topically related—an option to subscribe to the site's e-mail newsletter, links to featured areas of the site, a list of the "gurus" who provide site content, and other links of interest to Basic programmers.

In Web-design vernacular, *buttons* describes graphics that represent links. The metaphor originated very early on from the use of images that, well, looked like buttons—something that you'd push if they were activated by touch.

In a variation on the left-column nav bar, Figure 9.2 shows the ApTest Software Testing Solutions site, located at `http://www.aptest.com`. The eight graphical buttons provide access to every major segment of the site. The colors chosen for the buttons, a solid gray background and a magenta type, blend well with the abstract color waves found beneath them and in the primary ApTest logo.



- Manual and automated testing
- Testing for functionality, portability, and performance
- Projects ranging from helping with last minute testing of a release to complete outsourcing of QA departments

Web Sites

- E-commerce and other multi-tier architecture sites
- XML, CORBA, Java, and WAP are among our specialties
- Web site load testing packages.

System Software and Middleware

- Operating Systems, ORBs, servers, compilers

Application Software

- Windows, UNIX, Java, WAP, XML, CORBA, GUI, API, n-tier client/server, . . .

Testing Programs

- Establishment and operation of conformance and certification testing programs

Figure 9.2: ApTest Software Testing Solutions: navigational buttons that really do look like buttons

Beyond the buttons, designers often use graphics to represent a site's subsections. The Ford Motor Company—found online at `http://www.ford.com`—takes this approach (see Figure 9.3). Each of the company's automotive nameplates is featured by its logo, linking the visitor to the brand's Web site.

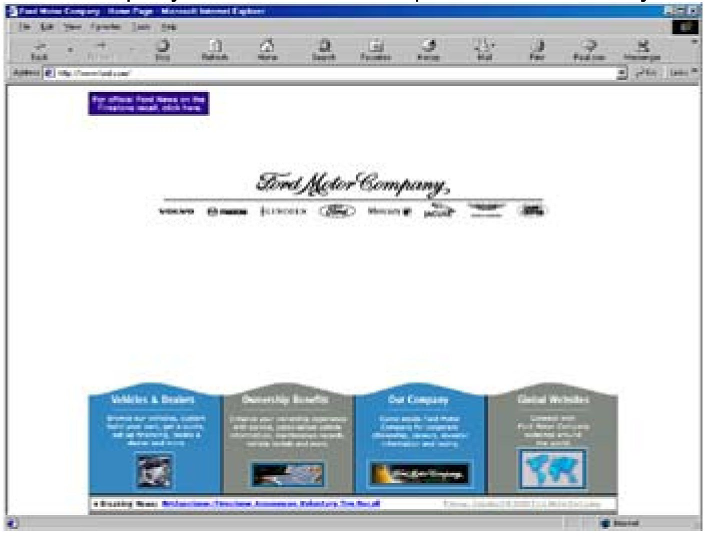


Figure 9.3: The Ford Motor Company uses representational icons for initial navigation.

Tip Notice that although almost every nameplate on the Ford Motor Company Web site includes a graphic (one nameplate has just stylized text), each incorporates the name of the brand. No matter how obvious you may think a visual representation is, you should provide the written meaning to prevent confusion or misunderstanding. For example, U.S. residents have little trouble understanding the red flag on the side of a mailbox. When it's up, it's alerting the mail carrier that mail is in the box to be picked up. When some online services began using that metaphor to alert users to waiting e-mail, many non-U.S. residents were perplexed by the imagery. Their postal delivery systems are handled differently, and they'd never seen mailboxes with red flags.

A well-designed navigational system will not only inform your site visitors of where they *may* go but also where they *are*. An often-frustrating reality of the hyperlinked state of the Web is that your visitors won't always come to your site through the "front door." People often bookmark and link in to pages several levels deep into your Web site. Or they may enter from any individual page through a search engine, as engines such as AltaVista can provide direct references deep into your site.

Figure 9.4 shows a subsection from the Office Depot Web site. This navigational system successfully implements the features already discussed in this section and, in the gray area immediately below the main red navigation bar, provides location orientation with the successive string of links representing each level of the site.

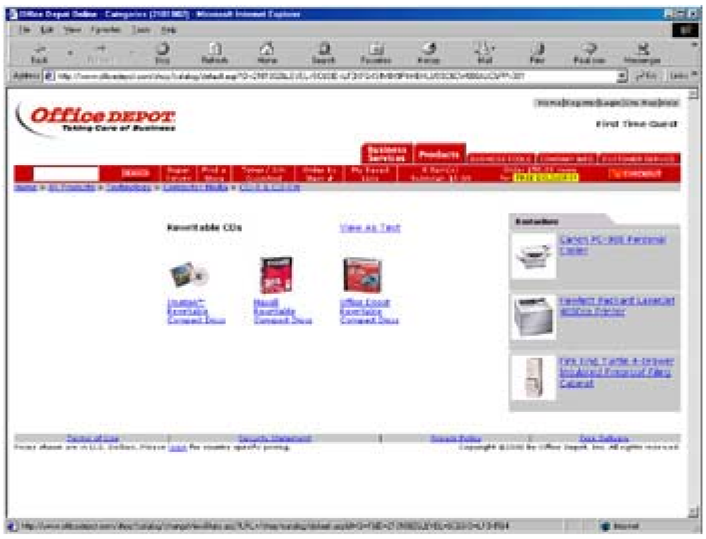


Figure 9.4: Office Depot: keeping track of where you are

Table 9.1 lists six features that you should consider incorporating into your Web site's navigational tools.

Table 9.1: Features that Enhance Navigation

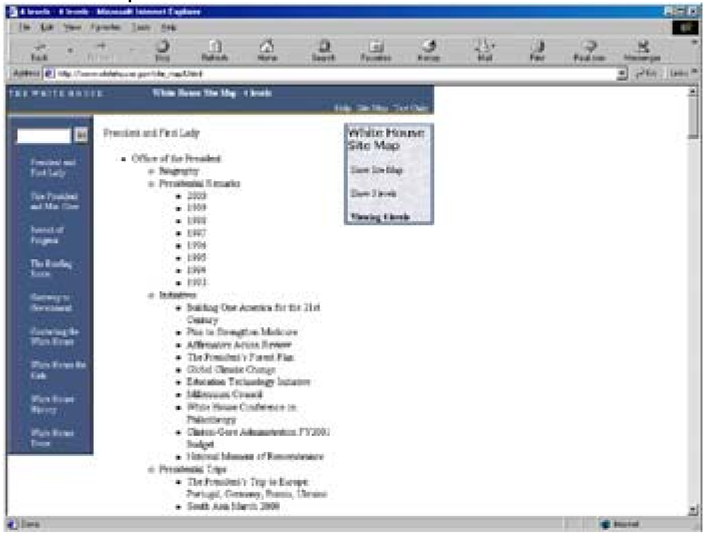
FEATURE	HOW IT HELPS

Table 9.1: Features that Enhance Navigation

FEATURE	HOW IT HELPS
Buttons	Even novice Web users are familiar with buttons from the Windows and Macintosh user interfaces. Carrying that feature into your site design, your user will naturally associate buttons with links.
Bullets	Bullets or other small geometric images help draw attention to lists. A row or column of items set off by bullets grabs the user's focus, helping them immediately notice the navigation area.
Distinct visual space	Setting off the links by using a border, an identifiable column, or another visually separating characteristic helps identify the panel as navigation tools.
Image labels	Don't assume every visitor will understand pictorial representations. Provide them with clear labels for each link, in addition to the images.
A consistent interface	If you use the same design and location for your navigation tools, the user will know where to look in each section. The individual items in the group can change, but the successful site keeps the look and placement across all pages.
"You are here"	Not everyone comes in the "front door" of your site. Provide users with cues as to their current location as well as where they can go.

Site Maps

Back in the preproduction planning phase, I encouraged you to create a basic site map or flowchart that described your site's structure. On large or complex sites, this can be expanded into a navigational tool for your visitors as well.
Figure 9.5 shows a site map for the White House Web site. Visitors have an option to see the map at two levels deep, with just one set of subheadings under each major link, or up to four levels deep, with an in-depth view of the site structure.


White House Site Map - 4 levels

President and First Lady

- Office of the President
  - Biography
  - Presidential Remarks
    - 2000
    - 1999
    - 1998
    - 1997
    - 1996
    - 1995
    - 1994
    - 1993
  - Initiatives
    - Building One America for the 21st Century
    - Plan to Strengthen Medicare
    - Affirmative Action Review
    - The President's Forest Plan
    - Global Climate Change
    - Education Technology Initiative
    - Millennium Council
    - White House Conference on Philanthropy
    - Clinton-Gore Administration FY2001 Budget
    - National Moment of Remembrance
  - Presidential Trips
    - The President's Trip to Europe Portugal, Germany, Russia, Ukraine
    - South Asia March 2000

Figure 9.5: The White House: a variable-depth site map

One of the difficulties of navigating through a large site is that to keep navigational cues intuitive, the amount of information provided in each content page may need to be limited. By providing a single comprehensive catalog of all the destinations available to your visitors, they'll always know where to turn if they find themselves wanting to jump to a new area quickly.

Nonvisual Alternatives

Most of the sites I've discussed so far have made use of graphical elements in order to bring attention to navigational tools. Thinking back to Chapter 2, you'll remember that some browsers, such as Lynx or adaptive programs for individuals with certain handicaps, do not have graphical interfaces. Also, many busy Web surfers will turn off the Automatic Loading of Images option in their browsers when they're searching for specific information and they want to skim sites quickly. Providing a nonvisual, or text-based, alternative for navigation will accommodate both situations.

At the bottom of each page on his Web site, Emeril Legasse, the flamboyant New Orleans chef, makes use of a text-based nav bar, as seen in Figure 9.6. It's simply a centered set of hyperlinks arranged in rows and divided by sets of colon characters (::). There are links to each major component of the site, allowing visitors with images turned off or with text-based browsers to navigate with ease. In fact, there are links to more sections in the text-based nav bar than in the graphical bar at the top of the page. Because only text is used to build the bar, more links can be included in a smaller space without sacrificing readability.

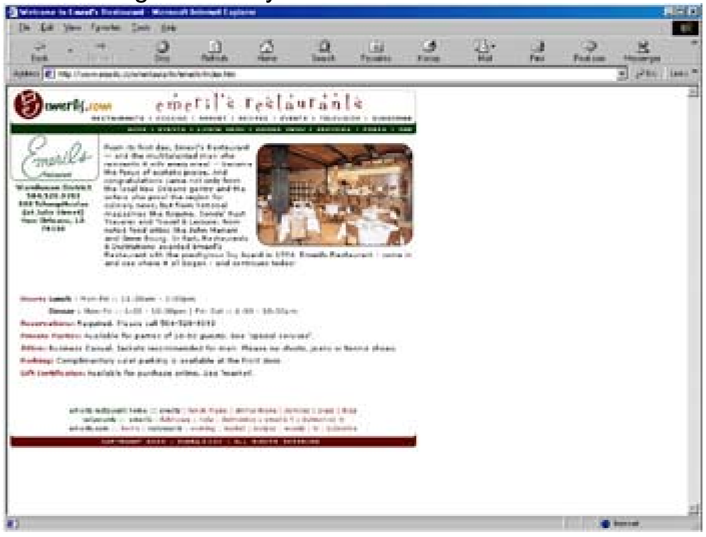


Figure 9.6: A text menu at the bottom of Emeril's Restaurant page

Image Maps

An image map has three components: the *image*, the *map definition*, and the *supporting XHTML*. The image component is something you're already familiar with. Almost any image—a line drawing, a simple shape, and even a photograph—can be turned into an image map.

In Figure 9.7, you can see the opening screen of the Florida Gulf Coast University Web site. At first glance, it's not very different than the other sites I've shown in this chapter. Links appear across the top and in the left-hand column, with a larger graphic in the middle. What's different, however, is when you run your mouse over the phrases arranged in the arc along the right edge of the photo.

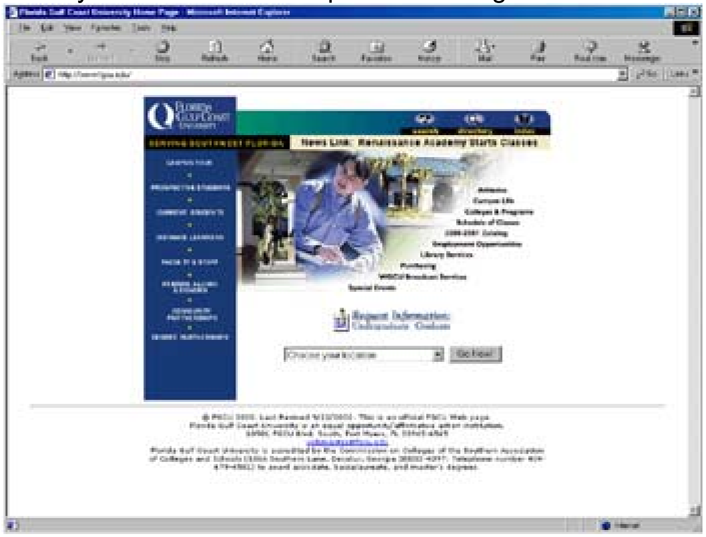


Figure 9.7: The entrance to the Florida Gulf Coast University Web site

Image maps locate the position of your mouse in *x,y coordinates*. If you think back to geometry class for a moment, you'll remember that a two-dimensional object set within two planes can be measured in distances from the x and y axes. With image maps, the concept is similar, with the unit of measure being the pixel. The starting point, where x and y are both equal to zero, is the upper left-hand corner of the image. The x value (the width) grows as the user moves the mouse to the right, and the y value (the height) increases as the user moves the mouse down.

When a user clicks on a *hot spot*, the image-map system reads the pixel coordinates of their mouse pointer and the browser undertakes the action that's specified in the hyperlink anchor for that area. Hot spots are the "clickable" areas within an image map. They can take three shapes: a circle, a rectangle, or a polygon. Each hot spot is hyperlinked to another file using the standard anchor tag you learned back in Chapter 4.

To create an image map, you'll need one new element and a new attribute for the `img` element.

Creating an Image Map

I'll step through the process of creating an image map by using an image that best fits this function—a geographical map. Figure 9.8 is a rudimentary map of the state of Hawaii. The image maps will have hot spots over the major islands, which will be linked to pages detailing resorts on each island.

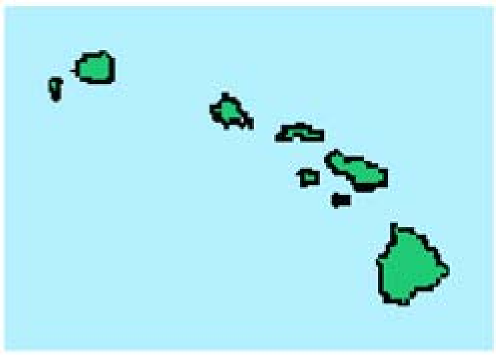


Figure 9.8: I'll use this image to create an image map.

Tip By their nature, image maps tend to be large. Be sure that you've chosen an image that can be optimized for as small a file size as possible to help decrease download time. File-size optimization will be covered in detail in Chapter 17.

The XHTML for your map begins with the familiar `img` element:

```
<img src="hawaii.gif" width="200" height="140" alt="Image map of
  Hawaii" usemap="#MyMap" />
```

The last attribute, `usemap`, is new. Its name identifies its function—it tells the browser an image map definition is associated with the image and that the browser should use the map definition `MyMap` to process it. The object name—in this case the map's name—must always be preceded by the hash mark `#`, as seen here. Next is the `map` element.

The *map* Element

The `map` element is the object that the `usemap` attribute refers to. It holds the x,y coordinates of the hot spots that the browser will interpret to process the hyperlinks. The element begins with the `name` attribute:

```
<map name="MyMap">
```

The name is the one previously referenced in the `usemap` attribute from the `img` element, minus the hash mark.

Within the `map` element is a set of `area` elements that define each hot spot. I'll start with the `area` element for the hyperlink over the island of Hawaii.

```
<area shape="circle"
   coords="165,104,20"
   href="hawaii.html"
   alt="Island of Hawaii" />
```

Tip In this example, each attribute of the area element is shown on a new line for enhanced readability. You do not have to format your area element in this manner.

The `shape` attribute is self-explanatory; the value can be `circle`; `rect`, for rectangle; or `poly`, for a polygon. The `coords` attribute defines the coordinates for the circle (represented by the x,y coordinates of the circle's center point, measured in pixels from the upper left-hand corner of the image) and the radius (also measured in pixels).

Determining the coordinate values can be done by hand or by using one of several handy image-mapping software products available on the market. (See the "Downloadable Image Map Tools" sidebar.) In this case, the information was put together using CuteMap 1.1. The process is described in the following steps. If you're using a different program, the tool names and menu choices may not be the same:

1. With the image open in the program workspace, click the Circle Mask tool from the toolbar.

2. A circle is created within a perfect square. Click your mouse at the top-center of the square that will contain your circle and drag down to encircle the portion of the image desired.
3. Release the mouse and the hotspot will appear with colored cross-hatches over it. You'll see a new area element appear in the lower pane of the CuteMap interface (see Figure 9.9).

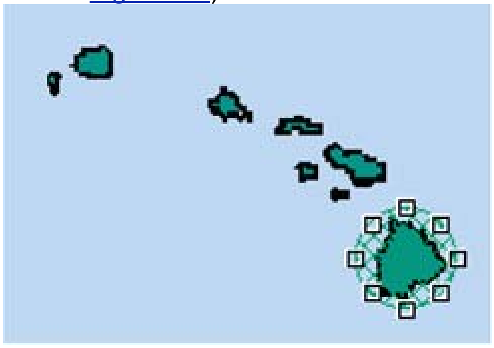


Figure 9.9: The hotspot is marked by a colored cross-hatch overlay.

4. Enter the URL and alt attribute values in the appropriate fields in the left-hand pane of the program.

Now you're ready to find the coordinates for the next `area` element.

The island of Maui, the next island up the chain from the Big Island, is near two smaller islands that are in the way for using a circle for a hot spot. Because the island is also tilted at an angle, a polygon would be the best choice here.

The coordinates for a polygon are listed in pairs of x,y values. The polygon will have five corners, so five pairs of numbers will be listed as the `coords` attribute value, in x1, y1, x2, y2,... order. The values seen in the following code were retrieved again in CuteMap by clicking on each point needed to surround the island and then right clicking on the last point to close the polygon:

```
<area shape="poly"
    coords="143,53,126,60,139,73,153,80,165,69"
    href="maui.html"
    alt="Island of Maui" />
```

Downloadable Image-Map Tools

Calculating hot-spot coordinates by hand can be a drag, especially when you have a considerable number of active areas on the map. Luckily, some software tools are available to help you through the process. Some of the favorites for Windows 95/NT include:

- **Live Image** This shareware tool from LiveImage Corporation, an extension of the well-known Map This!, can be found at `http://www.mediatec.com/`. The Current version is 1.35 (with the trail version remaining at version 1.3). Version 1.26 was a winner in *PC Magazine's* 1997 Shareware Awards.

- **Mapedit** The shareware version by Boutell.Com, Inc. can be downloaded at `http://www.boutell.com/mapedit/`. It is also available for Linux, Java-enabled Unix, and the Mac.

Favorites for the Macintosh include:

- **Enhance 4** From MicroFrontier Online, Enhance 4 is a retail product with a downloadable demo. It is available at `http://www.microfrontier.com/products/enhance40/index.html`.
- **MapMaker** This shareware tool from Twin Moon Development and Design produces code for client-side and server-side maps and can be found at `http://www.kickinit.net/mapmaker/`.

Lanai can be easily covered with a rectangle. The `coords` attribute value is noted by two pairs: the x,y coordinates of the upper-left corner and the x,y coordinates of the bottom-right corner, as seen here:

```
<area shape="rect"
```

```
coords="110,45,133,55"
href="lanai.html"
alt="Island of Lanai" />
```

Warning Be sure when working in tight spaces not to overlap any of the hot-spot boundaries. The map won't function properly if you do.

Warning You'll notice that CuteMap outputs elements and attributes in uppercase and isn't aware of XHTML requirements for closing `img` and `area` elements. You'll need to edit the output in order to be compliant with XHTML requirements.

Adding in the `area` elements for Oahu and Kauai, the final `map` element for `imagemap.html` appears here:

```
<img src="hawaii.gif" usemap="#MyMap" width="200" height="140"
  border="0" />


  <map name="MyMap">
   <area shape="circle"
coords="165,104,20" href="hawaii.html"
alt="Island of Hawaii" />
 <area shape="poly"
coords="143,53,126,60,139,73,153,80,165,69"
href="maui.html"
alt="Island of Maui" />
 <area shape="rect"
coords="110,45,133,55"
href="lanai.html"
alt="Island of Lanai" />
 <area shape="rect"
coords="76,32,105,58"
href="oahu.html"
alt="Island of Oahu" />
 <area shape="circle"
coords="37,26,13"
href="kauai.html"
alt="Island of Kauai" />
  </map>
```

Figure 9.10 shows the finished image map displayed in a Web browser. When the mouse pointer is over a hot spot, the `alt` attribute for that `area` element is displayed as a tool tip in newer browsers like the one shown here.

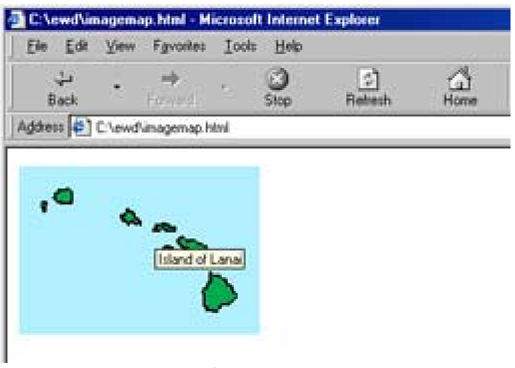

C:\ewd\imagemap.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home
Address C:\ewd\imagemap.html
Island of Lanai

Figure 9.10: The finished image map with the pointer over the hot spot for Lanai

The XHTML seen in the `imagemap.html` listing is known as a *client-side* image map. Client-side means that the client program, in this case the browser, processes the image map. Other image-map renderings are available that are processed by the Web server. Those are known as *server-side* image maps. Server-side processing is generally done with a CGI script. The map data is defined in a file that also resides on the server, instead of with the `map` element, as was done in `imagemap.html`. That file is formatted specifically for the type of Web-server software running on the server where it will be placed. Server-side processing is generally considered the "older way" of handling image maps, though you may wish to include them if you're concerned about accommodating the portion of your audience with older browsers that can't support client-side maps. Creating these files goes beyond the scope of this text. Pointers to information found online are located in the section *Online Resources for Server-Side Image Maps*.

Online Resources for Server-Side Image Maps

Image-map configuration files come in two major types, NCSA or CERN. Your ISP will be able to tell you which one you should use or will provide you with alternate instructions.
A tutorial for image maps created for NCSA servers can be found at `http://hoohoo.ncsa.uiuc.edu/docs/tutorials/imagemapping.html`.
A reference for the CERN format, including the CGI script to go with it, can be found at `http://www.w3.org/Daemon/User/CGI/HTImageDoc.html`.

Summary

How your visitors will navigate your site is as important a part of your design efforts as choosing a color scheme or your content. The process begins with preproduction planning. Mapping out the site for yourself not only aids the development process but also forms the basis of site maps and navigation bars to be produced for the visitor.

Visitors to your site will recognize navigational links best if they're visually set apart from the rest of your content. Popular methods include using nav bars offset with different background colors, a set of custom graphical icons, or even frames. A comprehensive site map gives the user a starting point for all of their explorations—it's like a roadmap for the Web. Image maps can allow you to use traditional graphics hyperlinked with hot spots for navigation.

Chapter 10: Search Engines

Overview

Any time large amounts of information are gathered in one place, supporting documents such as a table of contents and an index are needed for people to find the most relevant data quickly. That's true for the World Wide Web. Several million Web pages may be out there, but how can users find the information of specific interest to them?

This chapter covers the following topics:

- A search engine or an index?

- A look at Yahoo!
- A look at AltaVista
- Optimizing your site for searching
- Getting to know robots
- More popular search sites
- Providing search capabilities on your own site

What Is a Search Engine?

The Internet version of a table of contents is known as a *search engine*. *Search engine* is a generic term for a site that provides a collection of links to other Web sites, with a method of logically searching through those links. Search engines can be directory-style services—think of the yellow pages in your telephone book—or they can have details stored in indexes or databases that respond to keyword searches.

The PC Webopaedia (http://www.pcwebopaedia.com) defines *search engine* as "A program that searches documents for specified keywords and returns a list of the documents where the keywords were found. Although search engine is really a general class of programs, the term is often used to specifically describe systems like AltaVista and Excite that enable users to search for documents on the World Wide Web and Usenet newsgroups."

Tip As the definitions here suggest, in Web vernacular, *search engine* has come to mean any resource that allows a user to search through Web and Usenet listings by topical category or through a programmed keyword search, rather than only by a keyword search. I'll apply the formal terms to the products and services described throughout this section, though you may hear them spoken of generically among your peers or in other media.

Search Engine or Index?

A *search engine* in the context of this chapter is a site that searches any document it can find on the Internet that matches the criteria entered into the search. A Web site can be part of a search engine's database or index through user submission or by being discovered by the search engine's Web robot.

The majority of search engines rely primarily on user submission for the acquisition of new sites. The process is pretty straightforward. The Web developer will visit the search-engine Web site and enter the data requested by that particular service. The developer may be asked for a site description, a list of keywords, the name and e-mail address of a contact person, and sometimes even a mailing address and telephone number. Once the data has been reviewed or merged into the existing service, the site will begin appearing in search results.

Tip Some services are faster than others when incorporating new submissions. Yahoo! has surfers that check all submissions. Others are automated; for example, Alta Vista gets your index page as soon as you submit it and includes it within 24 hours. Anywhere from two days to three weeks can be considered a "normal" response time.

Robots

World Wide Web robots (also called *spiders*) are programs that have been instructed to visit and *parse* Web sites. Parsing is the process of breaking something down into individual pieces. For example, a sentence can be parsed into individual grammatical objects. A Web site is parsed into individual pages and links. What the robot does with the parsed information depends on its programming. Most will, at a minimum, keep a record of the content of the page and note any links contained within it, queuing the links into the list of pages to visit next. The result of all the recording of links and content is a huge database of information that is gathered into the search engine's databases.

But is having *everything* on the Web cataloged and indexed really a good idea? For the most part, it's an invaluable tool. How else could just a few keystrokes bring thousands of documents on subjects as varied as biochemistry to classical music right to your fingertips?

Some portions of Web sites shouldn't be included in the databases of these search engines. Common situations include the contents of a site's `cgi-bin` directory—where CGI scripts are stored and run—or data and log files that are available to the Web-server software but aren't intended for direct display to the user.

In 1994, a group of Web-robot authors realized that a standard set of behaviors for robots would help all site owners control the robots' access to their sites as well as protect the owners from an overly enthusiastic robot that nudges aside human users on limited-bandwidth sites. Discussions evolved into what is today known as the *Robots Exclusion Standard*. The standard allows site owners to control access by robots, keeping them out of binary directories, such as the `cgi-bin`, and blocking access to private data. The agreement isn't the result of work within a standards body such as the W3C, nor is it enforced or guaranteed in any manner. However, in the cooperative environment that exists in many parts of the Internet, the authors of most robots have supported the effort and will likely to continue to do so.

The Robots Exclusion Standard

Robots that honor the exclusion standards are all programmed to look for a file named `robots.txt` in the root directory of every Web site they visit.

Tip The requirement that the robots.txt file reside in the root directory of a Web site does present some difficulties for sites that aren't housed on their own servers or on virtual servers. If this applies to you, ask your provider if they will include information that pertains to your site in their robots.txt file.

Your `robots.txt` file needs to contain, at a minimum, two statements that might look like this:

```
User-agent: *

Disallow: /cgi-bin/
```

The first line addresses a specific *user-agent*; in this context, that's another name for a robot. The asterisk character (*) holds special meaning here, similar to the use of the asterisk as a wildcard in DOS and Windows environments. When present, the asterisk tells any compliant robot that, unless it is named specifically in another section, it must follow the directions stated here. So, if the two lines here were the entire contents of a site's `robots.txt` file, all compliant robots would know to stay out of the `/cgi-bin/` directory.

More than one `disallow` statement can be made per user-agent group. For example, if a site owner wanted to keep robots not only out of the `/cgi-bin/` directory, but also out of directories named `/internal/` and `/private/rawdata/`, the `robots.txt` file would look like this:

```
User-agent: *

Disallow: /cgi-bin/

Disallow: /internal/

Disallow: /private/rawdata/
```

More than one user-agent group or individual agent can be specified as well. Each requires its own set of disallow statements. For instance, one of the known robots is the CACTVS Chemistry Spider. It searches the Web and FTP servers for chemical structures in chemical MIME formats. A site that doesn't deal with chemistry could choose to exclude this spider from all areas yet allow all other known spiders (robots) access to everything but the `/cgi-bin/` directory. The `robots.txt` file would appear as follows:

```
User-agent: cactvschemistryspider

Disallow: /

User-agent: *

Disallow: /cgi-bin/
```

In the first set, the single forward slash tells the CACTVS Chemistry Spider that it's not allowed in the site at all. Other robots skip that instruction and move on to the next set that tells them they may look at anything but the CGI scripts.

Tip Webcrawler maintains The Web Robots Database, a collection of known robot names, contact information for their owners, and descriptions of their activities. It can be found online at http://info.webcrawler.com/mak/projects/robots/active.html.

You can't always wait for the robots to find you. Being listed on an index is an important factor in visitors being able to find your site! In recent years, the number of search engines has exploded. So where do you begin? Consider the granddaddy of all Web indexes:Yahoo!

Yahoo!

Yahoo! (`http://www.yahoo.com`) is one of the Internet's great success stories. What started as a public copy of two Stanford University Electrical Engineering graduate student's browser bookmarks has turned into a multibillion-dollar enterprise. The site now lists hundreds of thousands of sites in hundreds of categories. The company describes its site as follows: "The Yahoo! directory is an online guide to the World Wide Web. It is created by a staff of editors who visit and evaluate Web sites and then organize them into subject-based categories and subcategories." Giving it a little more classical academic bent, Yahoo! public-relations materials note that the *San Jose Mercury News* recently said, "Yahoo is closest in spirit to the work of Linnaeus, the 18th-century botanist whose classification system organized the natural world." See Table 10.1 for a description of some of the search engine's characteristics.

Tip Did you ever wonder where the name "Yahoo!" came from? Turns out it's an acronym for Yet Another Hierarchical Officious Oracle, though the two former grad students have also claimed it's because they view themselves as "yahoos," a term for a race of brutes from the novel Gulliver's Travels and, more frequently, for a bumbling idiot. An interesting self-description for this highly successful pair, don't you think?

Table 10.1: Yahoo! Site Details

DETAIL	EXPLANATION
Submission method	Via form from within the desired Yahoo! category.
Robot name	None
Indexing method	From user-provided descriptions of 25 words or less, approved and potentially edited by Yahoo! surfer staff. Entry within multiple categories at Yahoo! staff discretion.
Adherence to Robots Exclusion Standard	N/A
Search Methods	By keywords. A site is returned if the keyword is found in category headings, listing titles, or individual site descriptions.

Tip One practice that has come under criticism recently is the introduction of the "Business Express" method of site submission to the Yahoo! surfer team. For a one-time nonrefundable processing fee of $199 (at the time of this writing), Yahoo! guarantees a seven-business-day turnaround in reviewing the site—without any assurance of being listed in the directory. Only time will tell what the public reaction will be, but with Yahoo! maintaining its dominance in market recognition, I suspect many commercial sites will ante up.

Yahoo! Organization

The primary Yahoo! site is divided into fourteen main categories, as seen in Figure 10.1.

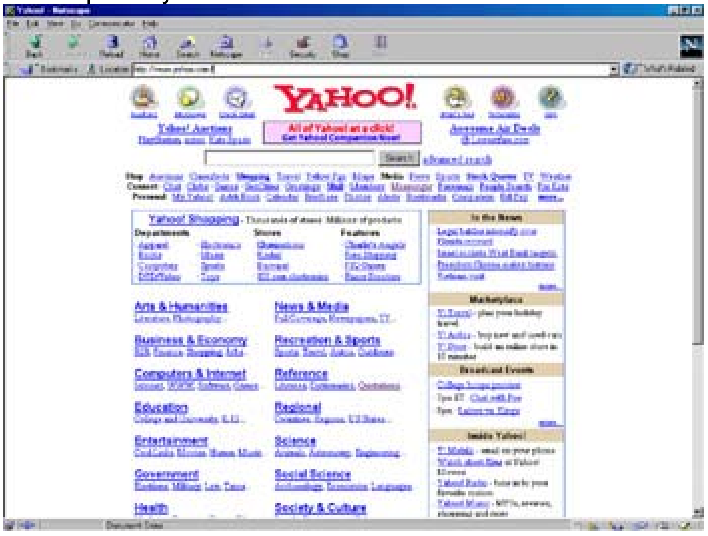


Figure 10.1: The Yahoo! Web site (`http://www.yahoo.com`)

Each major category contains dozens of additional subcategories and subcategories of subcategories, some more than a half-dozen levels deep. One quirk of the system is that all commercial sites—defined in its help files as a site that "sells something, promotes goods and services, or promotes a company that sells goods and services"—are listed under the Business and Economy section.

Early on, Yahoo! made the decision to keep a clear division between the commercial and noncommercial Web sites. However, cross-references to the commercial sites are available in the other areas, under a subcategory title called *Companies@*, as seen in Figure 10.2.

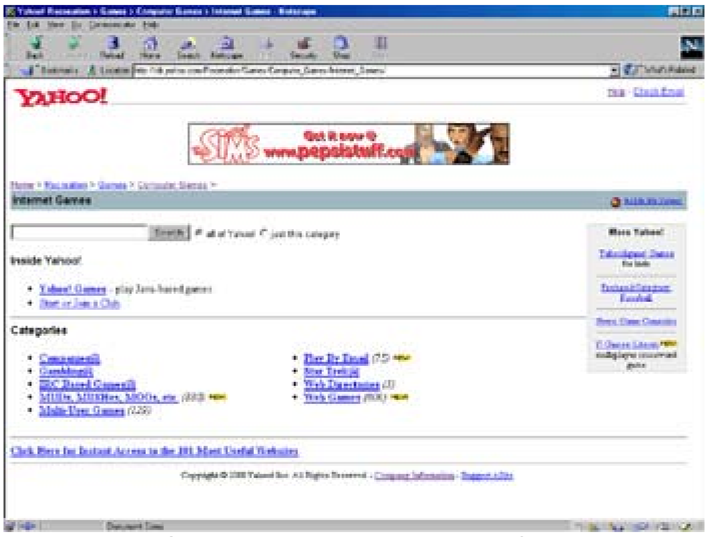


Figure 10.2: Commercial sites are cross-referenced in other major categories, under the Companies@ heading.

Fun Yahoo! Features

Looking for something new or nifty on the Web? Yahoo! has a handful of collections that can point you on your way:

- **Today's Web Events** Almost anything live on the Net is listed here, at `http://events.yahoo.com/`. Events can be as varied as professional-sports broadcasts via RealAudio, celebrity interviews in text-based chat rooms, and concerts in full video.
- **Weekly Picks** The Yahoo! surfers see a lot of sites during the course of their workdays. Each week they pick a few to shine the spotlight on. Look at their picks at `http://www.yahoo.com/picks/`. Want to have them delivered to you? Subscribe to their pick-of-the-week e-mail newsletter.
- **My Yahoo!** Want to build your own custom Web directory, get headlines, stock updates, and weather reports? Create your own Yahoo! here, at `http://my.yahoo.com`.
- **Yahooligans!** Yahoo! has a version for the preteen set, a collection of kid-friendly sites intended for Web surfers 7 to 12 years old. Visit it at `http://www.yahooligans.com`.

Searching on Yahoo!

Yahoo! offers several search methods:

- **Intelligent Default** The default is a standard keyword search. If more than one keyword is entered, sites that include multiple matches will be presented first. That is, if the words **Western United States** were entered, categories or sites that include *Western* and *United States* would be presented before categories or sites that only matched on *United States*.
- **Exact Phrase Match** Only categories and sites that match the keyword exactly will be returned. For example, an exact phrase entry of **art** would not return *arts* or *artistic*, as a default keyword search would.
- **Matches on All Words** Returns must include all words entered. The search method functions like the Boolean *and* operator.
- **Match on Any Word** A category or site will be returned if it matches on any one word entered. **United States** entered as a keyword could possibly return sites about the United Arab Emirates and the United Auto Workers.

Tip Keywords can be specifically excluded on Yahoo!. A search on California could be entered as ***California-Southern*** to exclude entries for Southern California. The minus sign (-) should be typed immediately preceding the word to be excluded.

To select a search option other than the intelligent default, click on the Advanced Search immediately to the right of the Search Submit button. You'll then see an expanded search interface, with radio buttons for each search type and select controls that allow for narrowing your search according to when a site was added to the database, for adjusting the number of returns presented per page, and for directing your search to Yahoo! or a record of Usenet newsgroup postings.

Submitting to Yahoo!

The process of adding a site to the Yahoo! database begins with a submission from a user, most often the Webmaster or an individual associated with the new site. To submit your site to Yahoo!, follow these steps:

1. Locate the most appropriate subcategory for your site within the existing Yahoo! categories. Remember that commercial sites must be placed within the Business and Economy section.
2. When you've located the subcategory that best matches your site, click on the Suggest a Site link at the bottom of the page near the copyright statement. You'll be taken to a Web page that asks you to confirm that you've selected the most appropriate category for a site that isn't currently in their database.
3. Click on the Proceed to Step One button to begin the submission process.
4. Enter a title for your site, such as Widgets, Inc.; the URL; and a brief description of the site. Yahoo! requests that your description uses 25 words or less, avoids repeating the title or URL, and is as descriptive as possible. Use the Submit button to proceed.
5. You can now suggest additional categories that would be appropriate for your site or even new categories that you feel would better serve sites like yours. Entries here are not required. When you are finished here, click the button to proceed.
6. Provide contact information as requested. Use the Submit button to proceed.
7. Fill in the blanks to describe whether your site deals with time-sensitive information. Examples of time-sensitive sites are those dedicated to a governmental election, an annual arts festival, or other activities that won't continue to be supported past a given date. You're also given the opportunity to make additional comments to the Yahoo! staff. Click the last Submit button, and you're done!

Now the process continues at Yahoo! Its staff of Yahoo! surfers will visit the site to confirm your choice of categories and to determine if additional placements are warranted. Because a real human reviews each submission, it can take some time before new sites appear in the database. A turnaround time of several weeks is not uncommon.

Warning You should be aware that Yahoo! reserves the right to decline submissions as well as the right to edit descriptions or change categories. Web designers should take care not to guarantee a listing on Yahoo! to a client or supervisor.

AltaVista

AltaVista, found at `http://www.altavista.com`, originated in 1995 as a project at Digital Equipment Corporation's Palo Alto Research Lab to index the entire Internet. An integral part of the success of the experiment was the development of Scooter, AltaVista's souped-up spider. More than just a robot following a single trail of links across the Web, Scooter was programmed to be *multithreaded*. Multithreading allows a computer program to tackle more than one task at a single time. Scooter's talents made AltaVista's collection of information about millions of individual Web pages possible in a very short amount of time. The AltaVista index now consists of over 350 million Web pages connected via powerful Digital AlphaServers.

The following table describes AltaVista's characteristics.

Table 10.2: AltaVista Site Details

DETAIL	EXPLANATION
Submission method	Via the form located at `http://www.altavista.com/cgi-bin/query?pg=addurl`
Robot name	Scooter
Indexing method	Indexes page titles, keywords, and descriptions provided through meta elements or the content of the page

Table 10.2: AltaVista Site Details

DETAIL	EXPLANATION
	itself
Adherence to Robots Exclusion Standard	Yes
Search methods	Natural-language queries, keywords, special search functions

Searching on AltaVista

AltaVista's index allows you to search through documents found on the Web or in Usenet newsgroup archives by using keywords from over 20 different languages. AltaVista recommends that you begin with what it calls a *natural-language query*, performed by simply typing in exactly what you're looking for. If you want to answer the question "What is the population of the United States?" just type that in and click the search button! If your question was phrased succinctly, the Web pages returned will likely be relevant, as demonstrated by the return for this question as displayed in Figure 10.3.

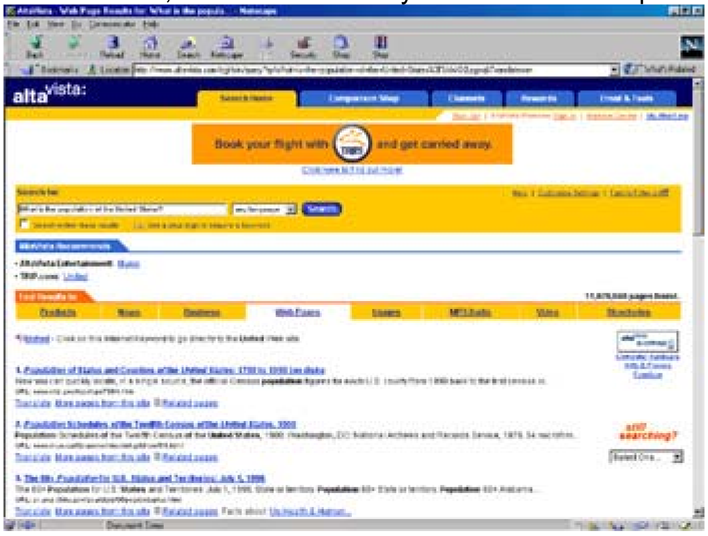


Figure 10.3: A natural-language-query response on AltaVista

A query using just the keywords *population* and *United States* produced identical results on the first page of returns.

Excluding or Including Search Criteria

You can require or exclude specific words by using the plus (+) and minus (−) signs. For example, if you wanted to search for information about World War II and specifically about the war as it pertained to France, you could enter **World War II +France**, which would require all returns to include France in the results.

Exclusion works the same way. To search for Web pages about dogs but not Chihuahuas, you would enter **dogs-Chihuahua** to exclude those short, nervous-looking animals.

Specialized Search Syntax

Just over a dozen special functions are available for searches. Sites can be located based on domain name, titles, pages that have *linked in* to a specific domain, and much more. Table 10.3 outlines the more commonly used functions.

Table 10.3: Special Functions for AltaVista Searches

FUNCTION	WHAT IT DOES
`link:`*`URL`*	Compiles all sites that contain hyperlinks to the specified URL. For example, `link:http://www.foo.com` displays all sites linking to `http://www.foo.com` or linking to any `www.foo.com` subdirectories and individual pages.
`image:`*`filename`*	Searches for images by complete filename (such as `logo.gif`) or searches for

Table 10.3: Special Functions for AltaVista Searches

FUNCTION	WHAT IT DOES
	characters included in a filename. An entry of `image:logo` would return matches on `logo.gif`, `logo.jpg`, or any other image filename containing the string `logo`.
`anchor:`*`text`*	Searches for instances of the specified string being wrapped in a hyperlink anchor. A search on `anchor:Click` here produces more than 130,000 matches!
`domain:`*`domainname`*	Locates Web sites within the desired domain. A search of `domain:fr` would return all French sites.
`title:`*`text`*	Hunts for Web pages with the specified text contained within their `Title` tags.

The Changing Face of Search Results

Over the past three years, nearly all search sites, whether index-based or directory-based, have moved toward a *portal* design. Portals are designed to be a Web surfer's first doorway, or "portal" to the Internet. Not only do they provide search facilities, but one-click access to shopping, news, Web-based mail, travel, entertainment, banking, auctions, and nearly any other service available on the Web.

The success of this portal strategy has been mixed. Yahoo! continues to dominate the field in size, breadth of services, and profitability. Other attempts have been less successful, leading to mergers, acquisitions, and even failures. The result is that the pioneers and the later bandwagon hoppers all tend to look alike, making it difficult for the consumer to determine the source and quality of the searching facilities available.

In the overall scheme of Web usage, whether a site delivers an index-based or directory-based search will not have a great impact on most users. What you should realize, however, is that each site will provide a different set of returns, and relying on one site to the exclusion of all others may result in missing the sites that provide just the information that you're looking for.

A Few Peculiarities

Here are a few practices peculiar to AltaVista:

- Keywords containing uppercase letters will be *case sensitive*. If you want case *insensitive* searching, use all lowercase letters.
- To return all forms of a keyword (discuss, discussion, discussed, etc.), use the asterisk symbol (*) as a wildcard entry (**discuss***).
- Enclose phrases in quotation marks if you want the phrase evaluated as a single word. For example, you could enter **"United States"** instead of **United States**.

Additional help on advanced AltaVista search topics can be found online at `http://doc.altavista.com/help/search/adv_help.html`.

Submitting to AltaVista

All submissions to the AltaVista index can be made from one location: `http://www.altavista.com/cgi-bin/query?pg=addurl`. The staff asks that you submit only one URL. Since Scooter the spider will be sent out to index your site and follow all the hyperlinks contained within it, it's not necessary to submit each individual page—Scooter will find them on its own.

Tip Additional tips and help for being "well indexed," as AltaVista calls it, can be found online at http://doc.altavista.com/adv_search/ast_haw_wellindexed.html.

More Popular Search Engines

Literally hundreds of search engines are available on the Web. Many strive to cover the entire Web as index-based sites like AltaVista do, whereas others concentrate on specific fields of interest, such as the Internet for the Fine Arts directory or geographical collections such as EuroSeek.
WWW To find the best places to list your new site, consider searching for additional search engines!
Other popular search sites include:

- Ask Jeeves, at `http://www.ask.com`
- Google! at `http://www.google.com`
- Lycos, at `http://www.lycos.com`
- Webcrawler, at `http://www.webcrawler.com`

Adding Search Capabilities to Your Site
Search-engine scripts for your own site have been available for some time, though they inevitably required programming expertise in Perl, or even C. Many Web designers begin their careers without the experience necessary to modify the scripts to their own needs. In response, the Web has seen a rise in search-service offerings from a variety of companies. Services range in price from free (in exchange for banner advertising on the results page) to monthly or annual subscriptions based on the size of the site. In this section, I'll visit two of the more successful contenders: Searchbutton.com and Atomz.com.

Searchbutton.com

Searchbutton.com offers services for personal sites, small businesses, and large corporations, with fees running from free to $5,000+. For the small site with less than 50 pages that don't change often, the free option would be viable, as they would only index the site once per month. If your site changes more frequently than that (and I generally recommend that it should), you might opt for the $19.95 monthly fee (or $199 annually) for up to 100 pages, or $29.95 monthly ($299 annually) for up to 250 pages.
The sign-up process for the Searchbutton.com service couldn't be easier. Simply enter the primary URL, your e-mail address, and a password, and you're on your way. You can have the XHTML for including the search functionality generated by their wizard, and you can see a preview on your Web site before you make any changes (see Figure 10.4). Additional options are provided for the look and feel of the results page, allowing you to choose colors for backgrounds, text, links, and more.

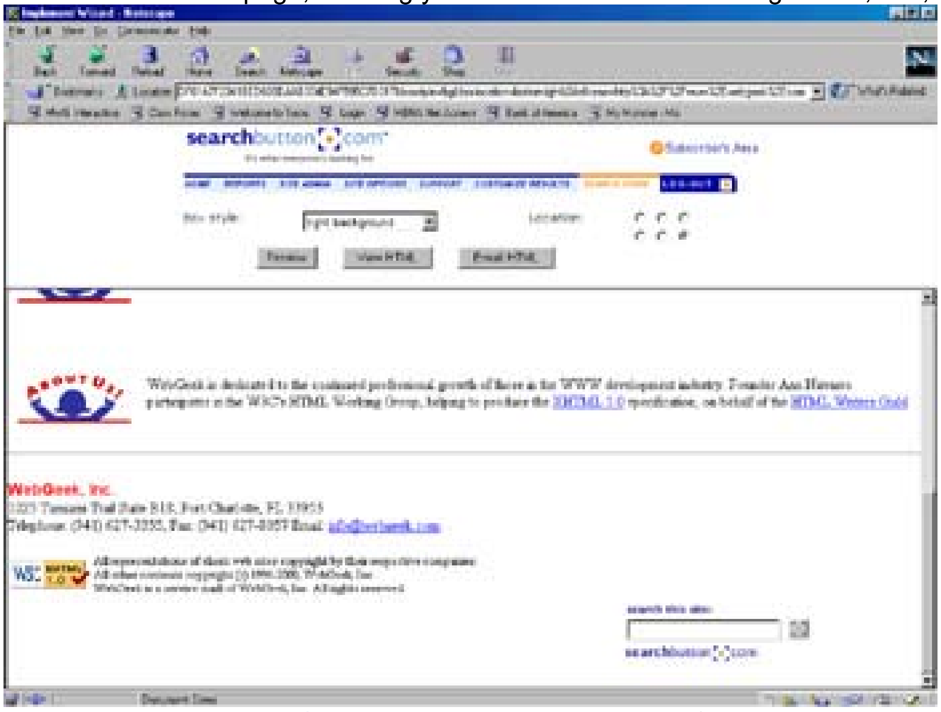


Figure 10.4: Previewing Searchbutton.com's results page (the page requires a little editing for alignment)

Paid Submission Services

With the explosive growth of Web-based search engines, submission "services" have sprung up all over the world. These companies, for a fee, will submit your site to a number of search engines for you, relieving you of what can be a time-consuming task.

Sounds great, doesn't it?

A number of legitimate businesses offer these services. Whether the cost of those services is a value to you is an individual decision. However, some companies make claims that would be very hard—nay, impossible—to accomplish, at any price.
Beware the company that claims to guarantee placement within a search engine or directory. As you've learned about Yahoo!, some services have staff review all submissions and reserve the right *not* to include sites. No company can guarantee inclusion.

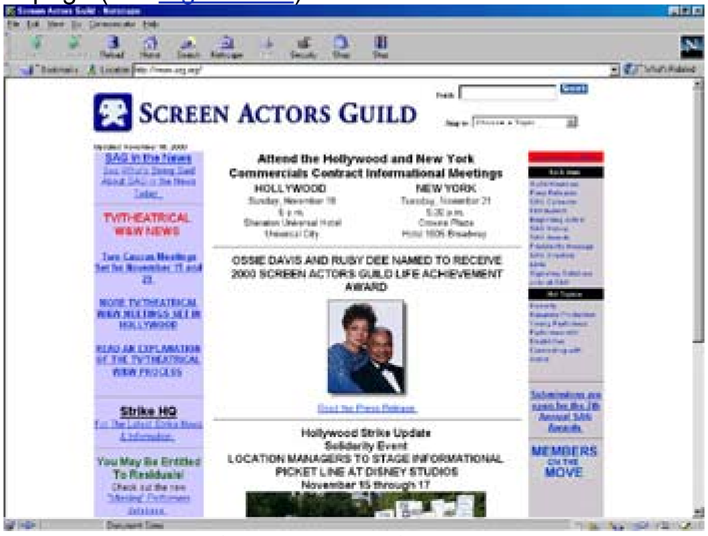
Also hesitate if a firm guarantees a high placement within a search engine. Web developers can increase their chances of being located and returned in relevant searches through the proper use of meta elements, carefully constructed site descriptions, and well-written XHTML. But no company can guarantee its customers will always come up in the top-10 or even top-100 sites returned on searches.

Atomz.com

Atomz.com also offers service for personal sites, small businesses, and corporate environments. The free service for personal sites offers more expansive services than Searchbutton.com, allowing up to 500 pages, weekly or on-demand indexing, templates for results pages (in addition to a full range of custom choices), and no advertising requirements provided you include the Powered by Atomz.com logo on your site.

Powered by Atomz.com

Business sites can subscribe to the service for as little as $25 per quarter ($75 annual option) for up to 250 pages, with pricing available for up to 5,000 pages before requiring a custom quote from the company. The paid service is less intrusive than either the free Atomz.com service or the Searchbutton.com design, as it looks like an internal component of most sites. Its clients include the Screen Actors Guild, the site of which provides search capabilities at the top-right corner of the home page (see Figure 10.5).



Attend the Hollywood and New York Commercials Contract Informational Meetings

HOLLYWOOD NEW YORK

OSSIE DAVIS AND RUBY DEE NAMED TO RECEIVE 2000 SCREEN ACTORS GUILD LIFE ACHIEVEMENT AWARD

Hollywood Strike Update
Solidarity Event
LOCATION MANAGERS TO STAGE INFORMATIONAL PICKET LINE AT DISNEY STUDIOS
November 15 through 17

Figure 10.5: The Screen Actors Guild uses Atomz.com search functionality.

Summary

A search engine is a Web site that provides links to other Web sites through a directory-style presentation of data or through keyword searching of an index or database. Hundreds of directories and search engines exist today, ranging from those that attempt to cover the entire Web to special-interest or geographically centered services. Sites are included in search engines through user submission or by robot data collection. A Robots Exclusion Standard was developed to allow Web-site owners to control robot access to their sites. Most search engines deploying robots have programmed them to respect the standard. Personal search services are available, allowing owners of Web sites to include search functionality within their site either free or for a fee, based on commercial or personal site functions and size.

Chapter 11: Validating Your Work

Overview

One of the most important concepts you can learn from this book is what makes XHTML *valid*. In Chapter 3, you learned about the DTD—document type definition—which sets the "grammar" rules for XHTML. Validation ensures that your markup hasn't broken any of those rules. Rather than simply being nitpicky, following all the rules for your selected DTD greatly increases your chances of the Web page being seen exactly how you intended by *as many* users on *as many* different platforms in *as many* different Web browsers as possible.

This chapter covers the following topics:

- What is a validator?
- Using the Kinder, Gentler Validator
- Using CSE HTML Validator Pro
- Interpreting the results

The Need for Validation

Remember the cross-browser, cross-compatibility concept introduced back in Chapter 1? Validation is compatibility in action. XHTML markup that doesn't conform to the rules set down in the DTD introduces an element of uncertainty into the display equation. That equation takes your markup, adds the Web browser's programmed response to expected markup syntax, and results in the final display. Factor in unexpected markup—the element of uncertainty—and the browser may not necessarily have a preprogrammed response ready. It must then begin to "think" on it's own and alter the display to how it has decided you really intended it.

Consider the results that spell checkers have within word-processing programs. Sometimes they suggest the word that you meant—and sometimes it seems they pull the suggestion out of a hat. Validation virtually eliminates the chance that the browser will need to guess.

With the advent of XML, you know that the requirements for well-formedness and validity are much stricter than they were for HTML. XML parsers aren't allowed to think their way through a malformed document; they instead halt on malformed documents. By learning to produce valid documents, you automatically produce well-formed documents and will not need to worry about parsers finding such fatal errors.

Two major types of validation services are available: *heuristic* validators and SGML-based validators. In this context, *heuristic* means that the validator searches through your XHTML files looking for errors and for constructs that are valid but considered poor form. SGML-based validators parse the files according to the strict rules of SGML (refer back to Chapter 3 under "The Document Type" for a discussion of SGML and its impact on HTML/XHTML).

Both types of validators have their advantages. Heuristic scanning can pick up omissions such as the lack of `height` and `width` attributes within an `img` element or the skipping from a level-1 heading `<h1>` to a level-3 heading `<h3>`. SGML-based validators don't make assumptions about your stylistic choices but do strictly interpret the structural integrity of your documents. In the next sections I'll look at three systems: the SGML-based W3C Validation Service and two downloadable software packages with heuristic components: CSE HTML Validator Pro and HTML Tidy.

Submitting an Error-Free Page for Validation

Of course, all designers hope that their XHTML is error-free from the start. I'll show the messages systems return for an XHTML page that's been written correctly.

The W3C Validation Service

The W3C Validation Service began life as the Kinder, Gentler Validator—or KGV, located originally at `http://ugweb.cs.ualberta.ca/~gerald/validate/`—and was created by Gerald Oskoboiny in August 1995 as a means for the lay HTML author to validate their markup and understand the results of the analysis provided. (The original location contains links to Oskoboiny's home page and the W3C Validation Service.) Previous tools based on SGML parsers provided error reports, but most of them did not have user-friendly interfaces.

Oskoboiny joined the W3C staff in September 1997 and continued work on the KGV project, resulting in the W3C Validation Service, which is now found at `http://validator.w3.org`.

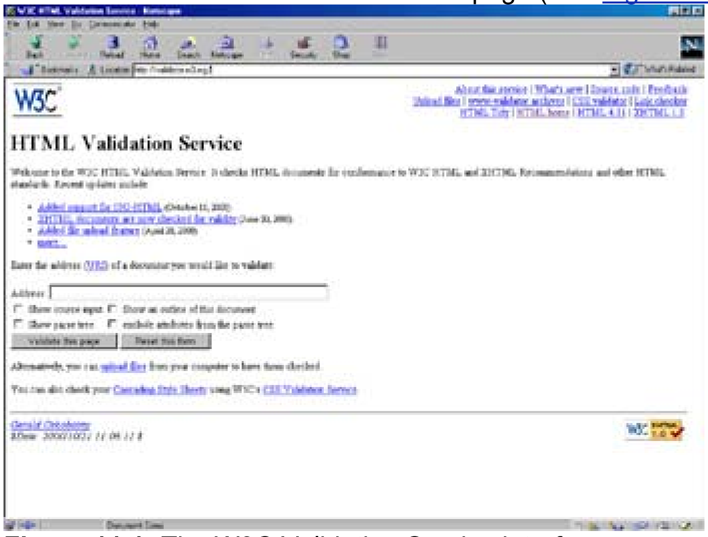
Listing 11.1 (also on the CD-ROM) shows a complete XHTML file that you saw in Chapter 6; it will be validated.

Listing 11.1: spacing.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Cellspacing</title>

</head>

<body>

<table border="1" cellspacing="20">

<caption>My Family</caption>

<tr bgcolor="yellow">

<td><strong>Name</strong></td>

<td><strong>Age</strong></td>

</tr>

<tr bgcolor="yellow">

<td>Dave</td>
```

```
<td>33</td>

</tr>

<tr bgcolor="yellow">

<td>Ann</td>

<td>34</td>

</tr>

</table>

</body>

</html>
```

In order to validate a Web page using the W3C Validation service, the page must either be available online so that the Validator's CGI scripts can retrieve it, or you may paste your markup into a form on the site. I've placed `spacing.html` online and submitted it for validation using the primary interface found on the Validator Service home page (See Figure 11.1).



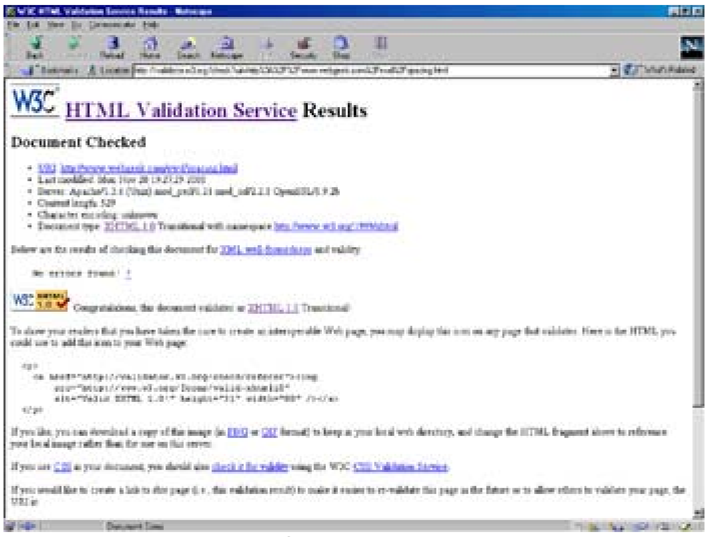
W3C

HTML Validation Service

[illegible]

Figure 11.1: The W3C Validation Service interface

The `spacing.html` file does indeed conform to the XHTML 1 Transitional DTD. If you submit the file for validation and it does conform, the service will present you with a page similar to the one seen in Figure 11.2, announcing that no errors are found on the page and congratulating you on the successful validation of the file. The Validator will also give you instructions for including on your page the XHTML 1 check-mark icon that the W3C makes available to sites that it has validated.


HTML Validation Service Results

Document Checked

[illegible]

Figure 11.2: A successful validation

CSE HTML Validator Pro

Now in version 4.5, the CSE HTML Validator Pro has long been a popular shareware product for Web developers. The interface of the CSE Validator is quite handy. When errors are found, they are highlighted in the document pane (see Figure 11.3). Each comment is listed on the Messages and Grouped Messages tabs, and the individual comments can be read in the comment pane to the right of the messages tabs. Later in this chapter, I'll examine how these features facilitate corrections.

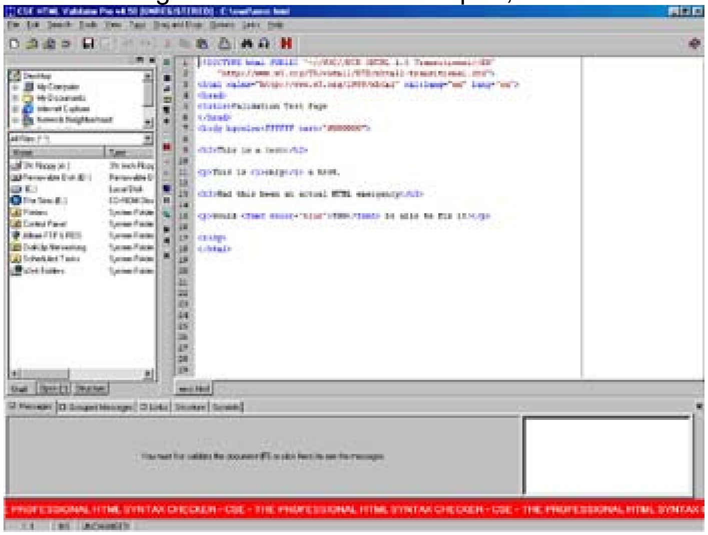

[illegible]

Figure 11.3: The CSE HTML Validator Pro interface (with a file loaded)

Interpreting Error Reports

In order to see how the validators respond to errors, consider a file, called `error.html` and shown in Listing 11.2 and on the CD-ROM, that contains a few basic errors.

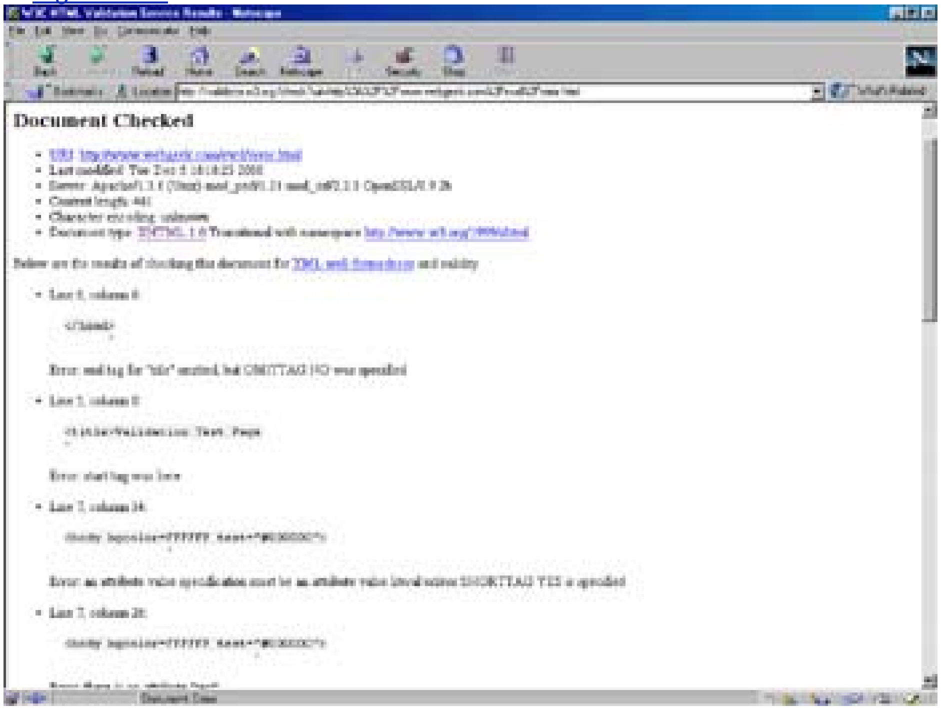
Listing 11.2: error.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Validation Test Page

</head>

<body bgcolor=FFFFFF test="#000000">
```

```
<h2>This is a test</h2>


<p>This is <i>only</i> a test.


<h3>Had this been an actual HTML emergency</h3>


<p>Would <font color="blue">YOU</font> be able to fix it?</p>


<body>

</html>
```

W3C Validation Service

When `error.html` is uploaded to the Web and submitted to the W3C validator, the results are seen in Figure 11.4.



Document Checked

- URI: [illegible]
- Last modified: [illegible]
- Server: [illegible]
- Content length: [illegible]
- Character encoding: unknown
- Document type: XHTML 1.0 Transitional with namespace [illegible]

Below are the results of checking this document for XML well-formedness and validity.

- Line 6, column 6:

  ```
  </head>
  ```

  Error: end tag for "title" omitted, but OMITTAG NO was specified

- Line 5, column 0:

  ```
  <title>Validation Test Page
  ```

  Error: start tag was here

- Line 7, column 14:

  ```
  <body bgcolor=FFFFFF text="#000000">
  ```

  Error: an attribute value specification must be an attribute value literal unless SHORTTAG YES is specified

- Line 7, column 28:

  ```
  <body bgcolor=FFFFFF text="#000000">
  ```

Figure 11.4: Error output produced by the W3C validator

The error-report language you see here is likely to be a bit different than what you're used to seeing on HTML 4 Web-page validation returns. You can see the difference in the first warning, a reminder that a closing tag is required for the `title` element:

Line 6, column 6:

```
</head>
    ^
```

Error: end tag for "title" omitted, but OMITTAG NO was

```
specified

Line 5, column 0:

    <title>Validation Test Page
    ^

  Error: start tag was here
```

The phrase *OMITTAG NO* refers to the validator's internal rules on whether an element is allowed to omit a closing tag. Since the well-formedness requirements of XHTML require that all elements be closed (either through a closing tag or the empty tag-closing shorthand), the fact that the `title` element didn't have a closing tag triggered the error. The validator further helps out by pointing you to the beginning of the element that's missing the closing tag.

The next two items pointed out by the validator both reference line 7, so I'll look at them together:

```
Line 7, column 14:

    <body bgcolor=FFFFFF test="#000000">
                  ^

  Error: an attribute value specification must be an attribute
value literal unless SHORTTAG YES is specified

  Line 7, column 26:

    <body bgcolor=FFFFFF test="#000000">
                              ^

  Error: there is no attribute "test"
```

The first error tells you, in rather cryptic form, that unless the attribute value is something very special, it must be enclosed within quotes (making the value a string, not a literal). The second error should be fairly obvious: the validator reports that an attribute "test" doesn't exist. Cursory examination reveals that the attribute should have been "text"—the validator can catch typos if they occur within element or attribute names.

With the `title` element, it's pretty easy to see where you began something. However, in complex XHTML documents, such as those with nested tables, it could be difficult to immediately pick out which tag wasn't closed properly. That extra hint from the validator becomes very helpful in those situations!

Next up, I have an error that at first glance looks pretty silly. Why wouldn't I be able to have an `h3` element where it is? Look:

```
Line 13, column 3:

    <h3>Had this been an actual HTML emergency</h3>
      ^

  Error: document type does not allow element "h3" here; missing
one of "object", "applet", "map", "iframe", "button", "ins",
  "del", "noscript" start-tag
```

When faced with an error that makes no apparent sense, the first place to begin searching in your code is the element immediately preceding the one noted in the error; in this case, it would be the following line:

```
<p>This is <i>only</i> a test.
```

What the validator is really trying to tell you is that the `h3` element cannot be contained within the `p` element, which hasn't been closed yet. Add the closing `</p>` tag, and the error referring to the `h3` element will go away.

Continuing down the list of errors, the validator also complains about the presence of the next paragraph:

```
Line 15, column 2:


   <p>Would <font color="blue">YOU</font> be able to fix
it?</p>
    ^


  Error: document type does not allow element "p" here; missing
one of "object", "applet", "map", "iframe", "button", "ins",
  "del", "noscript" start-tag
```

Remember, the last acceptable line to the validator was the `h2` element. It found a problem with the `h3` element, due to the missing closing `</p>` tag. It reported the error and then continued moving forward looking for that closing `</p>` tag. Instead, it finds a new opening `<p>` tag and again reports it as not being allowed to appear where it is. The error still traces back to the initial missing closing `</p>`. Had you revalidated the document after fixing the paragraph when the validator complained about the `h3` element, the second error would not have been re-presented.

Finally, the validator gets to the end of the document and reports the following:

```
Line 17, column 5:


   <body>
      ^


  Error: document type does not allow element "body" here


  Line 18, column 6:


   </html>
       ^


  Error: end tag for "body" omitted, but OMITTAG NO was
specified


  Line 17, column 0:


   <body>
   ^


  Error: start tag was here
```

Each of these items surrounds the same error. If you examine the original document, on line 17 you'll find `<body>`. An opening body tag certainly doesn't belong at the end of the document. Even though the validator is complaining that this body tag isn't closed, the problem really is that the first body wasn't closed—the document author simply did not insert the closing slash, as in `</body>`.

The remainder of the "errors" being reported refer back to issues already dealt with. The `html` element can't be closed until all other elements have been, and the validator points out where that `p` element began and was never closed. It then finally also points out that the original `body` tag wasn't finished either:

```
Line 18, column 6:

   </html>
         ^

  Error: end tag for "p" omitted, but OMITTAG NO was specified

  Line 11, column 0:

   <p>This is <i>only</i> a test.
   ^

  Error: start tag was here

  Line 18, column 6:

   </html>
         ^

  Error: end tag for "body" omitted, but OMITTAG NO was
specified

  Line 7, column 0:

   <body bgcolor=FFFFFF test="#000000">
   ^

  Error: start tag was here
```

Did the Kinder, Gentler Validator pick out all the mistakes in `error.html?` I'll take a look at what CSE HTML Validator Pro has to say about it.

CSE HTML Validator Pro

One extraordinarily convenient feature of the CSE HTML Validator Pro package is the ability to see the error reports *and* your XHTML document at the same time. Although it's easy to flip back and forth between a Web browser presentation of errors, which you'd have with the W3C validator, and the text editor in which you're editing your document, it's far easier to work with both sets of information within the same space. Figure 11.5 shows the results given by the CSE package for the same `error.html` file.

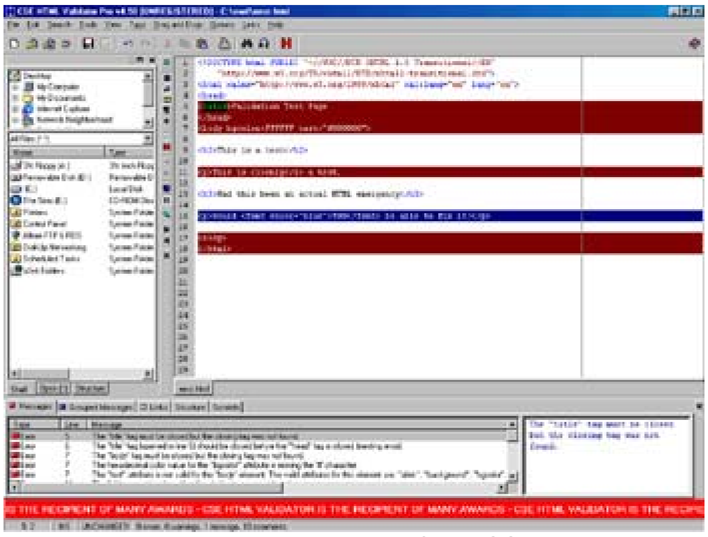


Figure 11.5: In a single visual interface, CSE HTML Validator Pro presents all the information needed to review errors and correct them.

The colored bars across the source text make a clear visual pointer to the problem area. The first error is even highlighted further with additional coloring and the cursor at the beginning of the problem element.

At the bottom of the screen are two panes, one on the left with a set of tabbed messages and the second a text pane that displays the message selected from the tabbed section.

On the Message tab, errors to be reported are marked with red boxes. The line number where the problem exists is given, along with a text description of the problem. You can read the message there on the tab, or over in the message pane to the right, whichever is more comfortable for you.
If you run `error.html` through the program and then scroll down on the Messages tab, you'll see additional items marked by a green square. These are informational messages or hints, not errors. Such is the function of a heuristic validator.

Under the Grouped Messages tab, you will find messages marked by blue squares; these are warnings about deprecated elements or attributes. As with the green items in the Messages tab, blue-square items aren't errors. They are hints that you may want to change the way you declared a particular property. The Grouped Messages tab also has green items, which are suggestions to enhance search-engine placement.

Strategies for Success

From the results seen using both the W3C Validation Service and CSE HTML Validator Pro, it's clear that a single tool won't always be the best way to look after your work. The heuristic tools provided by CSE HTML Validator Pro can pick up the stylistic issues regarding markup that, while technically valid, will create problems for some browsers.
Most Web designers make use of at least two validation tools to give themselves the most assurance possible that they've constructed quality Web pages. In addition to the two validators described in this chapter, quite a few more validators, both online and offline, are available for your use. A few more are listed in Table 11.1.

Table 11.1: Additional Validators

VALIDATOR	WHERE TO GET IT
Doctor HTML	`http://www2.imagiware.com/RxHTML/`
WebTechs HTML Validation Service	`http://cq-pan.cqu.edu.au/validate/`
Weblint	`http://www.weblint.org/`
Net Mechanic	`http://www.netmechanic.com/`
** WebTechs has suffered an unfortunate incident with its primary domain name and have at least temporarily lost ownership of it. The URL listed here is a mirror site. Should the primary*	

Table 11.1: Additional Validators

VALIDATOR	WHERE TO GET IT
site come back, the URL would be http://www.webtechs.com/html-val-svc/.	

Warning Not all of the online services have gotten up to speed with XHTML 1. It is my hope that by the time this book is printed, each of these resources will support XHTML 1 directly.

Summary

Validation is the process of insuring that XHTML markup was completed using proper syntax and style. SGML-based validators compare your markup to the allowable syntax defined in the DTD. Heuristic validators will also look for poor choices in style and markup that may result in inconsistent display across browsers. Many authors choose to use at least two validators to be sure to get a second opinion about their sites before going live.

Chapter 12: Knowing Your Audience

Overview

You may think it is a simple matter to know who visits your Web site. But defining your audience is harder than it sounds; the variables can be extraordinarily complex.

This chapter covers the following topics:

- Personalization
- Cookie security
- Writing cookies
- Browser support for cookies
- Privacy on Web sites
- Writing a privacy policy

Why Bother?

It's important that you think of your Web site as a valuable resource for distributing information to your audience. Your Web site must therefore target the people you want, especially if you are using it to promote a business. On the other side of the coin, you must know about who may happen to visit your site. Information you gather about visitors can range from the impersonal, such as whether a particular page is ever looked at, to the much more personal, such as shoe sizes. As you'll see, you can gather information in different ways; some methods are automatic, and some require audience participation.

Knowing your audience as well as possible benefits both you and the audience in the following ways:

- You can target your Web site toward the demographic you want.
- You can learn how people navigate your site and move pages accordingly.
- You can get valuable comments from people browsing your site, which allows you to correct confusing elements or update outdated information.
- You can ask your customers if they are interested in receiving further information; for example, you could add them to a mailing list to receive further announcements.

Personalization

Web designers who strive to have visitors interact with their sites on a regular basis almost inevitably offer some form of personalization. It can be as basic as username and password that is remembered for login or as complex as a complete rearrangement of the home page to suit the user's preferences. In this section, I'll look at examples from both extremes.

A basic personalization of login can be found at Travelocity (`http://www.travelocity.com`), an online travel-reservations service (see Figure 12.1). When setting up my user account with the site, I asked it to remember my login name. Using cookies, which are discussed later in this chapter, the site can remember who I am and greet me accordingly when I return.

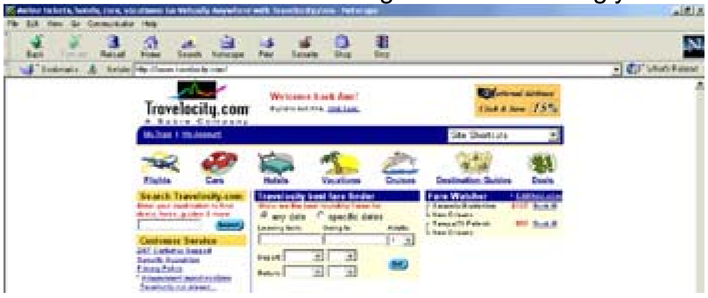


Figure 12.1: Personalization by easy login at Travelocity.com

A travel site that goes a bit further is Biztravel.com. Not only does it offer to remember my username but my password as well. Then I can set preferences that include my home airport and keep a database of my frequent-flyer and frequent-hotel-stay accounts so that appropriate notes can be entered when reservations are made. Although I dislike the framed display the Webmaster team switched to during the last year, for pure convenience and business-traveler support, Biztravel.com is a hands-down winner (see Figure 12.2).

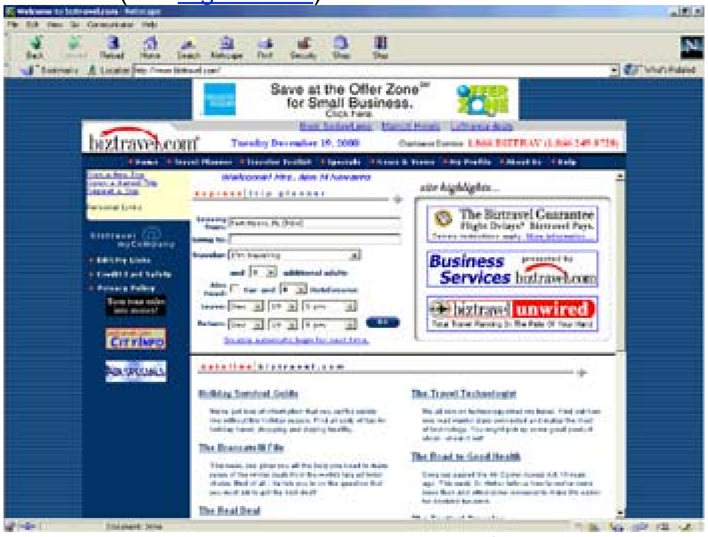

Save at the Offer Zone for Small Business.
biztravel.com
site highlights
The Biztravel Guarantee
Business Services biztravel.com
biztravel unwired

Figure 12.2: Additional personalization after login, including, for quick reservations, a preference for home airport

One of the most complex implementations of personalization can be found at Yahoo!. To get started, click the Personalize icon in the upper icon bar across their main page. Because I've already personalized my page, the layout I see is the result of my choices. I also get a greeting at the very top of the page (see Figure 12.3).

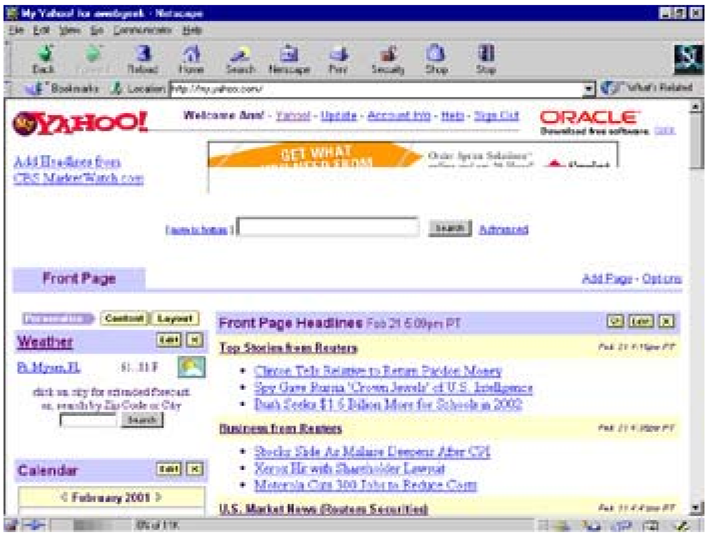
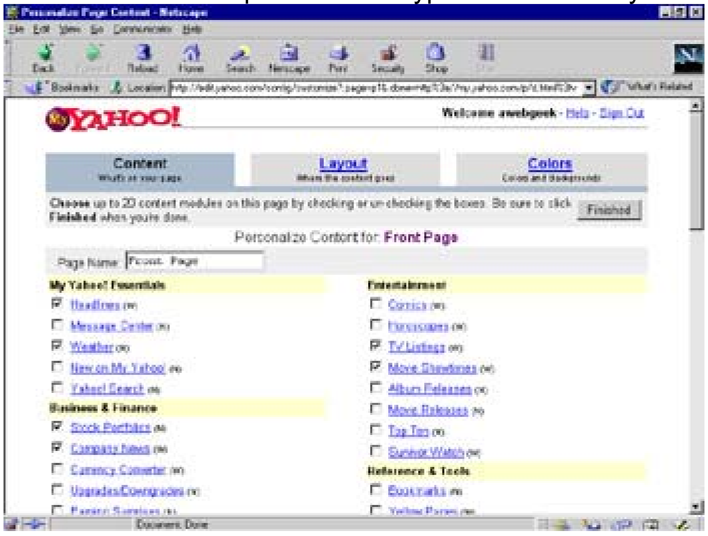


Figure 12.3: A Yahoo! greeting on My Yahoo!

To change the layout for this *front page*, as Yahoo! calls it, use the content and layout buttons found under the Front Page heading. If you start with content, you'll find several dozen options available (see Figure 12.4), each with a (W) or (N) after the description, which stands for a wide or narrow module. You'll need to keep the module type in mind when you choose your layout.


Personalize Page Content - Netscape
Welcome awebgeek - Help - Sign Out
Content
Layout
Colors
Finished

Choose up to 20 content modules on this page by checking or un-checking the boxes. Be sure to click **Finished** when you're done.

Personalize Content for: **Front Page**

Page Name: Front Page

My Yahoo! Essentials
- [✓] Headlines (W)
- [ ] Message Center (N)
- [✓] Weather (N)
- [ ] New on My Yahoo! (N)
- [ ] Yahoo! Search (N)

Business & Finance
- [✓] Stock Portfolios (N)
- [✓] Company News (W)
- [ ] Currency Converter (W)
- [ ] Upgrades/Downgrades (N)

Entertainment
- [ ] Comics (W)
- [ ] Horoscopes (N)
- [✓] TV Listings (W)
- [✓] Movie Showtimes (W)
- [ ] Album Releases (N)
- [ ] Movie Releases (N)
- [ ] Top Ten (N)
- [ ] Survivor Watch (W)

Reference & Tools
- [ ] Bookmarks (N)

Figure 12.4: My Yahoo! content options

When you're finished choosing content, click the Layout tab to arrange the modules you've just picked. You have a choice between a two-column and a three-column layout; a three-column layout lends itself best to users who have mostly narrow-module content and only a few wide-module selections. Finally, the Colors tab allows you to create the look and feel of your page from several predesigned selections or from a fully customizable option (see Figure 12.5).

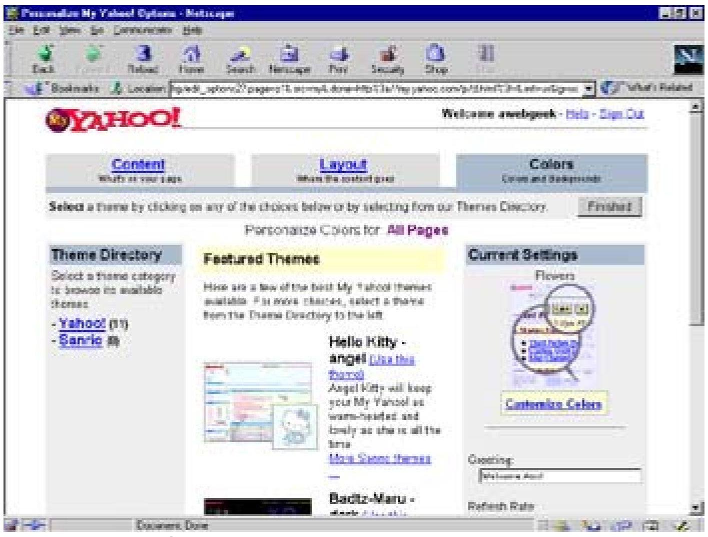

Personalize My Yahoo! Options - Netscape
Welcome awebgeek - Help - Sign Out
Content
Layout
Colors
Select a theme by clicking on any of the choices below or by selecting from our Themes Directory.
Finished
Personalize Colors for All Pages
Theme Directory
Featured Themes
Current Settings
Flowers
Yahoo! (11)
Sanrio (8)
Hello Kitty - angel
Customize Colors
Badtz-Maru -
Greeting:
Refresh Rate
Document: Done

Figure 12.5: Colors and designs available to liven up your personalized Yahoo!

The user's custom-option data is too large to transfer to Yahoo! every time the user logs in. Instead, that information is stored in a database on Yahoo's system and displayed when the user returns with the same ID stored in what is called a *cookie file*. How these cookies (IDs) are set and managed is discussed in the next section.

Tip Other than the basic method for setting cookies, the programming needed to fully personalize sites is beyond the scope of this book. It generally requires dynamic page generation or other database-backed solutions.

Using Cookies Effectively

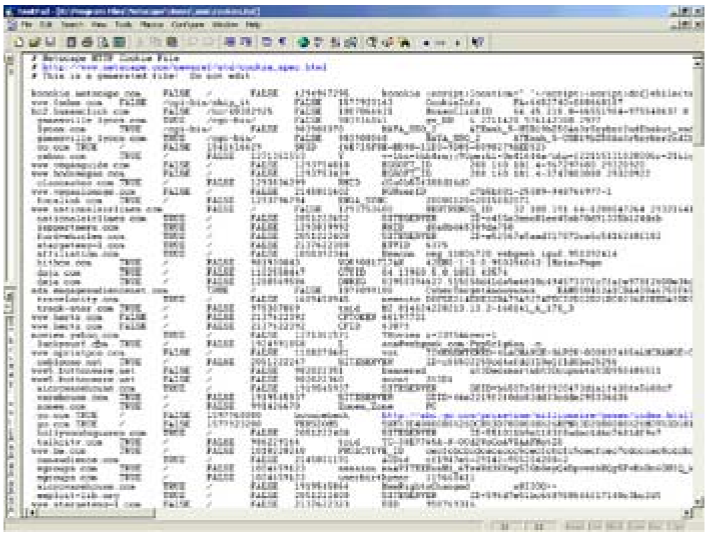
Rather than a tasty treat stored in a jar on the kitchen counter, *cookies* on the Internet refer to bits of information written, by the visited site with the help of the browser, to a file on the user's computer. These bits of information allow the site to manage user preferences, keep track of items placed in a shopping cart, or perform other activities that often relate to maintaining *state* across more than one Web page. State is a record of current choices, settings, or other information that traditionally cannot be passed from one Web page to another.

The Web is inherently *stateless* because the connection between Web browser and Web server is severed after each page is delivered. When the visitor clicks a link to move on to another page or to perform an action such as placing a product in a shopping cart, the Web server has no nascent knowledge of whom that user is. Cookies allow the server to keep track of who's who, even if only labeled with a number, and to manage the broader user interaction across the site.

Peeking Inside the Cookie Jar

The name and location of the file used to store cookie information will vary based on which browser you're using. Netscape Navigator users will typically find theirs in `X:\Program Files\Netscape\Users\`*`Username`*`\cookies.txt`, where `X` is the drive on which Navigator was installed, and *`Username`* is the name given to the individual user profile. In my case, it was found on my laptop at `D:\Program Files\Netscape\Users\ann\cookies.txt`.

The bits of information stored in this file aren't necessarily meaningful when read directly from the file, but they do have meaning to the server that placed them there. Figure 12.6 shows my cookie file generated by Netscape 4.76. My file has 277 unique entries in it, so you can't see it all, but what you see here is a typical sampling of what your file may look like.



```
# Netscape HTTP Cookie File
# [illegible]
# This is a generated file!  Do not edit.

[illegible]
```

Figure 12.6: A sample cookie file (as viewed in a text editor)

Warning Although you won't kill your browser by editing your cookie file, you can disturb personalizations and other state information stored there by Web sites that you frequent. So take heed of the warning at the top of the file and resist the urge to edit it.

In Figure 12.6 you'll see, near the bottom, a line that begins with `movies.yahoo.com`. Table 12.1 below dissects what each field of the cookie stores, using that entry as an example.

Table 12.1: Dissection of a Cookie

FIELD	WHAT IT MEANS	VALUE IN THE EXAMPLE
`domain`	The domain name of the site that created the variable defined in the cookie is stored here. Only that site can read the variable.	`movies.yahoo.com`
`flag (TRUE/FALSE)`	The true/false value refers to whether or not all the machines within the domain can access the variable. This means that `www.netscape.com` and `home.netscape.com`, for example, can both access the cookie.	`TRUE`
`path`	The `path` field holds the path for which the variable is valid. Try thinking of it as a directory structure. The slash (/) means that the cookie is valid for all the pages in the domain.	`/`
`secure (TRUE/FALSE)`	This is another flag.	`FALSE`

Table 12.1: Dissection of a Cookie

FIELD	WHAT IT MEANS	VALUE IN THE EXAMPLE
	It tells whether or not the connection needs to be secure in order to access the variable. An example might be if the variable is a credit-card number.	
`expiration`	The server can set the expiration date for the cookie. It's later converted from GMT into Unix time. Unix time is the number of seconds since 12:00 AM on 1/1/70. Using Unix time, this cookie will expire well after the year 2001.	`1271361571`
`name`	The `name` field holds the name of the variable. It is set by the programmer and is hopefully descriptive. In this case, it stands for Yahoo! Movies zip code.	`YMovies z`
`value`	The `value` field contains the actual value of the variable. My zip code when the query was made of the movies database was 33954. The ver=1 attribute has some meaning to its programmers, but it's not immediately apparent.	`33954&ver=1`

If I were to revisit the Yahoo! movies site, it would remember the zip code I last entered when searching for theatres and present me with the same information. This is a convenience to me, and it gives up very little privacy.

Other sites noted in my file have stored a username, an indication of whether my last visit was a first visit, whether I'd answered a survey, my IP address, and, in one rather disconcerting instance, both a username and password to a site designed to manage secure merchant details. The last instance was troublesome in that the site left a human-readable record of my account information available to anyone who could access my computer system—be it an employee or hacker. Banking details should not be that accessible. (The company in question has since been contacted to correct that practice.)

Cookie Security

Much concern has been expressed over the years about cookies, mostly because sites are writing data directly to the user's hard disk, leading to the question of what other portions of the system a Web server may access. There have also been concerns about whether Web servers can review and retrieve information left in the `cookies.txt` file by other Web servers.

First I should clarify how data is written to the cookies file. Servers don't actually have access to your system. They only pass the information to the browser, which then writes it to the cookie file. Although more than one file may be associated with cookies, they are all text files. Text files are harmless to computers; they cannot contain viruses. The only information servers can pass to the browser for recording is either supplied by the server (such as the expiration date of the cookie) or by the user (such as the zip code in the previous example). In most cases, if the user supplies any information, it is the value of the variables. The cookie cannot contain any personal information unless the user directly supplies it.

Second, I need to examine the sending of data from the hard disk back to a Web server. The browser will only send data stored in the cookies file back to the server. It does not scour your system looking for configuration information or other personal data. Additionally, when the browser receives a request from the server for cookie information, it must compare the domain that request originated from with the domain already written in the cookie file. If they do not match, the data will not be sent.

Writing Cookies

Writing cookies takes some knowledge of programming, which is mostly beyond the scope of this book. Various programming languages can be used, from relatively simple scripting languages like JavaScript to the more complex Perl scripts and CGI scripts written in C or by middleware products that interact between the server and the Web browser.

The most common way scripts and programs add a cookie is to insert a *`Set-Cookie` header*, which means that the server sends additional information to the client when the client makes a `get` request. A generic `Set-Cookie` header would look like this:

Set-Cookie: NAME=*VALUE*; expires=*DATE*; path-*PATH*;

  domain=*DOMAIN_NAME*; secure

The programming part comes in when this information is sent to the client, because it is not sent manually. Usually, a program is written (like a CGI script) that automatically gathers the information and sends it. For example, a `Set-Cookie` header from WebGeek might be:

Set-Cookie: NAME=user; expires=Friday, 30-Jan-01 12:00:00 GMT;

  path-/; domain=webgeek.com; secure

A separate program is used to handle the cookie when the user returns to the site. When a user with the same browser (using the same profile) comes to the WebGeek site, the browser checks the URL that the user has chosen (`webgeek.com`) and tries to match it to the different URLs in the cookies file. If the user has come to the site before and a cookie was set, then the browser will locate that cookie and send it to the server so that the server will have the cookie information readily available. How the server uses the cookie information depends on the contents of the cookie. One of the most common ways is an auto-logon feature that allows the Web site to remember the visitor.

Browser Support for Cookies

Out of the browsers I have focused on in this book, only Lynx doesn't fully support cookies. Netscape Navigator began the trend by introducing and supporting cookies with version 1, and they are supported in all subsequent versions. IE also supports cookies. As newer versions of browsers have been released, the user has been given more and more control over the treatment of cookies. For example, the earliest versions of Navigator did not give the user any choice—all cookies were accepted. In the latest version, users can decide if they want to be warned before accepting a cookie, to reject all cookies, or to accept all cookies. They can even decide if they only want to accept cookies that are sent back to the original server. The latest version of IE offers users a similar set of choices.

Some people have gone so far as to recommend making your `cookies.txt` file read only, preventing any new data from being written to it. Take this step lightly, as the valuable conveniences of cookies will disappear along with the uses meant for marketing.

For the purpose of Web design, realize that users may choose to browse your Web site without cookies. If your site is written to *depend on cookies* for proper functioning, you will have some unhappy users. When planning for your site, covered in detail in Chapter 13, be sure to keep this in mind.

Sending Cookie Data to Other Servers

The server that has written a particular cookie or a server *specified by that server* can read the cookie data. But cookies are still very secure because most Web-site operators do not want other sites to know what data they're collecting or storing. The ability to specify access by other servers is almost exclusively used for marketing purposes, when a Web site might collect cookie data and send it to a marketing firm rather than analyze it in-house.

Users were becoming annoyed that their cookie data was sent directly to marketing firms, so Netscape added the option of only accepting cookies which were set to be returned to the originating server. The change hasn't really fixed the problem because now marketing firms simply set their servers to have the same domain name as the site the user is on. That is, the user may visit the domain `foo.com`, and the server that gathers the cookie data for the marketing firm might be `marketing.foo.com`. The people at `foo.com` don't really have anything to do with that machine; they have only given permission to the marketing firm to use that name. Provided that the cookie is set up to be valid for the path /, the user's machine will accept all of the cookies as if they were returning data to the original server. Please see Table 12.1 if you don't remember how the path works. Cookies can be disabled in ways other than simply turning off the preferences.

Privacy Issues

I touched briefly on privacy issues when discussing what information is stored in cookie files. But the concerns about privacy on the Web reach far beyond cookie technology. Consumers have become savvy about personal data-collection practices, in large part through publicity in the traditional media (print and television). Inevitably the most sensational stories surround people or sites that have suffered a disastrous loss of privacy. People hear of individuals who have had their identities stolen and been subjected to huge debts run up online by a thief or companies whose sites were broken into by malicious hackers who took confidential customer information, including credit-card data.

Although the Web-surfing public needs to be aware of the extreme dangers—just as everyone needs to be aware of the possibility of being struck by lightning or being on an airliner that goes down—the risk of being victimized by such an online invasion of privacy is nearly nil. Instead, the average user should focus on what a Web-site owner will *do* with the information the user provides. To determine this, users should first turn to a site's *privacy statement*.

Sites that have formed such a policy will nearly always provide a link to it from the home page. In the case of Yahoo!, it's the very last item at the bottom of the page. You'll be taken to an extremely comprehensive Privacy Center detailing the collection and use of data for just about any service Yahoo! offers, links to privacy resources, and a special privacy section regarding information collected from children (see Figure 12.7).

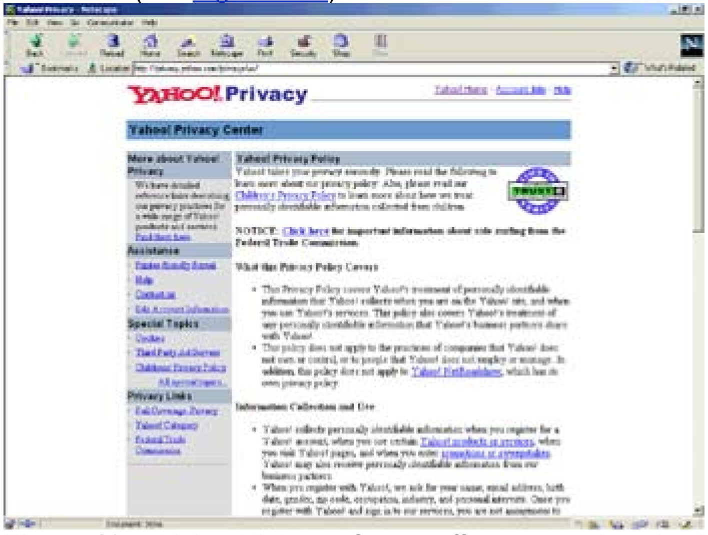


Figure 12.7: Yahoo! Privacy Center offers comprehensive information.

TRUSTe

TRUSTe is an online certifying authority that affirms that sites have provided a privacy policy in line with their actual practices and that the site conforms to all TRUSTe requirements. As an independent third party, the organization's seal of trust can give users a feeling of confidence.

However, sites that do not carry the TRUSTe seal are not necessarily unscrupulous or careless with your personal information. Small companies may find the minimum certification fee of $299 annually too expensive or may think that their own published policies and practices fulfill the same role.

WWW Whether or not you choose to apply for the seal, the TRUSTe site provides helpful resources for writing a privacy policy that conforms to its requirements. The sample may be found online at `http://www.truste.org/webpublishers/pub_modelprivacystatement.html`.

Warning If you do not intend to engage TRUSTe's services, be sure to remove all information in its sample policy that refers to the company. You may only refer to TRUSTe or display its seal by contracting with the organization for services.

TRUSTe suggests 17 specific areas that your privacy policy should cover. I will discuss them here.

Information Collection and Use

A traditional assertion is to say that the company owning the site becomes the owner of all information collected on the site. If that information will be shared with others, including any subsidiary or parent companies, that must be disclosed here. A blanket statement affirming that no usage will occur other than stated elsewhere in the policy is also advisable.

Registration

If users are required to register to access all features of a Web site, the owner must state which pieces of information are required and how that information will be used. The minimum is usually a name and e-mail address, used to contact the visitor about the site's services, products, and offerings they have expressed an interest in. Any additional demographic information collected should be labeled as optional; when collected, the owner should express its eventual use (personalization, generation of audience demographic reports, etc).

Ordering

You should list information collected during the course of a purchase or other ordering transaction. Telephone numbers or e-mail addresses should be used only in support of fulfilling the order in case of problems. If the information may be used in another manner, it must be explicitly stated.

Cookies

This section of the statement should list in detail when and under what conditions cookies will be set. It should also explicitly state which portions of the site may not function properly without them, such as the ordering portion, a message-board area, or site personalization.

If third parties, such as an ad-banner rotation service, add content to your site and they set cookies, a statement about that activity is required. A pointer to that service's privacy policy is helpful, though care should be taken to indicate that you have no control over their usage of information.

Log Files

Discuss in this section any analysis you perform on your log files, such as tracking user movements on the site by IP address for aggregate user behavior.

Warning The sample policy provided by TRUSTe indicates that IP addresses are not linked to personally identifiable information. Be very careful if you assert this. Many shopping-cart and e-commerce packages record the IP address in use at the time of a transaction. By performing a reverse DNS lookup, the site owner can determine what domain the IP address belongs to and, in certain cases, the specific computer to which it has been assigned. If credit-card transactions are challenged by the cardholder, the IP information can help determine if the purchase was legitimate.

Sharing

Any information that will be distributed to any other party, including subsidiaries, parent companies, partners, drop-shippers, credit-card processors, banks, or any other third party must be disclosed here. Which information and under what circumstances it is released should be discussed. Any restriction the site owner places on the use of the information by the third party should be noted.

Links

This section disclaims liability for the privacy practices on sites linked from the owner's Web site.

Newsletter

If the site runs a newsletter, the information used to manage subscriptions is detailed here, and any use outside the management of that subscription must be disclosed. If the newsletter subscription list is sold, rented, or otherwise shared, these details should be made clear, even if the information is covered again elsewhere (perhaps in the Sharing segment).

Surveys and Contests

If your site conducts surveys or giveaways and other contests, the data that would be collected should be labeled as an optional collection, provided that it is not required contact information. Contact information is normally used for the distribution of prizes and other contest-related communication. Survey opinions are used for maintenance, upgrades, and content changes for the site, unless otherwise stated. Sharing should be disclosed, especially when drop-shipping or other fulfillment services are used to deliver physical prize products.

Tell-a-Friend

Some sites attempt to gain grass-roots publicity by making it easy for visitors to tell friends about their site, generally by letting the visitor provide the friend's name and e-mail address. What occurs next should be spelled out. Will the friend get a one-time-only e-mail that says a friend recommended the site? Will the friend receive multiple mailings if they don't respond by visiting after the first message? Options for removal of the friend's name should be given in the e-mail.

Site Security

The security section should discuss not only the security in place when users enter sensitive information, such as during the use of SSL (secure socket layer) and when the indicators of such status (the padlock icon in most browsers) are visible, but also where the company will store information once the transaction is complete. Is it stored in a database? In a locked filing cabinet? Which employees have access to the information? Which information is kept in-house and which remains with third parties, like real-time merchant processors?

Supplemental Information

Certain categories of Web sites may need to compare user-supplied data with information retrieved from outside sources. A typical example is a company, such as a bank, that grants credit privileges to users, or other major retailers that provide credit cards or purchasing accounts. Users opting to apply or make use of such services agree that the personal information provided may be compared to other information and reported to such third parties as disclosed in the terms-of-use agreement.

Special Offers

If introductory e-mail is sent to each user, the specifics should be listed here. If additional mailings are to be a normal part of participating in the site, opt-in/out choices should be made available at the time of registration, or full details on opting out should be given in the welcome message (the former being preferable).

Site and Service Updates

Notification to registered users about site and service updates (or planned downtimes) may occur. If your site notifies users via e-mail, postal mail, or telephone, the method must be disclosed. If you believe these notifications are important enough that all users should receive them, you must state that the user may not opt out of such notifications.

Correcting/Updating Personal Information

If a site collects personal information, some means for review and correction or removal of that information must be made available to the user. You can provide such options through an online form, a customer-support address, a telephone number, or a postal address.

Opt-In/Out Choices

A full disclosure of information collection, sharing, and usage should be made at each point where information is requested. For instance, when registering as a user on a site, the visitor should be given the choice of opting in or out of a weekly newsletter, sharing information with partners and affiliates, or other exchanges of information. Privacy proponents suggest setting the defaults to opt out, a position that customer-satisfaction surveys suggest results in fewer unhappy responses to additional contacts. Of course, the growth of user information may slow considerably. Each site owner must weigh the pros and cons.

Notification of Changes

Changes to a privacy policy should always be made clearly available on the primary Web-site page. If changes are made as to how personal information is handled, especially when the new policy is more open rather than restrictive, the user must be given the choice to opt out of the new terms of use. Where possible, the user should be able to access portions of the site while still being covered under the previous privacy terms. The ramifications of not accepting the new terms should also be made plain.

Summary

You don't have to be completely in the dark about who's visiting your Web site and what they are doing there. The first step to knowing your audience is to personalize it for each visitor. Then you can use cookies to store and track user movements and preferences and to provide convenient features to the user. Cookies have many myths surrounding them; to help combat these problems, a comprehensive privacy policy should be developed by all sites that collect information from the user.

Chapter 13: Planning the Design of Your Web Site

Overview

With your XHTML skills honed, you're anxious to get started building your Web site. But where do you start? In this chapter, you will learn the steps necessary to design an effective Web site.

It's time to push back from the computer, pull out a pad of paper and pen, find yourself a comfortable chair, and roll up your sleeves for a serious analysis-and-planning session. Design is a step-by-step process where you choose what materials to present, how they should look, and how to structure your Web site.

This chapter covers the following topics:

- Identify, identify, and identify some more!
- Developing a look and feel
- Establishing design constraints
- Planning your Web site
- Tying it all together

Identify, Identify, Identify!

Effective design results in the successful communication of your message to your intended audience. Though it sounds simple, consider that your message will be one of many messages available on the Web. People are saturated with information from the Web, e-mail, postal mail, newspapers, commercials, billboards along the side of the road, and even the back of the morning milk carton! How do you make your message the one that will get your audience's attention and response?

The first step to an effective design is to identify the following three points:

- What do you want to say?
- Why is your information needed?
- What do you want people to do as a result of your message?

When you answer these questions, you will be well on your way to developing an effective design that will accomplish your mission—to communicate your message.

What Do You Want to Say?

Identifying your message may sound obvious, but many Web sites fail on this point. How often have you visited a Web site that was confusing in its message? You weren't sure if the site was promoting a product, trying to inform you, or just wasting your time. How long were you willing to stay and figure out what the author was trying to communicate? Did you ever go back to that site again? Most importantly: Did you respond to the Web site in any tangible way? Did you e-mail or phone for more information? Did you take out your credit card and buy anything?

When people visit your Web site, they need to be able to quickly recognize your message, be interested enough to stay and learn more, and respond. By identifying what you want to say, you're on the way to getting your message delivered.
Web sites usually serve one or more of the purposes listed in Table 13.1. Use the following table to help identify your purpose, step one in developing your Web-site plan.

Table 13.1: Common Purposes for Web-Site Owners

PURPOSE	TYPICAL WEB-SITE OWNERS
To inform or educate	Universities
	Schools
	Charitable foundations
	Nonprofit organizations
	Government
	Businesses
	Political organizations
	Users with personal homepages
To entertain	Magazines
	E-zines
	Galleries
	Museums
	Media clubs
	Community organizations
	Users with personal homepages
To market, sell, or persuade	Businesses

Table 13.1: Common Purposes for Web-Site Owners

PURPOSE	TYPICAL WEB-SITE OWNERS
	Political organizations
	Nonprofit organizations
	Universities
	Schools
	Users posting personal resumes

It's very likely that you will have several purposes; in fact, most Web sites serve at least a couple of purposes. "To inform people about my company and to persuade them to purchase my services and products" is a typical statement of purpose.

Now it's time for that pad of paper. Jot down what you want to tell your audience. It doesn't have to be a lengthy description; in fact, a simple sentence describing your purpose would be ideal. If you need to write multiple sentences, that's fine.

Tip Nonprofit organizations usually have multipurpose Web sites. Typically, a nonprofit has a multipurpose mission statement that includes a requirement to disseminate information about its purpose, solicit memberships and donations, and offer other opportunities to its membership and the public. Although the multiple purposes make for a complex site plan, prioritization of the purposes will simplify the project.

Why Is Your Information Needed?

The key to capturing your audience's attention is to understand why they need or want to know your information. By focusing on the audience's needs, you make the planning process easier for yourself, and the resulting Web site will be more likely to interest your audience.

Make a list of all of the reasons that someone would want to learn more about your message. For example, will it:

- Solve a problem?
- Make them feel good?
- Get them involved?
- Tell them something they don't already know?
- Save their life or their families' lives?
- Give them an opportunity to realize a dream?
- Show them a new way to do something?
- Raise their consciousness?
- Expand their horizons?

The list isn't meant to be all-inclusive but is a starting point for your own list. Remember, your audience is bombarded with demands for their attention and at best is indifferent to your message. Recognize their indifference and analyze their need for your message.

Tip If it's possible, as you jot down the reasons your audience needs to know your message, prioritize them. Perhaps your product will solve a problem, is good for their health, and also costs substantially less than your competitor. Which selling point is most likely to get your audience's attention? Prioritization now will make your job easier later when you develop a site structure.

Now that you have identified what you want to say and why it is important to your audience, you're ready to determine what you want your audience to do when they receive your message.

What Do You Want People to Do?

For a moment, assume you have designed this great Web site and people are flocking to it and getting excited about your message. Quick! They are focused on your message—here's the opportunity to have them act. But you need to decide what you want them to do. Grab that pad of paper again for more lists.

Here are some of the possible actions your visitors could take:

- Send you e-mail for more information

- Fill out an order form
- Phone you to purchase a product or service
- Complete a survey or application form
- Write a letter
- Call their political representative
- Vote
- Volunteer
- Join a mailing list
- Send fan mail
- Return to your Web site on a periodic basis

It's extremely important to identify what you want your audience to do. It's likely that you want people to take several of the above actions as well as others not listed. Be sure to note all desired outcomes and, if possible, prioritize them in your list.

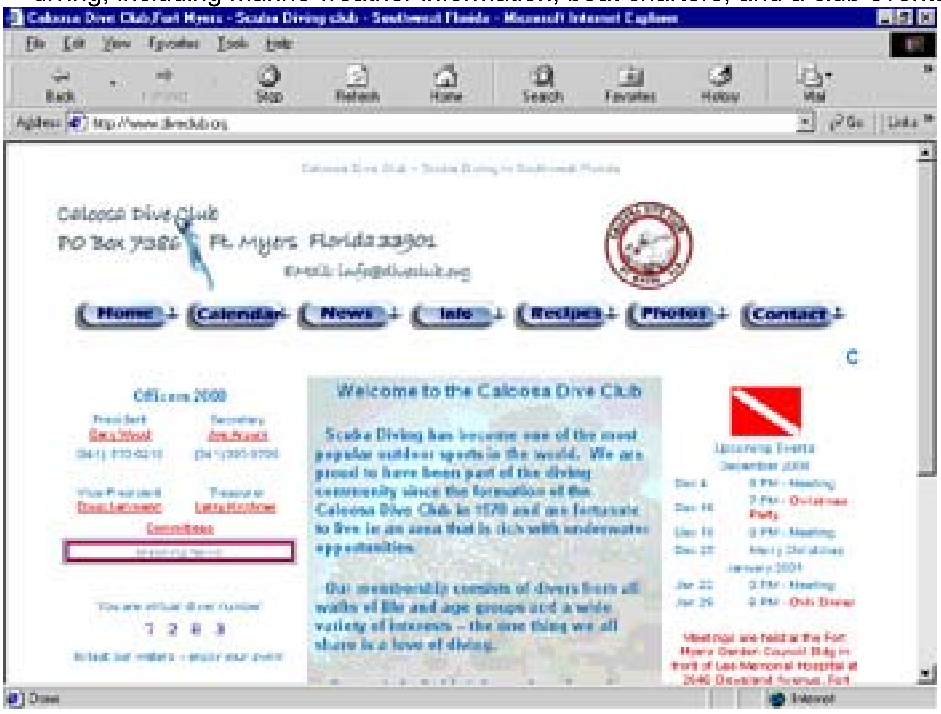
Tip Feedback from your Web site is often very useful. Although a Web-site statistics program can tell you the number of hits and how much time visitors spent viewing your site, nothing provides you with better perspective than the quick e-mail from a visitor. Make it easy for people to drop you a note to let you know what they think of the site and perhaps to make suggestions for improvements.

Develop a Look and Feel

Now you get to the exciting part: designing the look and feel of your Web site! Your online image should support your message and invite your audience to receive that message. Without the appropriate image, your message may never be read—your visitor may glance at the site and move on to the next. Finding your look and feel is a matter of creativity and (yes, you guessed it) planning. Next, you'll learn some tricks to do both!

Portraying an Appropriate Image

I'm not talking about political correctness here; rather, I'm discussing the appropriate match between message, image, and audience. You encounter messages every day that strive to get your attention via a look and feel. Consider the hard edge feel of many blue jeans commercials on TV, the escapist images presented by a bubble bath, the fast-paced, party themes of soft-drink commercials. Each of these advertisements set a strong mood and pulls you into the scenario portrayed, if only for a moment. Figure 13.1 shows The Caloosa Dive Club Web site, devoted to local residents and visitors who share "a love of diving." The site is informational and offers the visitor comprehensive information about local diving, including marine weather information, boat charters, and a club events calendar.



Welcome to the Caloosa Dive Club

Scuba Diving has become one of the most popular outdoor sports in the world. We are proud to have been part of the diving community since the formation of the Caloosa Dive Club in 1970 and are fortunate to live in an area that is rich with underwater opportunities.

Our membership consists of divers from all walks of life and age groups and a wide variety of interests – the one thing we all share is a love of diving.

Figure 13.1: The Caloosa Dive Club Web site (`http://www.diveclub.org/`)

The designer has chosen to use a watery blue theme to evoke feelings of the ocean, contrasted by the red and white of a diver's flag. The site's logo features the club patch, also done in the traditional dive colors in keeping with the theme, and a complete and easy-to-use navigation system that leads to the dozens of pages about the club, it's members, and resources for all.

Figure 13.2 presents Florida International University's Web site for its Latin America and Caribbean Center. The site provides for the visitor a wide variety of information, detailing everything from academic programs to publications and research to outreach and events. The designer has selected a background of glyphs used in ancient Mayan carvings, along with contrasting photographs of modern Latin American cities and ancient ruin sites, evoking a strong cultural flavor and an emphasis on heritage. A graphical text menu encourages the audience to visit the various sections of the site. The site feels well organized, conservative, and hints at upward mobility—an image universities typically portray.

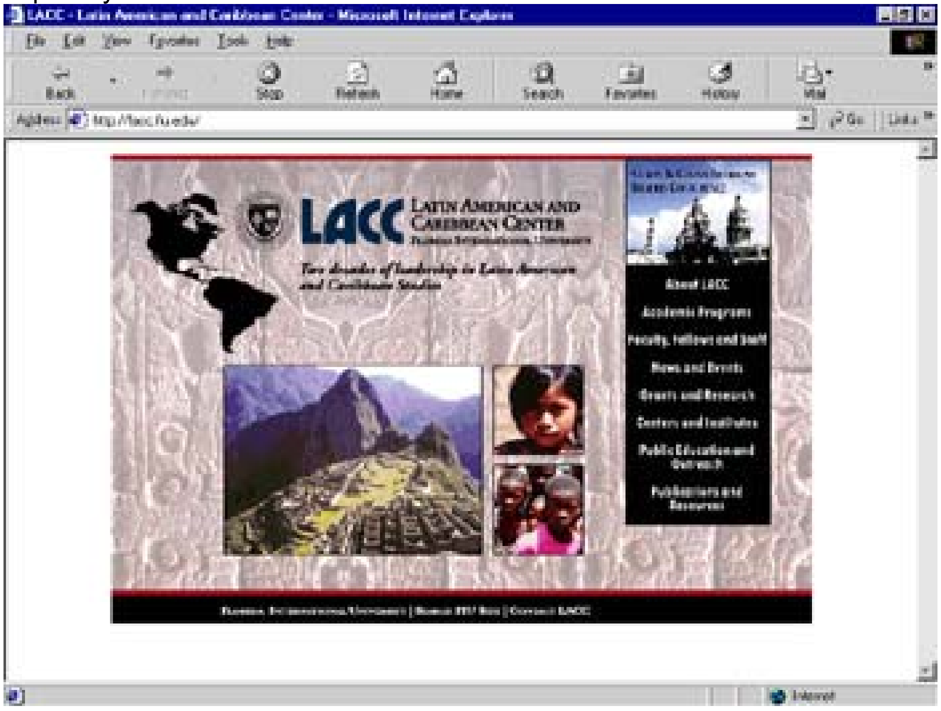
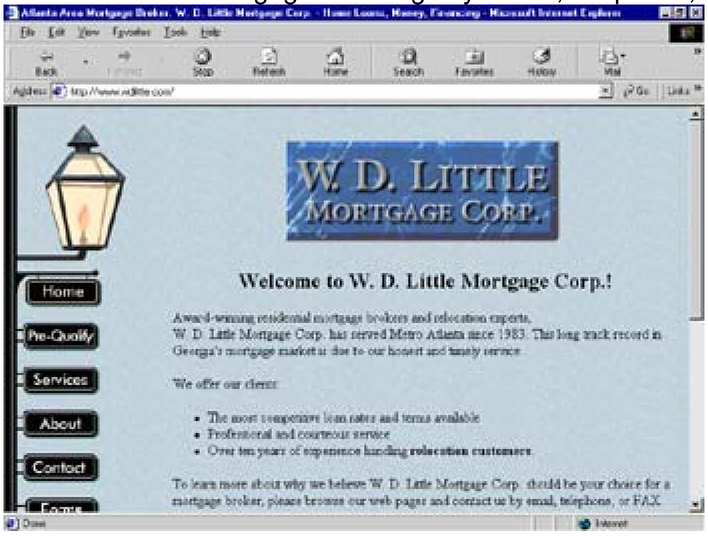

LACC - Latin American and Caribbean Center - Microsoft Internet Explorer
http://lacc.fiu.edu/
LACC Latin American and Caribbean Center
Two decades of leadership in Latin American and Caribbean Studies
About LACC
Academic Programs
Faculty, Fellows and Staff
News and Events
Grants and Research
Centers and Institutes
Public Education and Outreach
Publications and Resources

Figure 13.2: LACC at Florida International University (http://lacc.fiu.edu/)

W. D. Little Mortgage Corporation's Web site, presented in Figure 13.3, is a commercial site of a mortgage brokerage in Metro Atlanta. The site uses a gaslight and navigation buttons disguised as street signs to give a period feeling to the site that reminds the user of longevity, stability, and old-fashioned good service. The marble logo implies affluence, desirable in a financial institution. On the *splash page*—the term often given to a short concise entry point to a Web site— the audience is invited to contact the mortgage brokerage by e-mail, telephone, or fax to discuss their service needs.



Welcome to W. D. Little Mortgage Corp.!

Award-winning residential mortgage brokers and relocation experts.
W. D. Little Mortgage Corp. has served Metro Atlanta since 1983. This long track record in Georgia's mortgage market is due to our honest and timely service.

We offer our clients:

- The most competitive loan rates and terms available
- Professional and courteous service
- Over ten years of experience handling **relocation customers**

To learn more about why we believe W. D. Little Mortgage Corp. should be your choice for a mortgage broker, please browse our web pages and contact us by email, telephone, or FAX

Figure 13.3: W. D. Little Mortgage Corp (http://www.wdlittle.com/)

Presenting your information with a look and feel complementing your message is essential to engage your audience. People make decisions both consciously and unconsciously based upon their emotions. One of your jobs as a designer is to create an image for your Web site to evoke the appropriate emotion in your audience.

What *Is* Inappropriate?

Inappropriate can't really be defined by a hard and fast rule—it's subjective and based upon your message. The easiest way to illustrate an inappropriate image for the message is by example:

- A travel agency promoting island-vacation getaways is unlikely to publish full color photographs of run-down neighborhoods or communities. These photographs would be inappropriate to promote a message of tropical relaxation.
- A financial institution offering investment opportunities is probably not going to use the phrase "Go For It!" to encourage people to invest in a high-risk fund. Financial organizations usually use conservative images, fonts, and words to promote a feeling of security and deliberate decision-making. A whimsical approach is unlikely to encourage investment.
- Another inappropriate image would be for a governmental Web site to use a photograph of its chief executive standing in front of a limousine or yacht purchased at taxpayer expense. This would be a negative message to the site's audience.

Visioning and Free Association

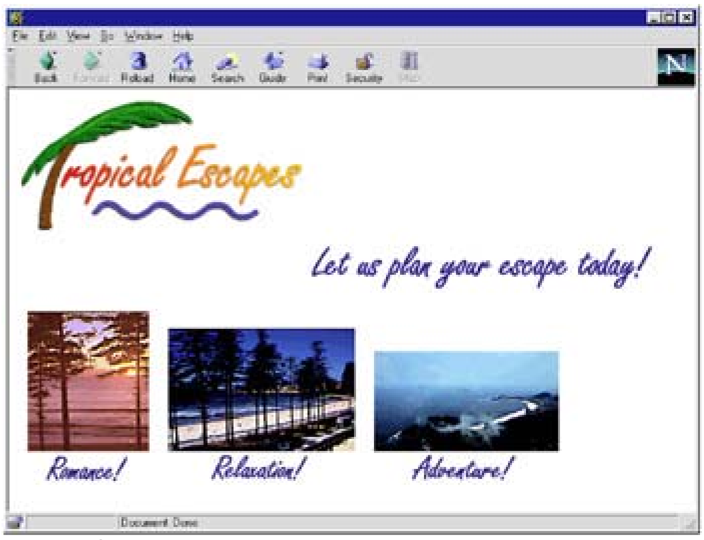
Can you get your message across to the audience quickly? Does your company name say it? Can you say it in a headline? How about one word? One image? Try doing some free association to find the best ways to deliver your message.
Visioning isn't difficult; you do it unconsciously all the time. When you drive home from work and imagine dinner on the table, when you make your holiday gift list, when you think about your vacation plans, you are visioning. You think about the event or the subject and develop images in your mind related to the topic. You can almost smell the dinner, see the smiles of loved ones opening presents, and feel the wind in your hair and the freedom from regular routine. Now apply this daydreaming to designing your Web site.

For example, pretend you are designing a Web site for a travel agency specializing in tropical vacations. What images and feelings come to mind? Blue oceans, tropical flowers, sailboats, relaxation, parrots in trees, palm trees, adventure, hot sun, white beaches, water rushing up to shore, shells on the beach, romance, tropical punch, exotic food, outdoor torches at parties…

You can appeal to people both emotionally and visually by using those images. Imagine the colors involved for a tropical vacation: the hot pinks and oranges of the flowers, the cool greens in the palms, and the serene blue of the ocean. A number of images could be used as graphical elements throughout the Web site: perhaps a parrot as a guide to various destinations? A conch shell as a bullet point? Photographs of sailboats against an azure sky generate a feeling of adventure. The romantic would identify with a photo of a candlelit table set with a bowl of exotic fruit and a bottle of wine. And what is more enticing than the image of people relaxing in the sun on an endless white beach?

Analysis of Tropical Escapes Web site

In the graphic below, some of the ideas I listed in this section are turned into a design concept for a hypothetical Tropical Escapes Web site. Although much remains to be done to improve the Web site for publication on the Web, you can already see that a theme is established.



"How?" you ask. It didn't take much to set the theme. The colors used in the logo, a gradient from orange to red, bring to mind the hot colors of tropical flowers and sunsets. The palm tree used as the letter *T* in the logo enhances the feeling of the tropics in the company name. The blue curvy line below the company name evokes water. The blue text "Let us plan your escape today!" triggers an emotional response of "Yeah!" (Everyone wants to escape.) The photographs of tropical scenes enhance the emotional appeals printed below them: "Romance!, Relaxation!, Adventure!" If a tropical vacation is a likely possibility for the Web-site visitor, it will be hard to resist exploring this site further.

Visioning is an excellent way to gather impressions, ideas, and feelings about your subject. As you envision, take notes and don't be analytical about your ideas. The analysis comes later after you've captured all free-flowing ideas on paper. First, jot down as many as possible; then, go back to some of those notes and expand on them individually with more free-association.

Once you have your list, analyze it for common themes. Are there strong emotional concepts? Do clear images come to mind to illustrate these? Can you translate these concepts into a headline or copy for your Web site? You're on your way to finding the right image for your site.

Not All Ideas Are Positive

Did you have some negatives among your list of ideas? Are those negative concepts likely to occur to other people as well? If you can answer "yes," you may have found a "feature" to add to your Web site to help distinguish yourself from the competition!

In the previous travel-agency example, you could have thought about the hassle of organizing travel visas and passports. No one likes to deal with red tape, especially when they are dreaming of their vacation!

One option for the hypothetical travel agency would be to provide a service to help solve red tape easily. It could offer a list of phone numbers and addresses on the Web site; the same list could be offered via e-mail if the visitor submits their e-mail address. Perhaps an agent could call with advice in exchange for a phone number sent via e-mail. Each is an excellent hook to make your audience feel welcome and want to come back again!

Review Your Competition

Another way to help identify an effective image for your Web site is to visit your competition's sites and do a little market research. Reviewing how they present themselves on the Web is an excellent way to learn from both their successes and failures. After all, your competition is vying for the same audience you are.

So, get yourself to a search engine (you learned about some of these in Chapter 10) and track down your competition. Grab your pad of paper and get ready to make some notes.

When you visit each site, note the following:

- What is the message?
- What look and feel is being used? Is it appropriate to the message?
- What action do you feel compelled to take, if any?
- Is their message the same as yours? Does it have a different twist? Is there a negative to their message?
- What tools or features are they using at the Web site? Graphics? Animation? Plug-ins? Sound?
- Are the features professionally executed? Do they support the site's message or do they confuse and distract?
- Are you still waiting for images to load after five minutes?
- Does the site offer anything unique? Is it useful? Were you interested enough to go deeper into the Web site to take a look?
- Is the overall presentation effective? Why or why not?

If your project isn't too obscure, you will likely have a variety of sites to see the different ways your message has been handled previously. You can compare the different sites to each other and to the ideas that you have for your own Web site. Analyze both sites that work and those that don't; you can learn from both.

Tip As you visit each competitor's site, be sure to take note of commonalities. For example, do you always find links to outside data, graphs illustrating some aspect of the information presented, or testimonials? If you notice common themes or presentation of information, you may have encountered an industry standard, formal or informal, which is especially valuable to know when planning your own site. If most of your competition presents information in a particular way, you need to understand why—even if you choose not to follow the convention.

Determine Design Constraints

Before launching into a detailed plan of your Web-site design, it's important to define any constraints that you may have. Keeping constraints in mind while you develop your site will save you grief, time, and money.

Corporate Presentation Rules

Many businesses, especially large corporations, have a graphics-standards manual or specification sheet that describes how their corporate logos, trademarks, and other proprietary marks may be displayed in print or other media. Typically, these specifications or rules are used to make it easy for people in different departments or even countries to provide consistent advertising and other corporate communications.
Obviously, not all businesses or organizations have this level of documentation or formality. However, smaller organizations often adhere to undocumented rules by convention. Before you develop a design, it's a good policy to check existing promotional materials for any existing rules or conventions. In Chapter 14, you will learn more about how to ensure conformity in your Web-site design.

Copyright Issues

Copyright protects intellectual property, created by a person or an entity such as a corporation, from theft. So someone cannot use something you created without your permission. Intellectual property can include written materials, artwork, music, software, choreography, and other forms of expression.
Web pages are copyrighted as soon as you save the text or the image to disk. Although United States law doesn't require it, it is generally a good idea to actually mark your materials with a copyright statement. Typically, you would include at the bottom of your page the standard copyright symbol (©) along with the year of the publication, the name of the copyright holder, and the phrase *All Rights Reserved*.

If you are considering using artwork, an article, software, or other materials that you did not originate, be sure to ask the copyright holder for permission to use the property. Publishing or using copyrighted materials without the author's permission can result in a lawsuit.

Additional information about copyright can be found at the following URLs:

- **The Copyright Web Site** This site, at http://www.benedict.com, is maintained by Benedict O'Mahoney and provides some excellent plain-English information about copyrights, their use, abuse, and some sample case histories. The site has especially good links to Web-specific copyright issues.
- **10 Big Myths about Copyright Explained** This site, at http://www.clari.net/brad/copymyths.html, is maintained by Brad Templeton of Clarinet and provides a good list of urban myths regarding copyright.
- **United States Copyright Office (Library of Congress)** The official U.S. Copyright Office Web site, http://lcweb.loc.gov/copyright, provides a place for you to register your works or find more information about the copyright process.

The following is a brief list of some questions you may want to ask:

- **Corporate Colors** Does the business use a specific color? Often businesses have adopted a particular color, usually described as Pantone ### (with ### being a number), for the printing of all of their materials. Unfortunately, the color used by the business for print may not have a corresponding safe palette color. (You will learn more about this in Chapter 16.)

- **Logo Specifications** Does the corporate logo require a particular layout, such as always appearing on the left of the page? Is it always sized proportionately to other graphical or text elements?

- **Trademarks** Are particular phrases always capitalized? Are product names trademarked? Are you using trademarks of other companies on the site?
- **Grammar** Does the organization use a style manual such as the *Chicago Manual of Style*? The style manual dictates grammatical structures, so be sure to check this one!

- **No Existing Image** Is the company brand new or does it desire a revised image for its Web efforts? Be sure to look over the previous section "Visioning and Free Association" for design ideas.

Budget Issues: Time and Money

Time and money constrain everything, including just how cool your Web site can be. Your Web-site plan needs to include some budget considerations even if you are a volunteer site designer. After all, you really want to do something else besides work on just this one Web site, right?

Although it's difficult to predict how long it will take for you to design your first Web site, you can make some estimates of the time required to gather your materials, have images scanned, and other chores. If you are working with a client or for a large organization that needs to be involved insite review and approval, be sure to build some time into your development schedule to accommodate this process.

Also, make some assumptions regarding the long-term maintenance and updates of your Web site. If you are contemplating building a site that requires daily updates that cannot be automated, be sure to realistically estimate how much time you can allot for updates and how you will handle sick days or vacations. Few things look worse on the Web than a site that was designed for daily updates but hasn't seen an addition in three months.

Consider your Web-site design from a financial perspective. What is it likely to cost you to keep your site up-to-date? Is your design sufficiently general that you won't need to redesign pages as you add information? Is your XHTML so complex that it is difficult to make changes easily? Consider those questions before you write a single page. It's always far more expensive (and tedious) to rethink a design when you are fairly far into your project, especially when you are very close to deadline.

If you are building a complex Web site and you're like most designers, you cannot personally create all features of your site. Peqrhaps you are a whiz at writing XHTML, but you couldn't write a script or draw an icon if your life depended on it. You have a few choices:

- Use public-domain graphics and scripts.
- Purchase the rights to use copyrighted images or software.

- Hire a subcontractor to provide you with original graphics or custom software.

Subcontracting is an excellent way to get high-quality materials designed specifically for your needs, and it usually will save you time. If you choose to use a subcontractor or purchase rights to use copyrighted materials, be sure to include the expense in your budget planning.

Resources for Scripts and Images

The Web has excellent resources for scripts and graphics that are either in the public domain or available for a fee. Check out the following URLs:

- The CGI Resource Index (http://cgi-resources.com/)
- Matt's Script Archive (http://www.worldwidemart.com/scripts/)
- Realm Graphics (http://www.ender-design.com/rg/index.html)
- Art Today (http://www.arttoday.com/)

Gather Your Existing Materials

Okay, roll up your sleeves and get ready to dig deeper into the serious business of planning. Here's where you get to shape your site from the lists you have made.

With your message defined, you will now determine the details of the information you will present to your audience. Remember, no matter how good your idea is, if it is poorly presented it won't communicate your message and may even be worse than having presented nothing at all. The key to successful communication is in the delivery.

Your mission is to locate all pertinent materials that will help deliver your message on the Web site. Does a brochure tell your story well? Perhaps you could use the copy in the brochure as a draft of a company-history page? With a little rethinking and some copy changes, the copy in corporate brochures can provide an excellent starting point for Web pages.

What about your organization's artwork? How much of it is already available electronically? Are existing photos or graphics appropriate for your site's message?

The following are examples of written content you can gather:

- Marketing materials (sales brochures, prospectuses)
- Fact sheets (parts lists, rate cards)
- Advertisements
- Press releases (also flyers, posters, invitations)
- Curriculum guides
- Training materials (manuals, guides)
- Periodicals (newsletters, magazines, bulletins)
- Forms (order forms, surveys, applications)

Check for these types of images:

- Photographs (original images when possible, see Tip below)
- Artwork (logos, illustrations, icons, nameplates)

Tip It is best to use original art, whether photos or artwork, whenever possible. Most images reproduced for printing, such as in a magazine or newsletter, have been *halftoned* (sometimes referred to as screened), a process that optimizes an image for viewing on paper. Halftoning creates beautiful printed output; however, if you scan the printed piece to use as a Web image, you will see an ugly pattern of dots, known as a *moiré pattern*. It is very difficult to remove without sophisticated image-editing software.

Here are examples of other materials you could gather:

- Videos (a commercial, the company picnic)
- News articles (print, voice, Web, or video)
- URLs
- Assorted reference materials

Get Selective

Now that you have your materials together, you can review your collection, keeping in mind your site's statement of purpose and your visioning exercise.

In some cases, existing materials aren't appropriate for the Web. For example, local advertising flyers don't usually translate smoothly to the Web. However, elements in those flyers likely could be used, with some modification, for the worldwide nature of the Web.

If you are designing a typical business Web site, you will probably have redundant information in your traditional materials. For example, your corporate brochure and product sales sheets may contain similar information about company history and background. In designing a Web site, you will want to provide that information only once. You can do so with hyperlinks very easily.

Tip "Less is more." These magic words will help you time and time again in the design process. Consider writing them on a sticky note and attaching it to your monitor. You need to walk a tightrope between providing your audience with sufficient information to make your point and boring them by repeating yourself. Web-site visitors have a limited amount of time they are willing to spend reading your message. Take advantage of every second that your visitor is willing to give you!

Marketing on the Web is different from traditional marketing. You have limited real estate on your splash page to present your message. You don't have to say everything on your splash page; use hyperlinks to provide detailed information on subsequent pages. Think of your splash page as the storefront to your Web site. Use that storefront to encourage people to come inside. Plan now to use that real estate effectively.

A Few Last Items

At this point, you have your materials selected, you have some ideas for creating your look and feel, and you are chomping at the bit to get started. Well, take just a bit more time to consider a few more items to complete your plan.

Do You Need to Do Research?

Is anything missing from your materials that would strengthen your presentation? Perhaps it would be helpful to provide some basic data about your topic from an independent source. Do you need to do some market research on the people that purchase your product or service? Would it be useful to offer your audience testimonials about your product?

A variety of resources are available for obtaining research: industry publications, market research firms, your own customer service department, and, of course, the Web. Don't forget to check out the search engines as an excellent source for supporting information.

Do You Need to Provide Off-Site Links?

Off-site links present both advantages and disadvantages. The disadvantage of having off-site links is that your visitor may leave your site before completely absorbing your message and acting on it. The advantage of off-site links is that you can provide your visitor with backup information related to your topic without having to duplicate information already available elsewhere on the Web.

It's tough to choose which way to go or whether to compromise. This decision needs to be made carefully and is very dependent on your Web site's purpose.

Two compromises on this issue are common:

- Provide off-site links only when the linked page offers some detail to assist in understanding your material. You can make a rule to never offer off-site links on pages that require the visitor to take some action, such as filling out an order form. After all, you don't want to lose your customer!
- Provide a links page with links to additional information. You are assuming that your visitor has browsed your entire site, is still looking for more information, and will leave anyway. A collection of links is then a friendly assistance.

Tying It All Together: A Case Study

The following is a short case study of the Oxalosis and Hyperoxaluria Foundation's (OHF) Web site. The foundation's mission statement reads: "OHF is committed to providing information, service, and support to patients and to families of those with oxalosis and hyperoxaluria and to the medical professionals who treat them." The foundation was formed in 1989 by parents of children affected with the rare disease hyperoxaluria. OHF has an international membership base and provides information and support to its members in a variety of ways: a quarterly newsletter, informational brochures, a support network of physicians, supportive people dealing with the same issues, and a toll-free phone number.

Most people learn about OHF when they are in crisis; that is, when their child (or, in the case of physicians, their patient) has been diagnosed with the life-threatening rare disease and they need help immediately.

Site-Planning Process

The OHF Web site must provide information to three general groups of people: lay-persons who are often frantic and in need of immediate support, medical professionals seeking information about research and treatment methods, and medical researchers seeking information and grant funds.

A great deal of thought went into the planning of the OHF Web site. A previous volunteer Web site had been successful at attracting new members to the organization and providing people with much needed information; however, it was out-of-date and needed additional data and services.

Site Purpose and Goal

OHF's mission statement served as a pretty clear site purpose: "to provide information and support to the people affected by the rare disease." OHF had some subsidiary goals that they also hoped to accomplish with the Web site: "to encourage financial support of the organization through a membership program and fundraisers and to solicit information from medical professionals via various surveys and grant applications."

Special Considerations

When the site-planning process was conducted, OHF's past experience with a small volunteer Web site was considered. Analysis of the organization's membership indicated that the membership wasn't very Web savvy and that it tended to use older equipment and slow modems. Past e-mail to the Webmaster indicated that users were relatively new to the Internet and needed information presented simply with few bells and whistles.

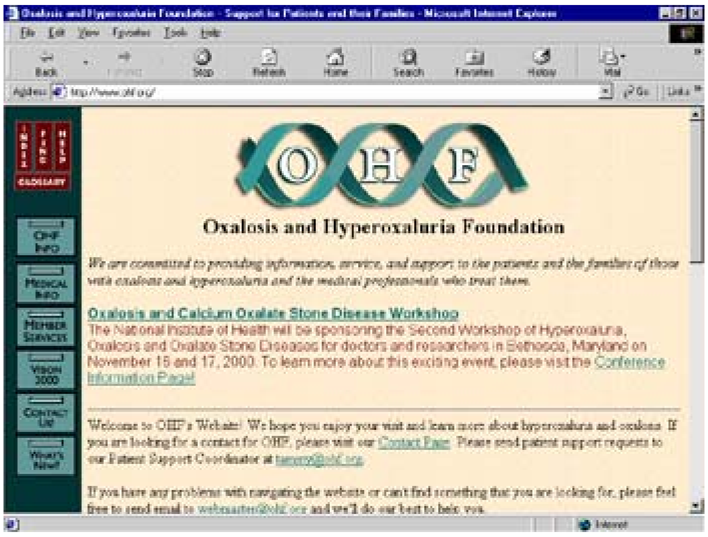
Development of a Look and Feel

Producing a look and feel for the OHF Web site was a challenge. The information they needed to provide to visitors wasn't especially visual. Few appropriate images come to mind to describe a rare disease. However, the site's purpose was to provide information, so the visioning process focused on images connected to presenting organized information to an audience unfamiliar with the Web.

A file cabinet metaphor was chosen for site navigation and graphical elements because most people are familiar with a typical office-filing system for organizing information. The concept of each drawer being dedicated to a specific subject and containing related files is intuitive and easy to remember. The metaphor of adding "files" to the collection of information is easy to extend and therefore makes site maintenance relatively simple.

Implementation

The file-cabinet metaphor was implemented by creating pictures of file-cabinet drawers labeled with the name of each second-level page of the Web site. These drawers appear on the left side of every page. Figure 13.4 shows the Oxalosis and Hyperoxaluria Foundation's (OHF) main page. The nonprofit's mission statement, displayed beneath the logo, clearly defines the site's purpose. On the left side of each Web page, a set of drawers presents the visitor with links to information that answers the questions listed in Table 13.2.


Oxalosis and Hyperoxaluria Foundation

We are committed to providing information, service, and support to the patients and the families of those with oxalosis and hyperoxaluria and the medical professionals who treat them.

Oxalosis and Calcium Oxalate Stone Disease Workshop
The National Institute of Health will be sponsoring the Second Workshop of Hyperoxaluria, Oxalosis and Oxalate Stone Diseases for doctors and researchers in Bethesda, Maryland on November 15 and 17, 2000. To learn more about this exciting event, please visit the Conference Information Page!

Welcome to OHF's Website! We hope you enjoy your visit and learn more about hyperoxaluria and oxalosis. If you are looking for a contact for OHF, please visit our Contact Page. Please send patient support requests to our Patient Support Coordinator at [illegible]@ohf.org.

If you have any problems with navigating the website or can't find something that you are looking for, please feel free to send email to webmaster@ohf.org and we'll do our best to help you.

Oxalosis and Hyperoxaluria Foundation - Support for Patients and their Families - Microsoft Internet Explorer
http://www.ohf.org/
OHF INFO
MEDICAL INFO
MEMBER SERVICES
VISION 2000
CONTACT US
WHAT'S NEW?
GLOSSARY

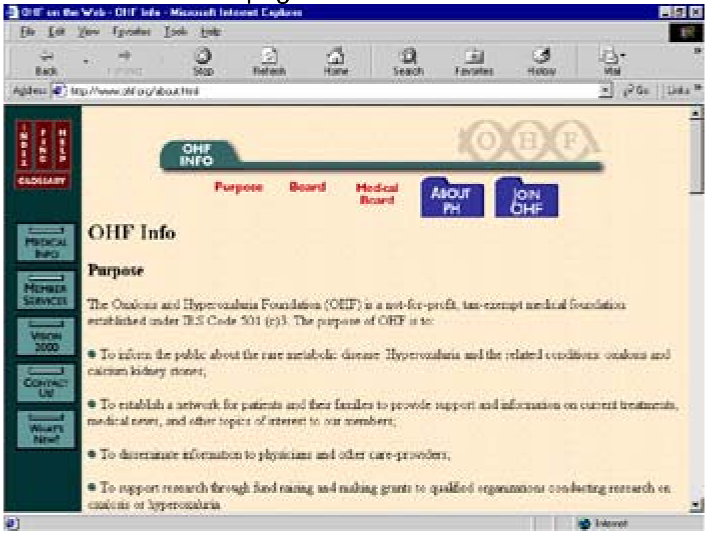
Figure 13.4: OHF Website (http://www.ohf.org/)

Table 13.2: Questions Answered on the OHF Site

FILE DRAWER	QUESTIONS IT ANSWERS
OHF Info	What is OHF? What does it do?
Medical Info	What kind of disease is oxalosis and hyperoxaluria? How does it affect people?
Member Services	What are the benefits to joining OHF?
Vision 2000	Why is the organization fundraising? How can I help? Does it have research grants?
Contact Us	Where is OHF located? To whom can I ask a question?
What's New	I've been to this site before—is there anything new here?

The copy on the main page welcomes the visitor, encourages them to explore the site, and invites their questions via e-mail. Also provided are text links to pages contained within the site that may address a visitor's special interests not targeted on the navigation graphics. The text links are designed to be easy for the Webmaster to change as the organization's needs change. One of the text links is to a physicians' survey. This online survey has been publicized in various medical journals and its placement on the splash page makes it easier for busy people to find.

Figure 13.5 illustrates the About OHF page, a second-level page in the Web site. The file-cabinet metaphor has been continued with a tabbed-folder image at the top of the page. The navigation bar at the top of the page also offers graphical links to sections within the existing page. Separate folders are links to third-level pages within the site.


OHF Info

Purpose

The Oxalosis and Hyperoxaluria Foundation (OHF) is a not-for-profit, tax-exempt medical foundation established under IRS Code 501 (c)3. The purpose of OHF is to:

- To inform the public about the rare metabolic disease Hyperoxaluria and the related conditions oxalosis and calcium kidney stones;
- To establish a network for patients and their families to provide support and information on current treatments, medical news, and other topics of interest to our members;
- To disseminate information to physicians and other care-providers;
- To support research through fund raising and making grants to qualified organizations conducting research on oxalosis or hyperoxaluria

OHF on the Web - OHF Info - Microsoft Internet Explorer
http://www.ohf.org/about.html
OHF INFO
Purpose
Board
Medical Board
ABOUT PH
JOIN OHF

Figure 13.5: About OHF

The file-cabinet drawers on the left side of every page give the visitor the ability to jump to another section of the site, thus reducing confusion for inexperienced visitors. Additional navigational tools always appear in the upper left corner of each page. These tools lead to a site-index page, a search page, a glossary, and a help page.

Conclusions

To meet the design goals of accommodating older equipment and slow modems, alternative text was generously used on all images so that nongraphical browsers and regular browsers running with images turned off still displayed links clearly. The site does not use animations, JavaScript, or other sophisticated plug-ins. A Perl script and a search engine are used on subsidiary pages.

To provide the quick support needed by patients and their families in crisis, the site offers the navigation graphics with links to medical information, member services, and the contact page. The needs of physicians and researchers are met with the same navigation graphics to medical information and the contact page.

The OHF Web site design has been quite successful to date. The original design goals of providing information for patients and professionals have been accomplished. Traffic to the site has been steadily increasing, questions regarding location of information are negligible, and site visitors use the interactive forms.

Summary

Effective Web-site design doesn't happen by chance; it happens by careful identification of your message and planning the best way to deliver that message. Designing your first site will probably be the hardest you will ever do—you're trying to incorporate and use many new skills at once.

Don't expect your first design to be perfect—Web sites are constantly evolving and changing. Often those necessary changes won't be obvious until you've had the site up for awhile and received some feedback from visitors. Even experienced Web designers continually make changes to their sites. Web design is a growth process—learn to enjoy the changes!

Chapter 14: Effective Visual Presentation

Overview

Now that you've identified your message, developed your Web site's look and feel, and selected the materials you are going to use, you need to decide where to place text and graphics for the best effect. This chapter will teach you how to achieve a unified layout and structure for your Web site. You will learn some basic art principles for presenting information visually, how to use a grid to simplify your layout decisions, and how to maintain a consistent presentation throughout your site. You'll also look at some examples of Web sites that have "done it right."

This chapter covers the following topics:

- Applying basic art principles
- Designing images, graphics, and text
- Using grids
- Putting it all together

Basic Art Principles, a.k.a. Art 101

Effective visual presentation doesn't just happen—it results from the designer's decisions about placing specific elements on a page. Whether you are designing a newsletter, an advertisement, or a Web page, you need to decide how to organize graphics and text to deliver your message in a logical and understandable manner. Although an experienced designer often makes these decisions unconsciously, they are based on an understanding of how the human eye perceives elements and organizes them into information that the brain can understand. In this chapter, you will learn some art principles that will form the basis for making those decisions.

Visual Unity

To understand visual information represented by a collection of graphical elements, you must be able to recognize a relationship among those elements—that is, some sort of pattern. In Figure 14.1, you can see a collection of elements that at first glance appear to have no relationship. Your eye looks at each of them, enjoys it, and then moves to the next element, much as you would view images in a scrapbook.

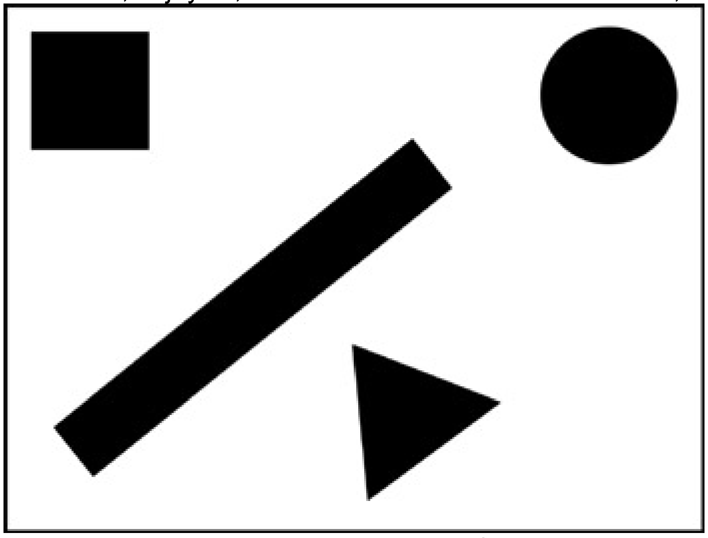


Figure 14.1: A scattered arrangement of shapes

Tip You'll find all the images in this chapter on the companion CD. Consider opening each of them as you read through the text to get the full impact of the design issues being discussed. Because the images are presented in gray scale here, some points I make may be lost visually if you just look at the pages.

After studying these elements briefly, you may notice that they are all geometric shapes—a square, rectangle, circle, and triangle. But the relationship of geometric objects is an intellectual relationship—not a visual relationship. Your eye sees elements on a page, and your brain interprets them and defines them as shapes. There is no visual unity in this example, only intellectual unity.

Unity by Proximity

In Figure 14.2, the same shapes are now visually related by proximity. Visual unity is achieved because your eye first sees the elements as a group rather than as a series of elements. The intellectual unity is still there; however, the whole image is now more obvious than the individual elements.

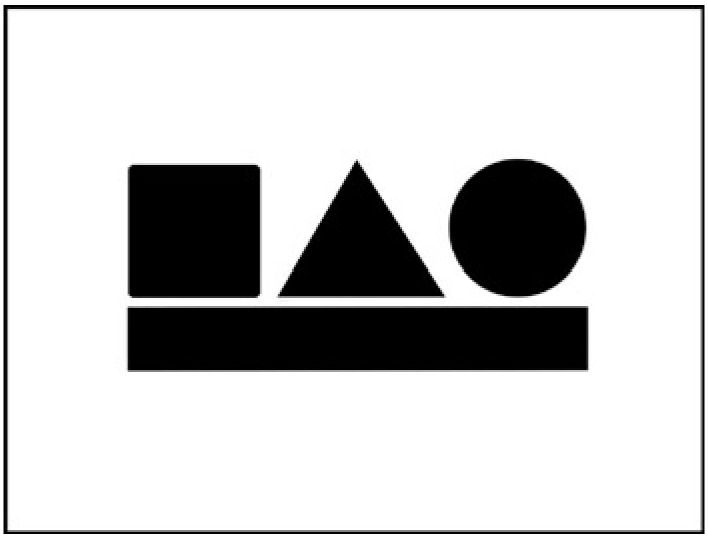


Figure 14.2: A more unified arrangement of the objects in Figure 14.1

Unity by Repetition

Another device for achieving visual unity is repetition. The repeating factor could be anything—a shape, color, line, angle, or nearly anything the eye will see and recognize as a repeated element. In Figure 14.3, you can see a variety of shapes enclosed by circles in a row. The repeated circle provides visual unity—the eye is drawn to the pattern and sees the series as a pattern. Also note that the shapes contained within each circle are approximately the same size in relation to the circle—another repeat factor that helps unify the images.

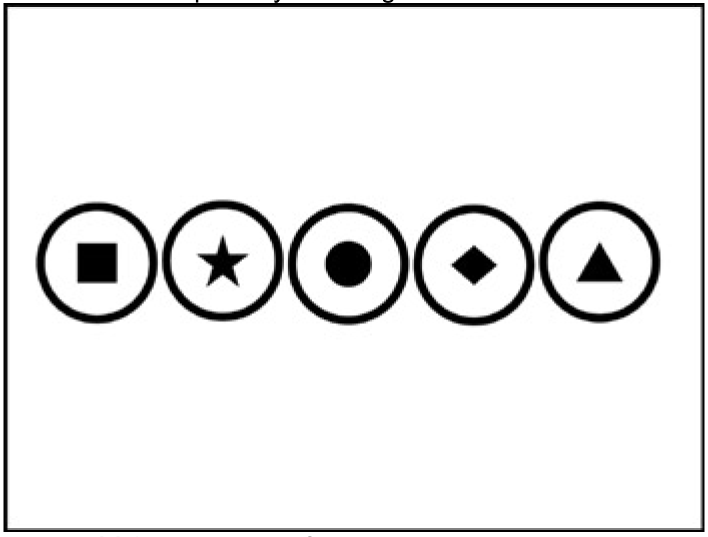


Figure 14.3: Repetition of the circles provides unity.

Figure 14.4 illustrates a common table-of-contents design that also demonstrates repetition, through text and graphic elements. Your eye recognizes the pattern of repetition in the triangles and circles. Your brain recognizes the information contained within this pattern. The elements clearly belong together; thus, visual unity and, again, intellectual unity are achieved.

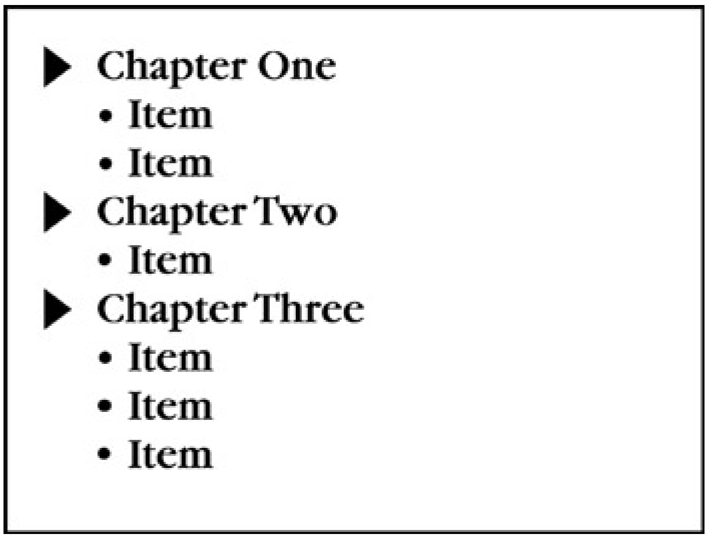


Figure 14.4: You can unify a table of contents through the use of bullets.

Creating Balance

People learn sense of balance as infants—if you hold young babies uncertainly, they feel off balance and startled. Children learn to feel balance in their bodies as they play on gym equipment and learn to ride a bicycle. People desire balance in their environments: They straighten a crooked picture frame on the wall unconsciously, they rearrange books stacked on a table until the stack looks stable, and they feel uncomfortable in a room with a sloping floor. Even language reflects the need for balance, as in phrases like "feeling off balance" that describe a feeling of disturbance.
The need for balance is again demonstrated in people's reactions to graphic design. Look at Figure 14.5. What is your immediate reaction? Do you feel a need to rearrange the circles on the page? Most people do. Your eye perceives the image as if it is divided in half with many circles appearing on the left of the imaginary line and only three circles on the right. This image feels out of balance and is visually uncomfortable for most people.

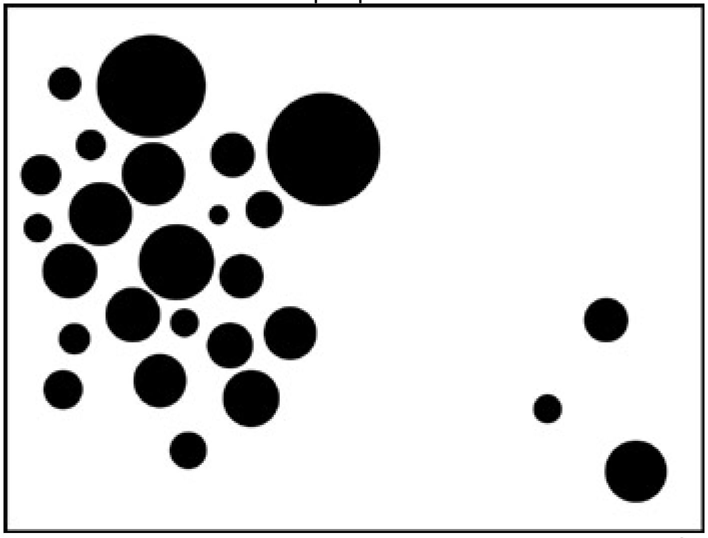


Figure 14.5: Are these circles balanced within the image?

Symmetrical Balance

You can use balance in a number of ways to arrange elements on a page. The simplest is symmetrical balance, which is best described as like elements repeated on either side of a center line. In effect, the image is mirrored.

Breaking Design Rules—Balance

As a designer, you can use the sense of discomfort generated by an out-of-balance composition to your advantage. You can draw attention to the element that is out of balance. The viewer will look at this item carefully, trying to make sense of it and searching for a way to integrate it into an overall feeling of balance.

Experiment with such a design idea and test it with some sincere critics before integrating it into your final design. It is a powerful tool and can be very successful if you employ it appropriately. If you use it improperly, it will certainly unsettle your viewer and perhaps frustrate him or her sufficiently to stop reading your message.

Figure 14.6 illustrates symmetrical balance. Two circles appear on either side of a central triangle. Although this is a simple example, you can see examples of symmetrical balance all around you, particularly in architecture and interior design. Symmetrical balance feels formal and rather static, but it's also comfortable and predictable.

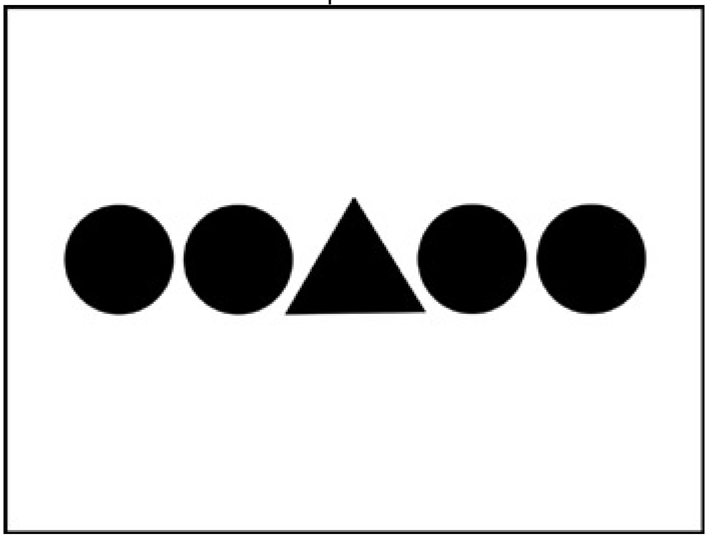


Figure 14.6: Symmetrical balance gives a feeling of "fit," like puzzle pieces properly in place.

Asymmetrical Balance

Asymmetrical balance has a more casual feeling and is typically more interesting to the eye than symmetrical balance. In asymmetrical balance, different elements are balanced by having a similar visual weight or eye attraction.

In Figure 14.7, asymmetrical balance is provided by similar weight on either side of the imaginary vertical line bisecting the large box and the eight smaller boxes. Although the elements are different, the image feels balanced to the viewer.

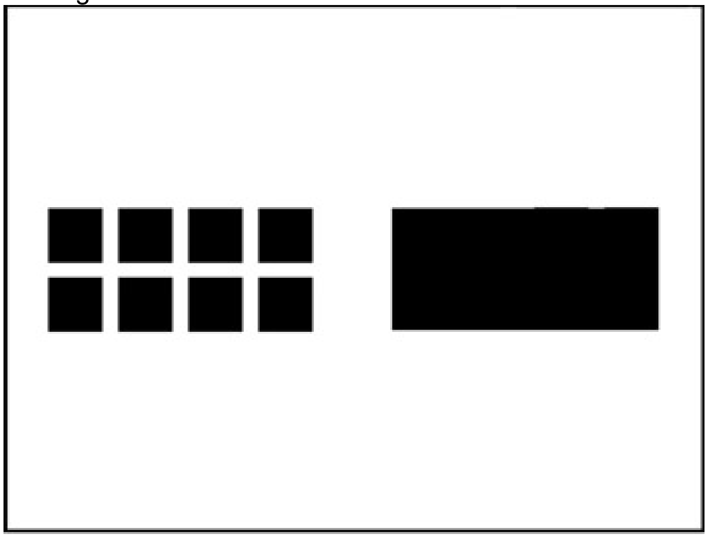


Figure 14.7: The group of smaller boxes on the left has the same total area as the single larger box on the right, giving the appearance of balance.

Figure 14.8 illustrates differing textures that accomplish asymmetrical balance. Your eye sees the text on the left side of the image as a visual gray texture, even though it also contains symbols with their own meaning. (Remember the difference between visual and intellectual information?) The gray texture of the text balances the solid smooth graphic because although the text is lighter in weight than the solid graphic, a texture is naturally more interesting to the eye.

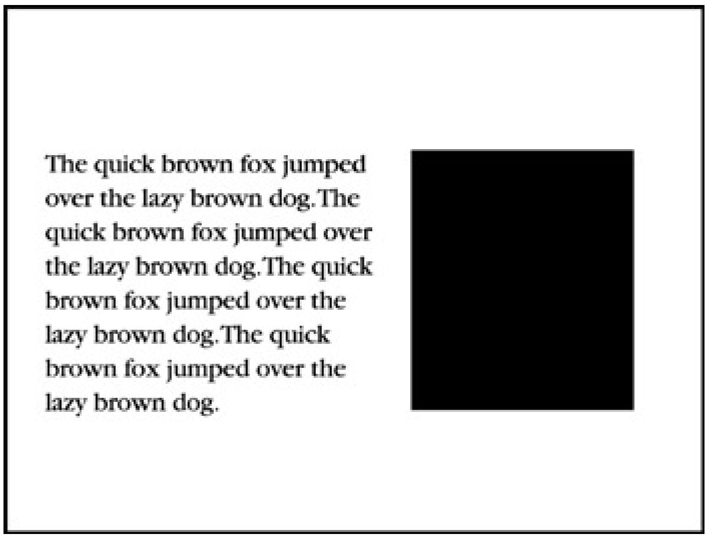

The quick brown fox jumped over the lazy brown dog. The quick brown fox jumped over the lazy brown dog. The quick brown fox jumped over the lazy brown dog. The quick brown fox jumped over the lazy brown dog.

Figure 14.8: Balance can be achieved with different textures.

Providing a Focus

You use a point of emphasis, or visual *focus*, to grab the viewer's attention and direct it to a specific element in a design; you want the viewer to consider the element more carefully than the surrounding elements. Although not all designs use a focal point, it is typically used when delivering a message. You can provide a focal point in any number of ways; some are subtle, and others are as strident as a flashing neon sign.

Using Contrast

One of the more common ways to direct the viewer's eye to a particular focal point is to use contrast. The contrast can be one of color, value, shape, texture, size, angle, or pattern. Figure 14.9 illustrates the use of sizing one element larger to create a focal point among a collection of small rectangles.

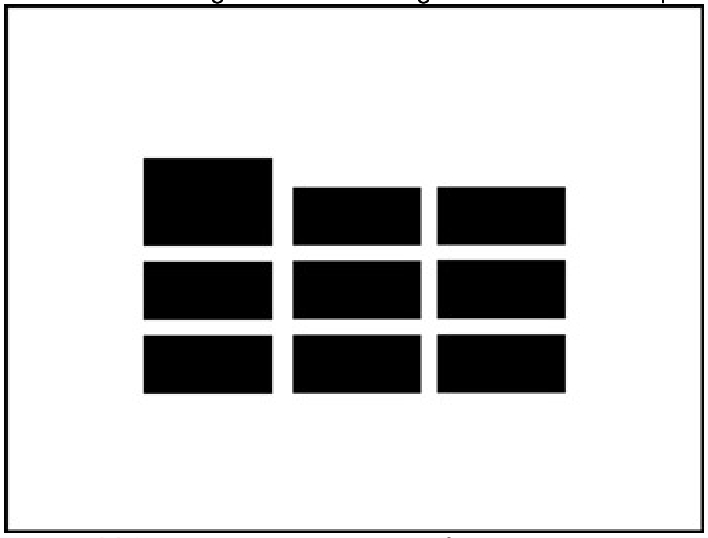


Figure 14.9: A designer can purposefully draw the viewer's eye to an object of dissimilar size.

Figure 14.10 uses a variant of the graphic in Figure 14.9 to illustrate using color as the contrast to create a focal point. The rectangle on the right is gray, and the other rectangles are black. In the image provided on the CD, the rectangle on the right is bright red; the drama of the bright red against the series of black rectangles is rather startling and draws the eye much quicker than the size difference shown previously.

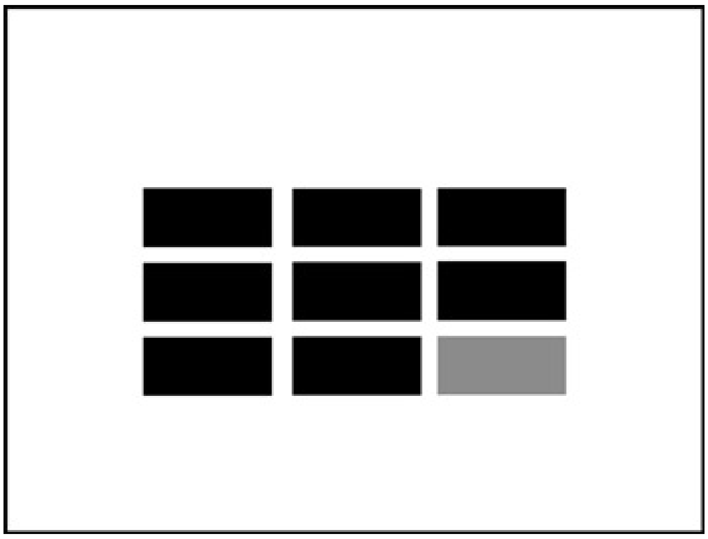


Figure 14.10: You can use color as well as size to produce contrast.

A Sense of Isolation

Another effective way to create a focal point is to isolate an element. Figure 14.11 shows an assortment of circles; however, one circle stands alone. Your eye is drawn to that lone circle, and you can't help but wonder why this circle is different from the rest.

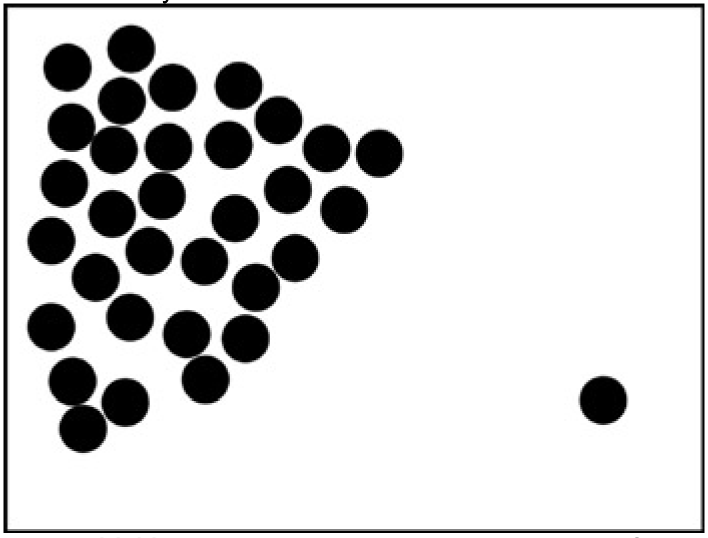


Figure 14.11: What is important about the lone circle?

Radial Placement

Figure 14.12 illustrates the common bull's eye—a simple focal point achieved by radial placement. The concentric circles surrounding the solid circle draw your eye to the center of the image, almost irresistibly. Imagine a photograph of a tunnel with something emerging from the tunnel or a crossroads with a person in the crosswalk. This is radial placement at work.

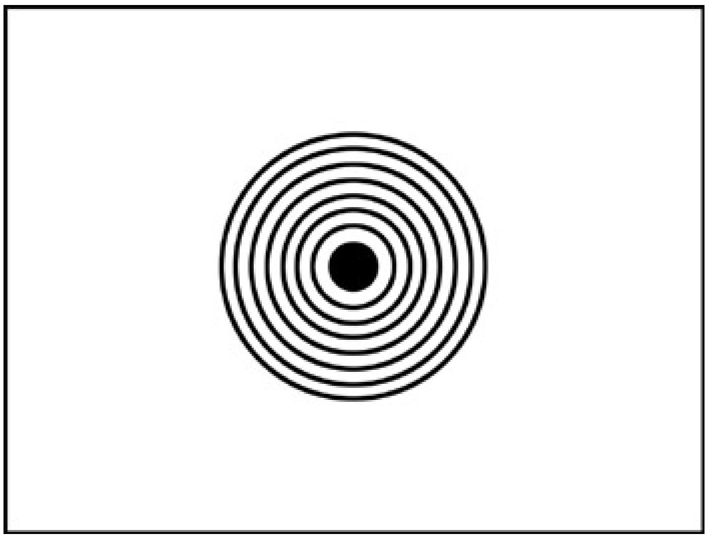


Figure 14.12: The bull's eye draws your eye toward its center.

Achieving Depth

Although Web-page design is a two-dimensional medium, you can achieve the illusion of depth in a number of ways.

Working with Size

Size is an easy way to imply depth; objects appear smaller as they recede into the distance. Figure 14.13 shows a series of rectangles that start tall at the right and left margins of the page and grow smaller as they approach the center. Your eye is fooled into believing that the images are receding into the distance, creating an illusion of depth.

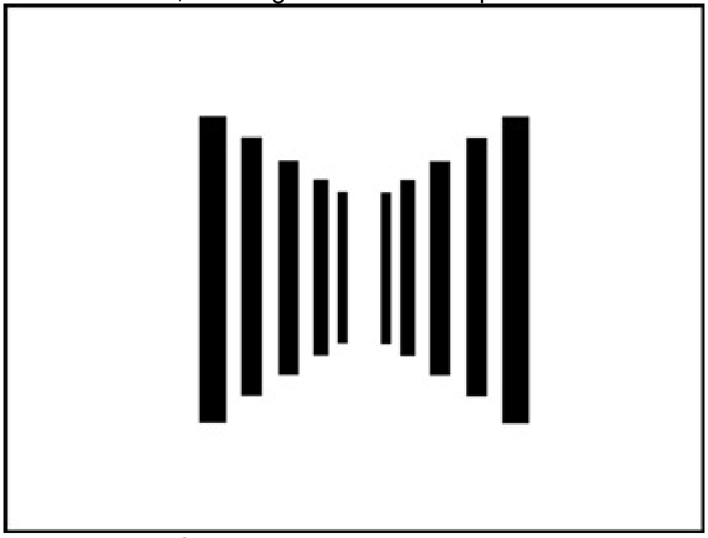


Figure 14.13: Shorter and thinner lines introduce the visual trick of depth perception on a two-dimensional page.

Add Layering

Another way to imply depth is through layering or overlapping elements. As you view a layered image, your eye perceives the top element to be "hiding" the partially covered element. As a result, you believe that you are viewing a three-dimensional image. Figure 14.14 illustrates this principle with a series of rectangles that appear to be stacked.

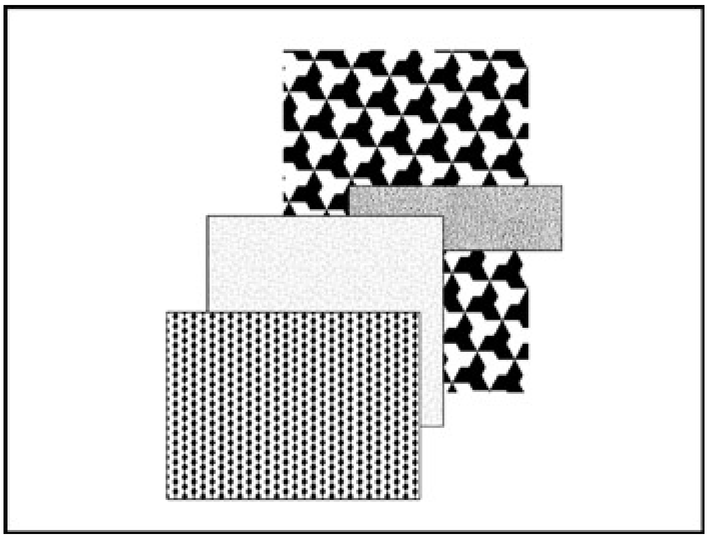


Figure 14.14: Is one image really on top of the other?

Creating Shadows

Adding a shadow to an element is another way to create an illusion of depth. Figure 14.15 illustrates several common shadow techniques used in desktop publishing and on the Web.

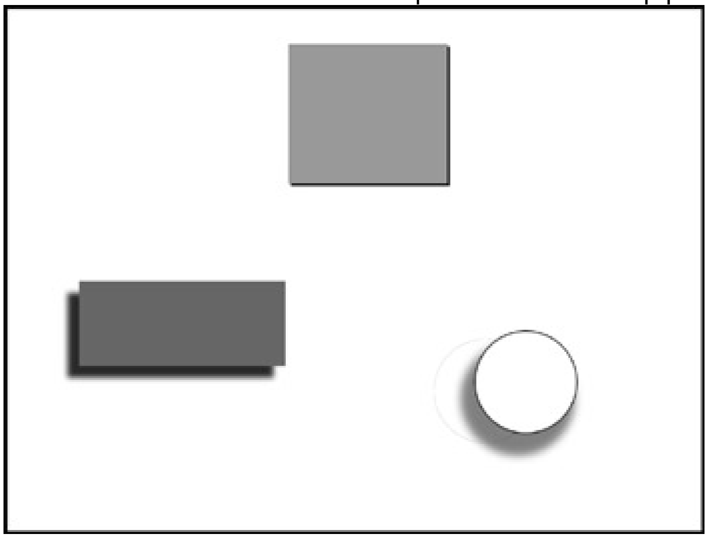


Figure 14.15: The different types of "drop shadows"

Images and Text

You can have the best copy, photos, and graphics and an exciting image for your Web site, but if they are not presented effectively, your audience won't give them a second glance. Effective presentation requires applying the basic art principles to your images and text. In the following sections, I have provided some specific examples that will help you present your Web site effectively.

Using Images

The liberal use of images is much of what makes the Web an appealing medium for providing information to the world. Full-color images have never been so easy! But presenting graphics and photographs in an appealing and accessible way requires some planning and an eye for detail.

Relevance to Copy

Illustrating copy isn't difficult. Many illustration resources are available to you, from do-it-yourself to professionally designed images. The key to effective illustration lies in determining the purpose of an illustration and achieving that purpose in your selection of images.

You might choose to provide illustration for your Web page for a number of reasons:

- Support or additional information for the copy
- Decoration
- Breaking up a dense text-oriented page
- Promoting a mood or feeling

Whatever your reason, you will need to identify it and be sure that the image you provide achieves your goal.

For example, if your copy is a step-by-step description of how to put together a technical device, placing a photograph of Monet's water lilies on the page won't contribute to your message and will almost certainly confuse and distract your visitor. This is an inappropriate use of illustration.

Illustrating a stock prospectus with a chart showing the performance of the stock over several years is an excellent use of an illustration. People grasp a well-designed information graphic much more quickly than a textual description.

Adjusting Sizing

Sizing your illustrations appropriately helps their overall effectiveness on your Web page. Pay careful attention to the balance of your illustration against your copy. An image too large or too small will unsettle your audience. Figure 14.16 shows an illustration that is too large for the copy.

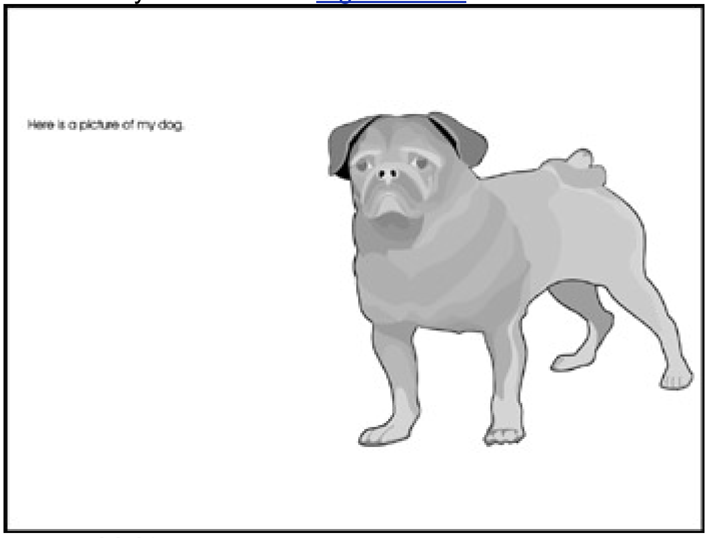


Figure 14.16: The text gets lost next to the large dog.

Incorporating Photographs

Photographs are an excellent way to illustrate your Web page; however, photographs often need some attention prior to placement in your pages. Look at the photograph to be sure that it is sharp, that the light and dark levels are correct, and that extraneous parts of the image have been cropped. Figure 14.17 shows a good photograph of a newlywed couple cutting their wedding cake. Although the photograph has plenty of interesting features, it also has a few problems. The original photo has some very dark shadows and the cake and table in the foreground quite bright white in color. Using photograph-retouching software, I was able to sharpen the image a bit and lighten some of the darker tones, bringing out some of the details in the shadows and lessening the brightness of the white foreground, as you can see in Figure 14.18.

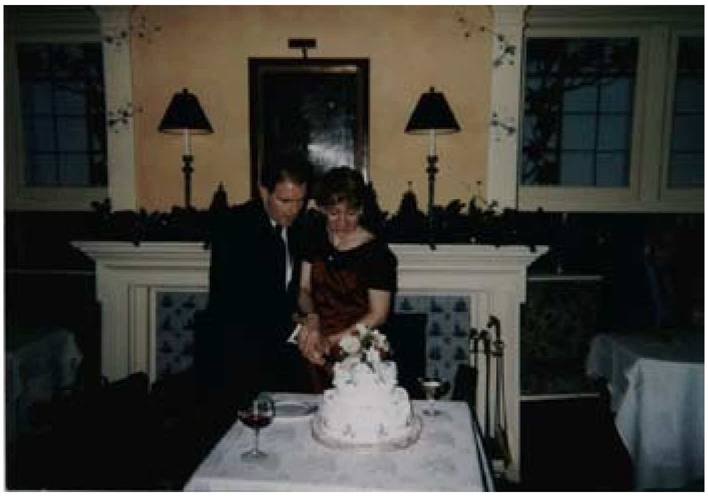


Figure 14.17: The original photograph

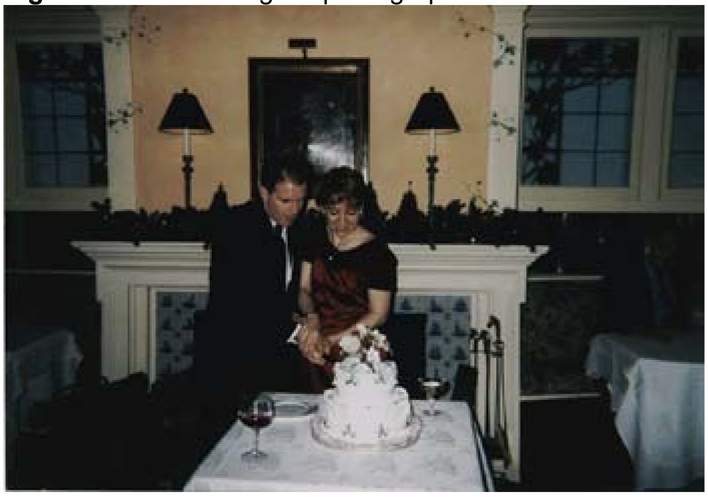


Figure 14.18: A sharper image, with more detail visible

The image isn't ideal yet. The far left has a darkened table and some other relatively unrecognizable dark lumps next to the mantel. On the right, the firetool set, the additional window, and table also distract from the focus of the image—the couple cutting the cake. By cropping out these areas, the photograph is much improved, as you can see in Figure 14.19.

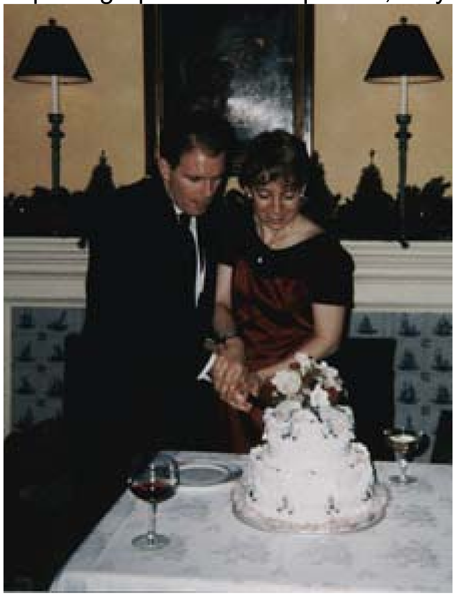


Figure 14.19: The cropped version removes the distractions.

Tip I'll take a look at some popular photograph-retouching software in Chapter 17.

Styling Issues

Once you decide on a look and feel, maintain consistency throughout the images on your Web site. You can use a variety of images, both graphical and photographic—just be sure that everything blends stylistically.

For example, it would be confusing to use the currently popular `50s retro black-and-white photographs next to Renaissance paintings and icons. Radical changes in style are distracting and often leave your audience puzzled and unsettled. Worse yet, if the changes are between pages on your Website, visitors will be even more confused. They will wonder if they have somehow left your site and are lost.

If your site has a great deal of variety in the images you are using, experiment with ways to create a feeling of repetition. Sometimes you can help tie together your images by using a common theme for presenting images, such as a consistent way of bordering photographs, a consistent style of shadow or spacing around graphic images, or a repeating color theme. In the sample sites at the end of this chapter, you will see how you might maintain the continuity of images throughout a Web site.

Effective Styling of Text

Of course, your copy is well written, is exciting to read, and has no spelling or grammatical errors. So it is ready to be pasted into your Web page—right? Well, maybe. You can use a few design principles for working with text to help your page go from good to great!

Tones of Type

When you look at a document, your eye takes in information about that page long before your read a single word. First you notice the tone, or value, of the page. Is the page covered with much of text, unrelieved by headlines, images, or white space? Your eye is evaluating the "tone" of the page. The quick look at a document sets the reader's expectations for that page: Will it be interesting? Dreary and dull? Short or long?

Black Space

Black space refers to any graphical information on the page, regardless of color. It includes photographs, illustrations, or anything else not either text or empty space. Your eye is naturally attracted to black space to avoid doing the work of reading. Have you ever decided to put down a document to read later because it looked like "work"? This is your eye's opinion of pages with little black space.

White Space

White space is your eye's second favorite thing to look at on a page. White space is empty space on a page—margins, breaks between paragraphs, and space around graphics. White space breaks up long columns of text, giving your eyes a rest from the work it will need to do in reading the copy on the page.

Gray Space

Gray space is the actual text on the page—your carefully written copy. The eye is least interested in looking at this information. It's hard work; all that text has to be processed. Consider your reaction to reading the fine print in a contract or reading classified ads in the newspaper. You don't want to do it, right?

Balancing Tone

Presenting text effectively requires balancing of the tone of text—providing sufficient black space and white space to offset the gray space of the text. By judicious placement of white and black space, you can provide your visitors with copy in the smaller doses their eyes crave, thus ensuring that your message gets read.

Using Grids

The basic grid is an invaluable tool to help simplify your layout decisions. A grid provides structure, defining where you will place elements on your page—both text and graphics. It also assists you with balancing tone and elements on your page.

Perhaps most importantly, a grid provides a strong feeling of continuity for your Web site. Your layout can provide your visitors with important clues about where to find information. If content always appears in a particular column and if your links and navigation graphics always appear in another, your visitors will naturally understand your design. That understanding makes them feel comfortable and reduces the possibility of their getting lost on your Web site.

If you analyze printed materials, you will usually discover an underlying grid. Business letters, invoices, tax forms, resumés, and even this book all use some variant of a grid for organizing information and helping lead your eye through the document. Even when the grid's lines are not apparent on the page, most documents use a series of columns and rows to format visual information.

One-Column Grids
A one-column grid is the simplest HTML page layout you can use. Text and graphics are laid out horizontally with or without wrapping text around graphics, as shown in the examples in Figure 14.20. Text and images can be centered, aligned left or right, or fit to the page without any formatting.

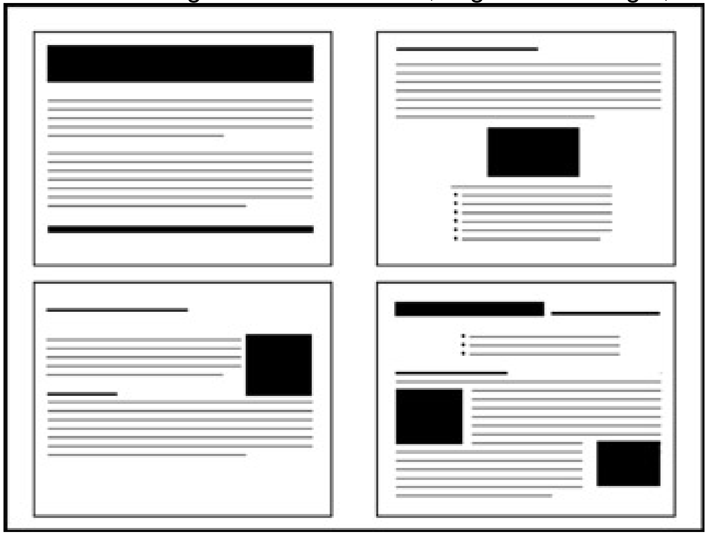


Figure 14.20: Four possible one-column grid layouts

A one-column grid works well for a variety of Web pages, particularly those that don't require elaborate formatting or presentation of complex graphics. It is especially good for the beginning designer. Simple XHTML is all that you need.

Long documents with sequential pages organized by chapter or section are particularly well served by a one-column layout. The addition of a table-of-contents page linked to each section or chapter page and navigation tools, such as a home, forward, and back button (to move through the document a page at a time), makes a one-column design ideal for manuals or other long documents read sequentially.

Tip The one-column grid tends to create very "gray" pages. Long uninterrupted paragraphs can create a dense and complicated look, making the viewer feel that the text may be tedious to read. You can break up the gray look by adding graphics, lines, and section headers where appropriate.

This variant of the one-column grid uses one wide column for presenting most of the content and a narrow column for links or other navigation content. It is commonly used both on the Web and in traditional paper publishing. For the Web, the narrow column most typically appears on the left side of the page.
In the examples in Figure 14.21, you can see that a single column is used for the content of the document, whereas the narrow column can be used for locating graphics, subheads, links, navigation buttons, or even small amounts of text. Formatting can be carefully aligned left, right, or center or can flow naturally, depending on the choice of the designer.

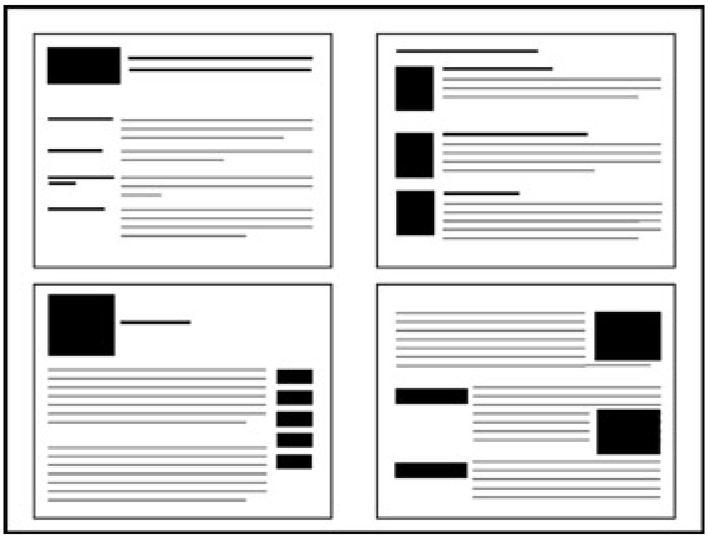


Figure 14.21: Several examples of a modified one-column grid

Although the variation requires tables, it is a sophisticated look for a Web site that isn't too complex for the beginning Web designer to produce. (Refer to Chapter 5 for more information on tables.) The design naturally incorporates white space, which sets off the elements in the narrow column and gives your pages a lively feeling.

Another variant of the one-column grid uses two narrow columns for presenting links and images, while continuing to restrict content to the wide column. Figure 14.22 shows several versions of this layout.

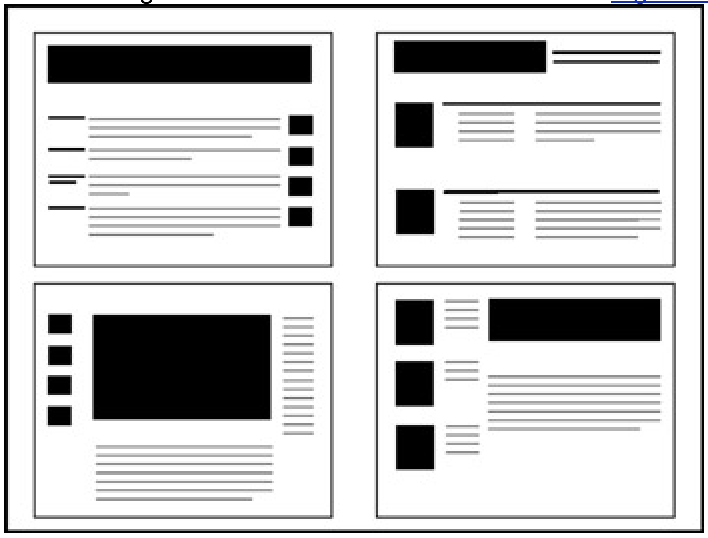


Figure 14.22: An additional variant of the one-column grid method

With the addition of a second narrow column, the flexibility of your design increases; however, so does the complexity of writing the HTML. Although the design is not complicated, planning becomes more important. Balancing this layout is a bit more of a challenge than with the previous designs.

This grid design works especially well for complex Web sites containing several sections or areas that your visitors might want to view in an unpredictable (rather than sequential) manner. You might use one narrow column for section navigation and then use the other narrow column for overall site tools such as search-page links, glossaries, and other links supportive of the overall site. When used consistently throughout your Web site, this structure is easy for your audience to understand.

Putting It All Together: A Gallery

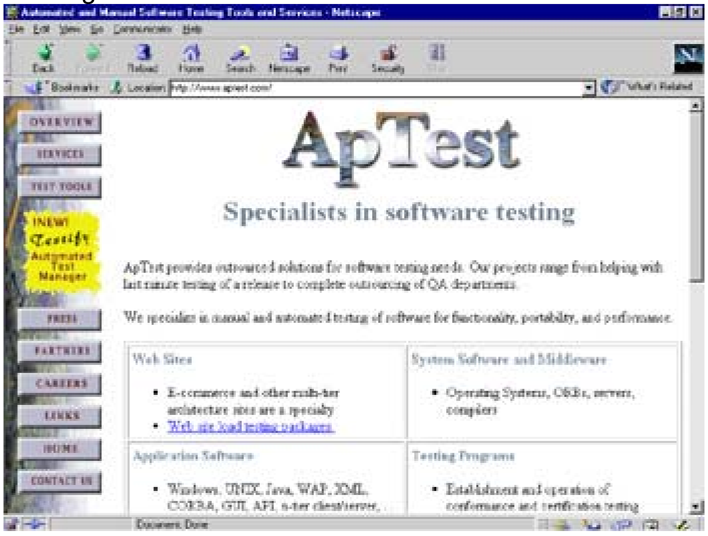
I've presented a great deal of information, and you're trying to figure out just how you integrate all of it—right? Well, don't despair; it's really not as difficult as it seems. Once you make some of the basic decisions about your look and feel and the grid you will use, many decisions will fall naturally into place. To help you see the integrated product, here's a quick tour of a couple of Web sites. I'll point out where some of the principles I described are used.

Applied Testing and Technology, Inc.

Applied Testing and Technology, Inc. (ApTest) provides companies with extensive outsourced solutions for software-testing needs. The Web site (http://www.aptest.com) is professional and has a sophisticated artistic touch, demonstrated in the logo and the background image that runs behind the navigation buttons on the left-hand side of the screen.

Art Principles

As you can see in Figure 14.23, the home page of the ApTest Web site, the navigation graphics on the left column of every page use simple shapes, symmetrically balanced. The shapes are the same silver-gray as found in the corporate logo; repeating the corporate color helps unify the site's look. The soft drop shadow behind the logo gives the illusion of depth and helps dress up the simple letter-based design.


Automated and Manual Software Testing Tools and Services - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.aptest.com/ What's Related
OVERVIEW
SERVICES
TEST TOOLS
NEW! Testify Automated Test Manager
PRESS
PARTNERS
CAREERS
LINKS
HOME
CONTACT US
ApTest
Specialists in software testing
Document: Done

ApTest provides outsourced solutions for software testing needs. Our projects range from helping with last minute testing of a release to complete outsourcing of QA departments.

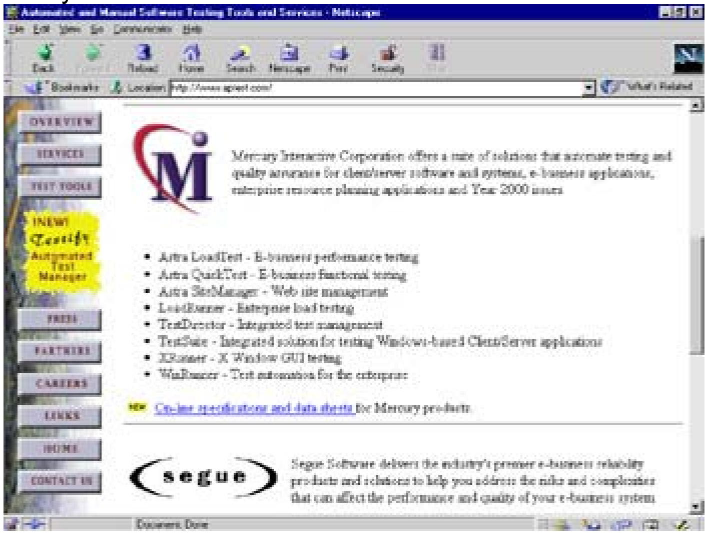
We specialize in manual and automated testing of software for functionality, portability, and performance.

Web Sites	System Software and Middleware
• E-commerce and other multi-tier architecture sites are a specialty • Web site load testing packages	• Operating Systems, ORBs, servers, compilers
Application Software	**Testing Programs**
• Windows, UNIX, Java, WAP, XML, CORBA, GUI, API, n-tier client/server,	• Establishment and operation of conformance and certification testing

Figure 14.23: ApTest

Using a Grid

The ApTest site primarily utilizes a modified one-column grid throughout the pages. Because one-column grids tend to look somewhat "gray," the site designer broke up the copy with plenty of white-space dividing lines between segments, as shown in Figure 14.24. Logos are provided for a little asymmetrical balance.


Automated and Manual Software Testing Tools and Services - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.aptest.com/ What's Related
OVERVIEW
SERVICES
TEST TOOLS
NEW! Testify Automated Test Manager
PRESS
PARTNERS
CAREERS
LINKS
HOME
CONTACT US
segue
NEW
Document: Done

Mercury Interactive Corporation offers a suite of solutions that automate testing and quality assurance for client/server software and systems, e-business applications, enterprise resource planning applications and Year 2000 issues

- Astra LoadTest - E-business performance testing
- Astra QuickTest - E-business functional testing
- Astra SiteManager - Web site management
- LoadRunner - Enterprise load testing
- TestDirector - Integrated test management
- TestSuite - Integrated solution for testing Windows-based Client/Server applications
- XRunner - X Window GUI testing
- WinRunner - Test automation for the enterprise

On-line specifications and data sheets for Mercury products

Segue Software delivers the industry's premier e-business reliability products and solutions to help you address the risks and complexities that can affect the performance and quality of your e-business system

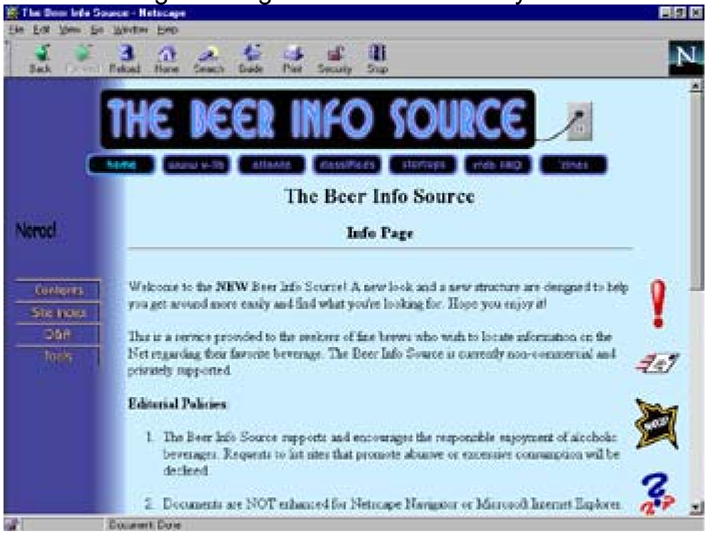
Figure 14.24: Images and white space with rules help break up the text.

Beer Info Source

WWW For the beer lover, I introduce you to the Beer Info Source (http://www.beerinfo.com/). It is a large Web site with more than 100 pages of information. The complexity of the site and the need for easy navigation mandated careful attention to structure and design.

Art Principles

As Figure 14.25 shows, the site's look and feel are unified by the neon signs, which are used consistently throughout the site. The large logo's rounded-corner rectangle shape is repeated in the site-navigation buttons shown just below the logo. The Home button is animated, providing visitors with a reminder of their location within the site. The neon and type styling are repeated throughout the site, maintaining a strong sense of consistency and site identity.


The Beer Info Source - Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
THE BEER INFO SOURCE
home
Nerod
Contents
Site Index
Q&A
Tools
Document: Done

The Beer Info Source

Info Page

Welcome to the **NEW** Beer Info Source! A new look and a new structure are designed to help you get around more easily and find what you're looking for. Hope you enjoy it!

This is a service provided to the seekers of fine brews who wish to locate information on the Net regarding their favorite beverage. The Beer Info Source is currently non-commercial and privately supported.

Editorial Policies

1. The Beer Info Source supports and encourages the responsible enjoyment of alcoholic beverages. Requests to list sites that promote abusive or excessive consumption will be declined.
2. Documents are NOT enhanced for Netscape Navigator or Microsoft Internet Explorer.

Figure 14.25: The Beer Info Source splash page

At the far right of the logo, you can see an electrical cord and plug going into an electrical outlet. The graphic particularly entertained visitors and raised a number of questions, such as, "What happens if I try to unplug the sign?" The site owner was so amused that he chose to add a hidden link—an "Easter egg"—to amuse the curious.

Figure 14.26 shows what happens if a visitor "unplugs" the electrical cord. The fun aspect of design at this site led to visitors actually having an impact on further Web-site design.

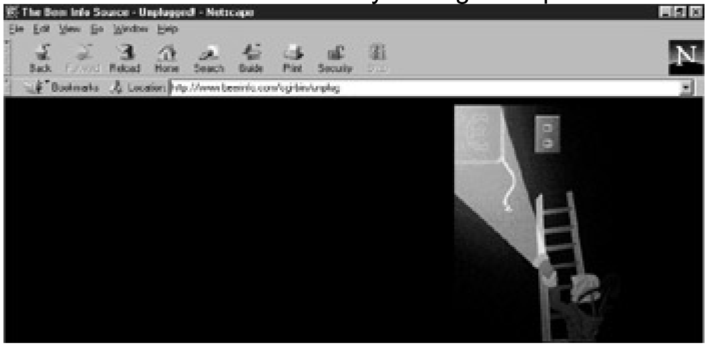

The Beer Info Source - Unplugged! - Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://www.beerinfo.com/cgi-bin/unplug

Figure 14.26: The Easter egg

Using a Grid

Beer Info Source uses a modified one-column grid consistently throughout the Web site. As you can see in the diagram in Figure 14.27, the structure is formal and consistent, despite the variety of sections. The visitor quickly learns the navigation scheme and can travel the site with ease. This design never requires the visitor to go backward in order to jump to a new section. All sections are available from all pages.

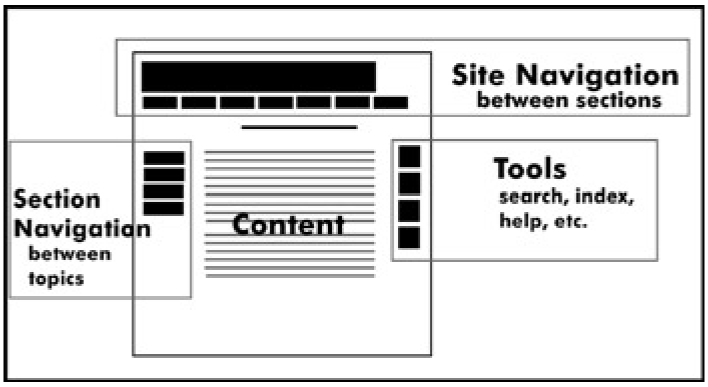


Figure 14.27: A consistent structure helps your visitors with navigation.

One page at Beer Info Source does use a somewhat different format, as you can see in Figure 14.28. The same basic structure of the site is used with the navigation buttons and tools; however, a calendar created by a separate table with the borders turned on is displayed in the area normally used by textual content.

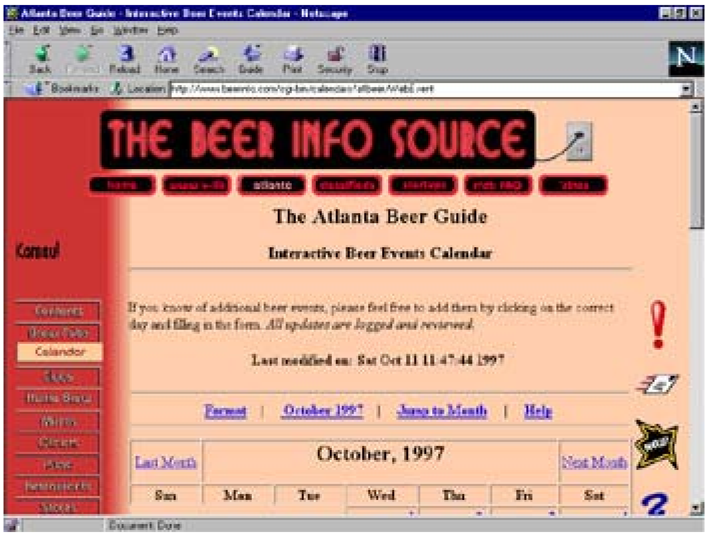

THE BEER INFO SOURCE
The Atlanta Beer Guide
Interactive Beer Events Calendar
Calendar
Last modified on: Sat Oct 11 11:47:44 1997
Format | October 1997 | Jump to Month | Help
Last Month
October, 1997
Next Month
Sun
Mon
Tue
Wed
Thu
Fri
Sat

Figure 14.28: The table structure used at `http://beerinfo.com/cgi-bin/calendars/atlbeer/WebEvent` helps offset the calendar information.

Summary

Effective visual presentation is not as complicated as it looks. By combining basic art principles, a good eye, design guidelines, and some common sense, you can design a Web site that delivers its message *and* looks good. Spend some time reviewing the Web sites presented here and visiting sites online. Take the principles you've learned and see if you can identify them in other Web sites. Learning design comes from studying the principles and then observing them in action. As you become more aware of the way Web sites are designed, you will find yourself feeling more comfortable in your own design decisions.

Chapter 15: Making a Statement with Fonts

Overview

As you know from working with traditional paper documents, fonts can make a really big difference in the way your documents look. You can invoke a wide variety of feelings just from the look of the type. You can go from a formal, almost rigid look to a whimsical or childlike presentation. You can use fonts for the same effect on the Web, as long as you keep certain limitations in mind.

This chapter covers the following topics:

- Typeface characteristics
- Changing font defaults in IE and Netscape
- Assigning fonts in XHTML
- Font properties in style sheets

A Typeface Primer

Three major terms are used to describe the styling and presentation of text on both paper and the Web: *typeface*, *font family*, and *font*.

A typeface is a specific design for a set of characters. The design is what gives the characters their unique look. Is there a curlycue at the bottom of the letter *g* or a long, waving top stroke on the capital letter *F*? Does the whole set look like the handwriting work of a five-year-old? All sorts of features like these go into designing the typeface, which is given a name by its designer. Common typefaces are Arial, Helvetica, Times New Roman, Garamond, and Courier.

A *font*, then, is a specific instance of a typeface. The additional properties of a font include the size measured in points, *pitch* (width of the standard character within the font, measured in characters per inch), and spacing. Therefore, if I were to correctly refer to a font, I'd need to say something like, "Courier 12 point, with pitch value of 10."

Tip When Web designers talk about typeface design, the words *typeface* and *font* are often used interchangeably, especially among those who don't come from a formal art or design background. The practice shouldn't necessarily be discouraged, but when working with a designer, vendor, or other contractor, be sure you're both using the terms in the same fashion.

The point value used to measure font size is a height measurement, with each point is approximately 1/72 of an inch. The Courier 12 point font, then, would be 12/72 of an inch high, or easier read as 1/6 of an inch in height. Many browsers generally use a 10-to 12-point font size as the default font size. Later in this chapter, I'll look at how the user and the designer can make changes to those sizes.

Finally, the font family is a set of fonts of the same typeface that have different sizes and other adjustable characteristics.

Typeface Characteristics

Typefaces and their associated fonts can be divided into two broad categories: fixed-width (also called monospaced) and proportional fonts. A fixed-width font gives the same amount of space to each letter, whereas a proportional font gives different amounts of space to different letters. Look at the difference:
`This sentence is written in Courier New, which is a fixed-width typeface.`
The typeface for this sentence, Helvetica, allocates more space for the letter *M* than it does for the letter *I*.

Many people consider typefaces to be the basis of graphic design. Well-chosen and well-placed typefaces are a key factor in whether documents communicate effectively to the intended audience. *Typography* is the study of the selection, combination, and placement of fonts.

Typefaces and fonts have been around for a long time—one could argue that ancient cuneiform was an early typeface. But they became more noticeable, and certainly more standardized, with the invention of the printing press. With the desktop-publishing revolution in the late'80s and early'90s, users often went crazy trying a multitude of fonts and typefaces in the same document. Many newsletters and other publications took on a sort of "ransom note" effect, as if each portion of it were cut out of a different magazine or newspaper that each used a different typeface and font. With the introduction of the public Web, the number of personal-computer owners has exploded. Those with an artistic bent, as well as traditional typeface design houses, have been busy working on new fonts that read well on a computer screen, rather than fonts that look great on the printed page but perhaps not in your DTP program.

In almost all aspects of computing experience, the user is able to choose which typeface to use in various interfaces, including the ones to be used for text displayed in your Web browser. Going further, not only can the user choose a typeface, but which specific font to use as well! However, the ability to make such changes and manage them to provide a pleasing result can be harder than it sounds.

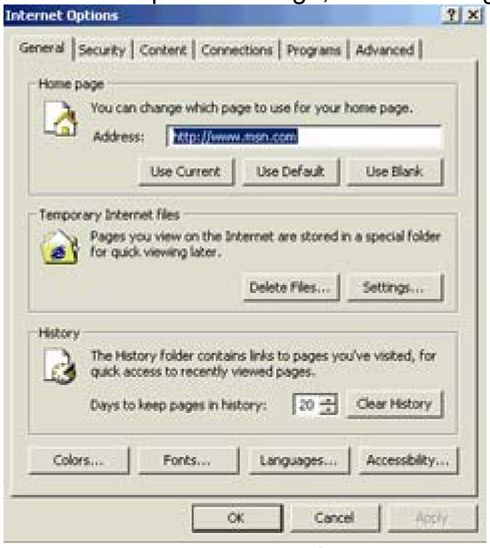
Default Fonts

Every application that displays text has a default font selected for display. Even the Windows and Macintos operating systems have default fonts for each portion of their user interface.

Most people never change their default fonts, quite often because they don't know that they can! Those who do know quickly find out what a handy tool it can be. Text menus can be made more readable by enlarging the font size; resumés can be crunched down to a single page by choosing a smaller-than-normal font such as an Arial 9pt. rather than Times New Roman 12. The possibilities are endless!

In the next section, I'll look at how to change your personal font settings in the most popular Web browsers.

Internet Explorer
Internet Explorer 5's menus underwent some housecleaning before the version was launched. As a result, the location for setting user preferences has changed. You can now find it by going to Tools → Internet Options. A large, tabbed dialog box that looks like Figure 15.1 will appear.



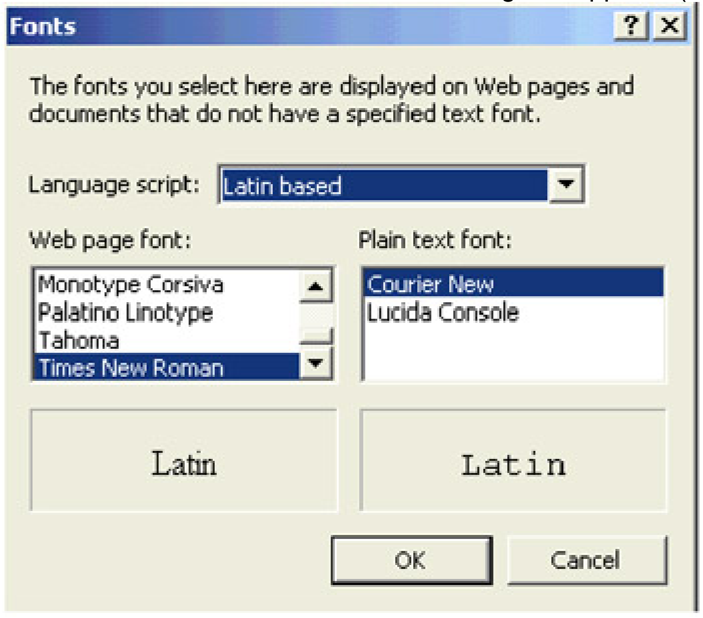
You can change which page to use for your home page.
Address: http://www.msn.com

Pages you view on the Internet are stored in a special folder for quick viewing later.

The History folder contains links to pages you've visited, for quick access to recently viewed pages.
Days to keep pages in history: 20

Figure 15.1: IE's Internet Options dialog box

The second button from the left among the row of four buttons immediately above the OK button is labeled *Fonts*. Select it, and a new dialog box appears (see Figure 15.2).



The fonts you select here are displayed on Web pages and documents that do not have a specified text font.

Language script: Latin based

Web page font: Monotype Corsiva, Palatino Linotype, Tahoma, Times New Roman (selected)

Plain text font: Courier New (selected), Lucida Console

Figure 15.2: IE 5.5 Font dialog box

The three major components are Language Script, Web Page Font, and Plain Text Font. The Language Script may be a description that's new to you, but don't let it confuse you. English falls into a broad category of Latin-based languages, as does Spanish, French, and many other Western languages. If you're curious, look at what other options are available in the menu; you'll see languages that use special character sets, like Cyrillic, Hebrew, Japanese, Arabic, and even Braille.

Warning In order for non-Latin-based language scripts to properly function on your system, the Internationalization support for those languages generally needs to be installed into Windows and most other operating systems. Without it, you'll see blank or gibberish characters.

The selections available in Web Page Font and Plain Text Font are limited to the fonts that you have installed on your system. All operating systems come with a basic set of fonts. Applications may install others for you (especially applications related to desktop publishing). You can also download/purchase and install fonts yourself.

The font selected from the Web Page Font list will be the default font for text display on your system. The Plain Text Font chosen will be used when a monospaced font is required—for instance, for rendering the contents of the pre element.

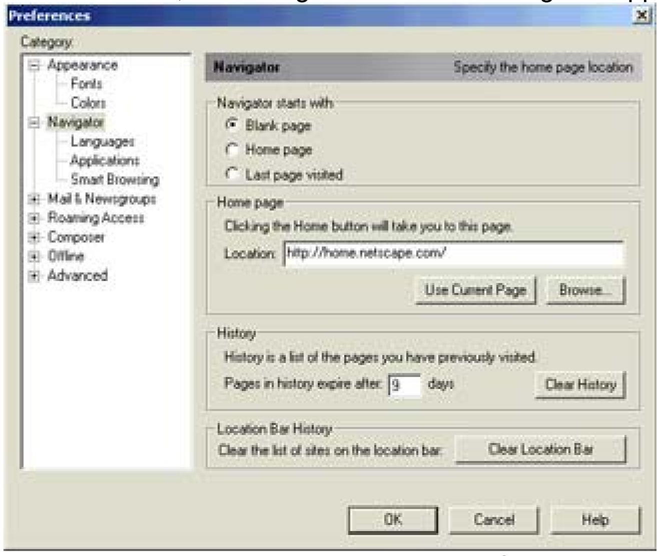
Note that these selections only choose the typeface, not the specific font size. To make that adjustment, you need to switch menus to View → Text Size and then choose a relative size from the list provided (from Largest to Smallest, with Medium being the middle of five choices).

Tip If you don't want your font choices overridden by a style sheet attached to a Web site, you'll need to make another change. From the same Internet Options dialog box, choose the Accessibility button and check two boxes: Ignore Font Styles Specified on Web Pages and then Ignore Font Sizes Specified on Web Pages.

It should be noted that the method for selecting and preserving font choices has changed significantly from versions before IE 5. In IE 3 and I.E. 4, the user would go to View → Options → General and click the Font Settings button in the lower right-hand corner.

Navigator

As always, the method for changing preferences varies from browser to browser. To make changes in Netscape Navigator, you'll need to visit a different set of menu options. This time, select Edit → Preferences, and a large Preferences dialog box appears (see Figure 15.3).



Navigator starts with
(●) Blank page
( ) Home page
( ) Last page visited

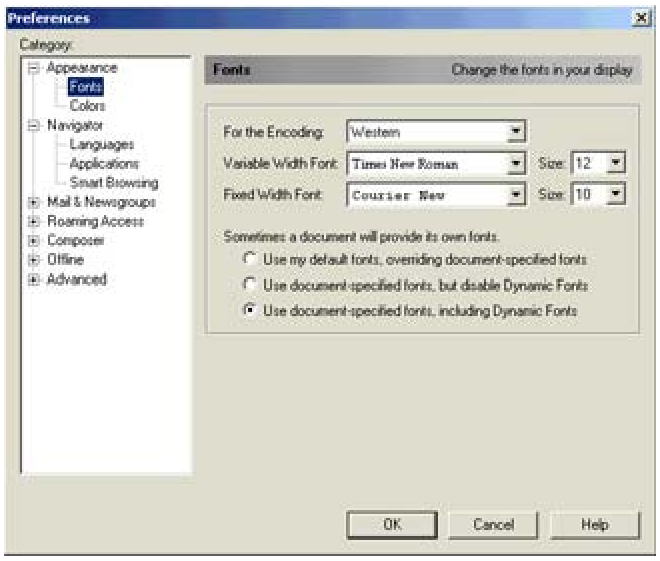
Home page
Clicking the Home button will take you to this page.
Location: http://home.netscape.com/

History
History is a list of the pages you have previously visited.
Pages in history expire after: 9 days

Location Bar History
Clear the list of sites on the location bar:

Figure 15.3: The Nestcape Navigator Preferences dialog box

To locate the font preferences, click on the Appearance item in the category list in the left-hand window of the dialog box. Two new choices appear: fonts and colors. Choose fonts. Netscape handily keeps all font-related options in the same area. Notice that the right-hand pane of the Preferences dialog has now changed to font-related properties (see Figure 15.4)



Fonts — Change the fonts in your display

For the Encoding: Western

Variable Width Font: Times New Roman — Size: 12

Fixed Width Font: Courier New — Size: 10

Sometimes a document will provide its own fonts.

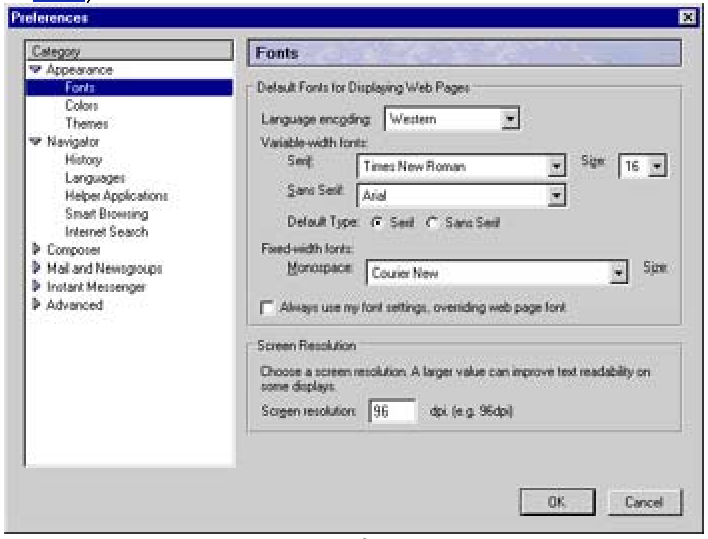
- ( ) Use my default fonts, overriding document-specified fonts
- ( ) Use document-specified fonts, but disable Dynamic Fonts
- [✓ Use document-specified fonts, including Dynamic Fonts]

Figure 15.4: Font properties for Navigator 4.76

The terminology used is slightly different than in Internet Explorer. Netscape provides choices for Encoding, Variable Width Font, and Fixed Width Font. These match the IE offerings of Language Script, Web Page Font, and Fixed Width Font. The term *Western* that you'll likely see preselected for the Encoding covers the Latin-based languages described previously.

Notice that in this dialog box you can directly set your font-size choice—not by a general "large" or "medium," but by specific point size. Finally, Navigator allows you to manage style-sheet font selections by always using your own font selections, using the document-specified font *unless* it's a Dynamic Font, or using the document-specified fonts *including* Dynamic Fonts.

The Navigator 6 font-selection options are rearranged just slightly from its 4.76 counterpart (see Figure 15.5).



Fonts

Default Fonts for Displaying Web Pages

Language encoding: Western

Variable-width fonts:

Serif: Times New Roman — Size: 16

Sans Serif: Arial

Default Type: [✓ Serif] ( ) Sans Serif

Fixed-width fonts:

Monospace: Courier New — Size:

[ ] Always use my font settings, overriding web page font

Screen Resolution

Choose a screen resolution. A larger value can improve text readability on some displays.

Screen resolution: 96 dpi (e.g. 96dpi)

Figure 15.5: The Navigator 6 font selection dialog box

Navigator 6 has only a single option to override (or not override) document-specified fonts, but it does introduce a new option to increase the screen resolution, which may also aid in reading smaller type.

Using XHTML to Specify Fonts

Web designers can specify the font that visitors are to see, provided that:

- The correct XHTML font specification is used.
- The user's browser supports document-specified font selection.
- The user has not opted to override document-specified font selection.
- The user's machine has the specified font installed.

That last item is of particular importance. You can specify a rare but particularly pleasing font (at least in your own eyes), but your visitors won't see the documents the way you can unless they have that font available on their machines.

When setting font selections for your documents, you have three options: using a `basefont` element to set the font selection for the entire page, using the `font` element within other elements to impact smaller portions of a document, or using a style sheet.

Tip The W3C has deprecated the making of font selections anywhere other than in a style sheet. However, with the uneven style-sheet support in browsers, many Web designers make the font specifications both in the style sheet and the document, to increase the chances that the user's browser will understand and support the designer's selections.

Using the *basefont* Element

You can specify the font typeface, size, and color in one swoop using the `basefont` element, which is placed at the top of the page, immediately after the `body` element. The `basefont` element can take three attributes to manipulate the font: `size`, `color`, and `face`. The `face` attribute is short for *typeface*. In use, the element might be written as:

<basefont size="+1" face="Arial" color="blue">

This configuration would result in the Arial typeface being used at one size larger than the default configuration, and all text would be colored blue. The +1 value of the size attribute is known as a *relative* font size. If the default font size were set at 10pt and the next available size in that typeface is 12pt., the display for the document would use 12pt. If you wanted to reduce the font size, you'd indicate a negative number, perhaps `size="-2"`. Keep in mind, though, that with a 10pt. or 12pt. default on many pages, a two-step downgrade would result in some pretty tiny type!

Warning You may see some designers using unsigned numbers as values for font sizes. These are called *absolute* sizes and should be avoided. Each browser has its own font size associated with each number, with 1 being associated with the smallest size and larger point sizes associated with higher numbers. You will run into a problem when a user has set a particularly large font as their default, say at 24pt., because of reading difficulty. If you use relative font sizing, a one-step decrease might only reduce it to a 20pt font. If you use absolute sizing and choose the value of 2, that user is likely to get between an 8pt. and 10pt. font size, enough to make the text unusable!

The `basefont` element sample used one of the basic-named color values. As with any other color-selecting attribute, hex values may be used and are actually encouraged. If you need a refresher on hex color coding, you can read more about it in Chapters 3 and 16.

Finally, the `face` attribute in the sample contained the typeface Arial. Designers are encouraged to list more than one font in their selections to cover the possibility that a user doesn't have the font specified on their system. By providing an alternative selection, the designer can still retain some control over the look of the page, even if the typeface used isn't their first choice. The typeface Arial is actually a Windows-based type. The Mac has a nearly identical type known as Helvetica. Both of these tyepfaces are sans-serif fonts, which as you learned earlier means they don't have flourishes or decorations at the end of each character. The smart designer then specifies both font faces, along with a third option, as shown here:

<basefont size="+1" face="Arial, Helvetica, sans-serif"

color="blue">

The third choice, `sans-serif`, tells the browser, "If you don't have Arial and Helvetica, at least display this in a sans-serif font." Inserting either `serif` or `sans-serif` at the end of your font `face` selection is always a good idea.

Using the *font* Element

The `font` element differs from the `basefont` element in only a single aspect: basefont sets the font choices for the entire document, and the `font` element sets the font choices for the data contained within the `font` tags.

Serif versus Sans-Serif Typefaces

Serif typefaces are those which have small extra strokes on the ends of the lines which make up the letters. One example of a serif typeface is `Courier`. Another example is Palatino. Do you see the extra strokes? If not, look at the bottom of the letter *r* in the `Courier` example. It could have just ended with the vertical line, which makes up the stem of the *r*, but a small horizontal line is at the bottom. That's the serif.

An example of a sans-serif typeface (a typeface with no serifs) is **Helvetica**. Note that no little lines are at the ends of the main lines that make up the letters. Another example is **Franklin Gothic Book**.

Serif fonts tend to be easier to read than sans-serif fonts because the serifs lead the reader's eyes to the next letterform. That is, the transition from one letter to the next is smoother. That's not to say that sans-serif fonts should not be used—they can be quite useful, especially to provide contrast to serif fonts, which are more common in text. You just might not want to use them for pages and pages of dense reading. You can see this principle in action if you look at any book that contains a large amount of text.

The `font` element has one important restriction: it's an inline element, meaning that it can only appear within block elements or other inline elements. So if you want to change the look of a paragraph, the `font` element must be contained *within* the paragraph, such as the one seen here:

```
<p><font face="Geneva" size="-1" color="green">The quick brown

fox jumped over the lazy dog.</font></p>
```

You may run into document source code that sets a font element near the beginning of the page and closes it down at the bottom. Many browsers will compensate for this error and change the font for all the elements in between. However, because `font` is an inline element, such construction isn't valid and cannot be relied upon to work properly. If you want the bulk of your document to have the same font properties, use the `basefont` element. Then as needed, change the styling using the `font` element.

Fonts in Style Sheets
In Chapter 7, you learned quite a bit about the use of style sheets to provide visual presentation instructions. As a reminder, the style-sheet properties used to manipulate font presentation are shown in Table 15.1.

Table 15.1: Common Font Properties and Their Usage

FONT PROPERTY	USAGE
`font-family`	Is equivalent to the `face` attribute of the `font` element.
`font-size`	Adjusts the size of the font, may be stated in points (pt.), percentages, or specific sizes, such as medium and large.
`font-style`	Changes the slant of the font. Values are limited to `italic` or `oblique`.
`font-variant`	Is unevenly supported; a variation called `small-caps` can be used to replace lowercase letters with lowercase-sized versions of capital letters.
`font-weight`	Covers the density of the type, what you'd normally refer to as *bold*.
`font`	A catchall property that takes a space-delimited list of property values described in the other properties in this table.

Browser Support for Font Changes
Browsers will go through each font face specified in your list until it finds one on the user's computer; if it fails entirely, it will substitute the user's selected default font. As you saw in the `basefont` element example, some operating systems have very similar fonts with different names. Table 15.2 shows a few common fonts, with the names used for the Windows and Mac versions of each.

Table 15.2: Font equivalents

WINDOWS FONT	MACINTOSH FONT
Arial	Helvetica
Times	Times New Roman
Courier	Courier New
Century Schoolbook	New Century Schoolbook

Two attempts have been made at developing systems to provide fonts dynamically to Web users. Neither has gained much footing in user support, and not surprisingly, the competing methods come from Microsoft and Netscape. Because they are not widely supported or used, I won't go into detail about them here; however, if you'd like to read about them on the Web, you can do so at the following URLs:

- WWW **Microsoft Web Embedding Fonts Tools (WEFT)** Version 2 is available and version 3 is in beta. Find them at `http://www.microsoft.com/typography/web/embedding/weft2/default.htm`.
- WWW **Netscape Compatible Dynamic Fonts** A well-written tutorial from guru Danny Goodman can be found at `http://developer.netscape.com/viewsource/goodman_fonts.html`.

Summary

In this chapter, you learned about supporting fonts on the Web, specifically, about setting a default font for an entire page, modifying a small portion of text with a new font, and using style sheets to define font properties. Additionally, you took a quick tour of typography and the differences between serif and sans-serif fonts.

Chapter 16: Color on the Web

Overview

Color is one of the most exciting elements of Web design. In traditional publishing, color is usually reserved for high-cost publications due to the labor-intensive production methods discussed later in this chapter. But on the Web, adding color is relatively easy for even the novice Web designer. By following a few guidelines, you can employ vibrant colorful graphics and photographs to help deliver your message.

Producing effective graphics to enhance the appeal of your Web-site design is extremely important. In this chapter, you will learn the basics of color and how to work with it. You'll also learn how to choose colors effectively for your designs, and I'll give you some guidelines for producing high-quality color images for the Web.

This chapter covers the following topics:

- A color primer
- Using color in your designs
- Color and your computer

A Color Primer

People are surrounded by color every day—from the colors they decorate their homes with to the advertising they see on television. In most cases, people don't think much about the colors around them unless they are purchasing a gallon of paint or a new article of clothing. Suddenly, the choice of color becomes an enormously important decision. How do you select colors? What colors go together? Do they match? Does it matter?

Before you can choose effective colors and color combinations, it helps tremendously to have a basic understanding of color theory. Once you have a grasp of how colors relate, choosing colors becomes much easier.

The easiest way to develop familiarity and comfort with color is to spend a bit of time studying the color wheel. The color wheel is the most common tool used for introducing basic color principles because it's fairly intuitive. Colors change incrementally as you go around the wheel.
The wheel, as shown in Figure 16.1, has 12 *hues* (hue is a gradation of color). In particular, this wheel illustrates the three primary hues—red, yellow, and blue. To achieve a new color, you mix various percentages of each primary hue.

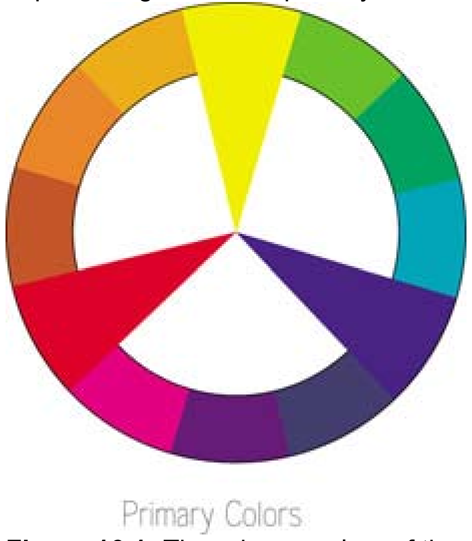


Figure 16.1: The primary colors of the color wheel

Tip To get the most out of this chapter, you need to see the figures in color. So take your book to the computer and place the companion CD in your CD drive. You'll find all the figures in the Chapter 16 folder. Trust me—you're in for a treat.

Mixing equal portions of two primary colors produces a *secondary color*. (Think back to kindergarten and finger painting!) For example, if you mix red and yellow you will get orange. Orange is found on the wheel halfway between yellow and red. The same relationship exists between yellow and blue (green) and between red and blue (violet).
But what about the yellow-orange color? If you mix the primary yellow with the secondary orange, you get yellow-orange, which I refer to as a *tertiary color*. You can see tertiary colors all around the wheel.

Color wheels do not always have only 12 hues. Depending on what you are trying to achieve, they can contain as few as 3 colors (the primaries) to as many as 30 colors. Regardless, color proceeds in an orderly progression around the wheel, and you create the wheel by mixing the primary hues.

Tip A color wheel is a terrific tool when you're designing Web sites or any other graphics, particularly if it shows colors in varying tints. You can obtain a cheap color wheel at nearly any art-supply store.

Figure 16.2 shows some adjacent colors on the color wheel. Usually, three or four colors are used in adjacent-color palettes. Such color schemes are an excellent choice for the novice designer—they are predictable and easy to select.

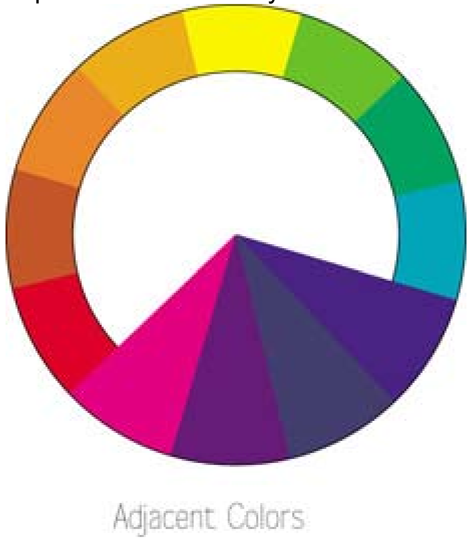


Figure 16.2: Adjacent colors on the color wheel

Tip In design terms, a palette is a range of colors to be used in a composition. For instance, a decorator's palette for a nursery would likely have pinks, blues, and soft yellows or greens in it, but not purple, chartreuse, or other dark and jarring colors.

Complementary colors are colors that are opposite on the color wheel. Figure 16.3 shows blue and orange. These combinations of colors are certainly attention grabbing; however, as you will see later in this chapter, complementary color schemes can also produce undesirable effects when used excessively.

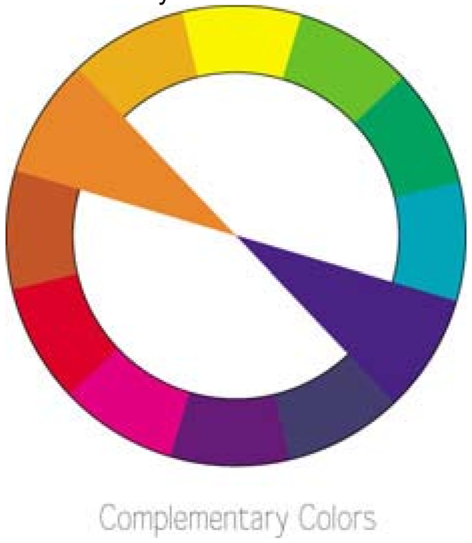


Figure 16.3: Complementary colors are very eye catching when used together.

WWW Figure 16.4 shows an example of *triads*—colors that are equidistant on the color wheel. Here, you can see orange, green, and violet. The primary colors (red, yellow, and blue) are another example of triad colors. Palettes based on triads are a popular design choice. ZDNet's home page uses the red/yellow/blue triad (`http://www.zdnet.com`), as does AltaVista (`http://www.altavista.com`).

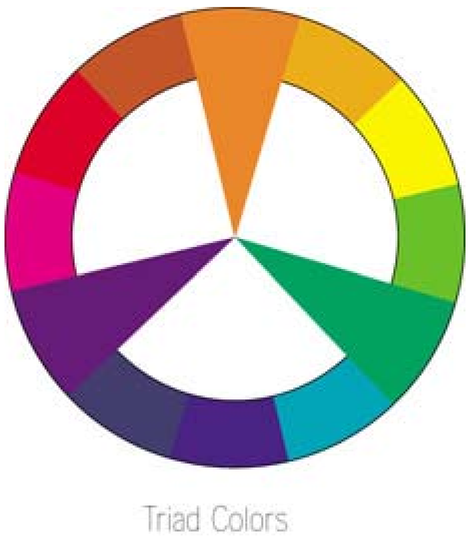


Figure 16.4: Triad colors

In the previous color wheel examples, I was looking at fully *saturated* hues. Saturation refers to the intensity of a particular hue. An intense hue is fairly pure in color. For example, a fully saturated red is bright and intense, not the red of a weathered barn or brick. Fully saturated hues are simple colors, much like a child's first set of crayons.

Figure 16.5 shows the color wheel with full saturation on the outer edge and increasingly lower levels of saturation toward the center. A fully saturated hue is the most vibrant and intense. Less saturated colors are grayer and duller, though they retain the original hue.

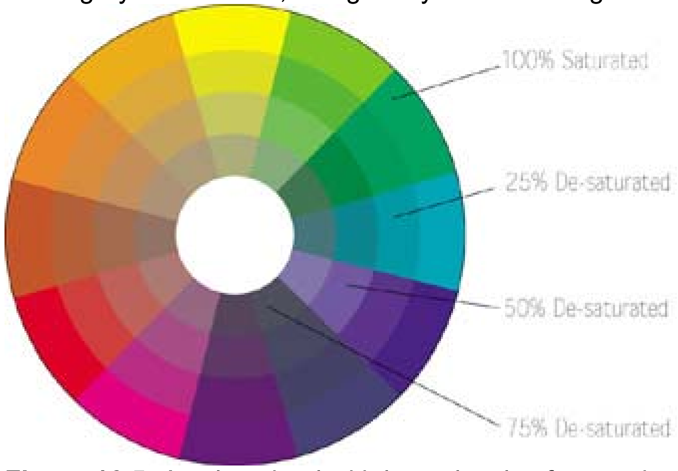


Figure 16.5: A color wheel with lower levels of saturation near the center

The last "color basic" to consider is *value* or *brightness*. In Figure 16.6, you can see a stripe of blue that ranges from a high value at the left to a low value on the right. Value describes brightness, or light and dark, in a color and determines a hue's placement between black and white on a scale.

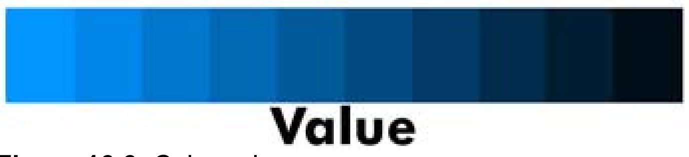


Figure 16.6: Color values

Color Theory: A Complex Topic

The color wheel represents only one school of thought on color theory. Although it is the simplest approach to understanding the use of color in design, it isn't ideal for explaining color on computer monitors. The color wheel was developed for mixing pigments to paint canvas.

Color theory is a complex topic, and experts from diverse fields have addressed it in great detail—from the physiology of the human eye and perception of color to the mechanics of color monitors. If you are interested in learning more about color theory, check out the following URLs:

- Basic Color Theory, by Douglas Barkey, is a good page with much information. Find it at http://www.coa.edu/campuslife/work/courseprojects/polcom/colort.html.

- Another excellent color tutorial with additional resources is Poynton's Color FAQ. It is probably the most comprehensive documentation regarding the creation of computer graphics and digital imaging. It is heavy reading for the truly committed. Find it at http://inforamp.net/~poynton/notes/colour_and_gamma/ColorFAQ.html.

Here are some books on color theory that I think are particularly helpful:

- *Designer's Guide to Color*, Ikuyoshi Shibukawa and Yumi Takahashi, Chronicle Books, 1990
- *The Psychology of Color and Design* (Second Edition), Deborah T. Sharpe, Nelson Hall Inc, 1982
- *The Color of Life*, Arthur G. Abbott, McGraw-Hill, 1947
- *Colour for Professional Communicators*, Andre Jute, Batsford Ltd., 1993

Don't confuse value with saturation. Many people do. Value is purely a measure of light and dark, not the amount of gray or dullness within a particular hue. It may be easiest for you to think of value in terms of sunlight and shadow. If an object is brightly lighted on one side and shadowed on the other, the color hasn't changed. The hue's value is different—from a high value on the lighted side to a low value on the shadowed side.

Using Color in Your Designs

Using color effectively depends on understanding and following some simple rules about how eyes perceive color and how color affects people. Now that you understand the vocabulary, I'll dive into some choices that you will have to make. What are your options when choosing colors and combinations of colors to present your message?

Color Relationships

Colors affect the way you see other colors. This phenomenon is used over and over in advertising, design, and decoration. You choose colors to enhance and emphasize other colors much as you would select a mat or a frame for a painting or a print.

Have you ever heard people describe themselves as a "winter" or an "autumn"? They are talking about their personal coloration—their skin tones, hair color, and eye color. When you purchase clothing, you automatically sense this—often unconsciously. Have you ever tried on a sweater that looked great on the shelf but appeared to make you look pale or ill? Do some colors of clothes make you look good? The reason is a color relationship between your personal coloration and color of clothes.
Figure 16.7 shows my first example of color illusions. The pink squares in the center of each of the two larger squares are the identical shade of pink; but they sure don't look the same, do they? That's because of the background color on which the pink square rests. The yellow background appears to increase the vibrancy or intensity of the pink, and the gray background seems to cause the pink to appear less intense or less saturated.

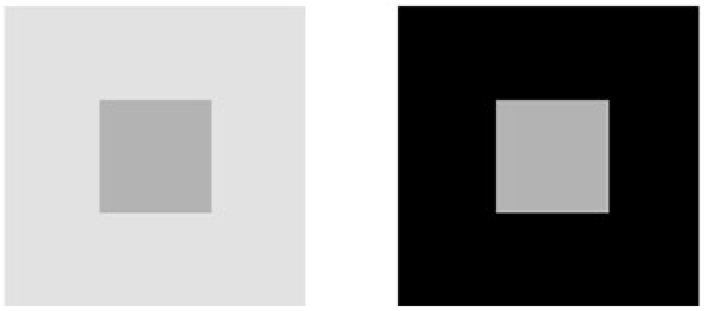


Figure 16.7: Do both pink squares have the same saturation?

Figure 16.8 illustrates a similar principle. The green in the smallest boxes is identical in both images. However, concentrate on the green on red graphic for a few seconds—does the small box appear to be vibrating? Does it appear to move toward you and then away from you? This is an optical illusion common when two complementary colors are next to each other. (Complementary colors are opposite each other on the color wheel, as you read in the previous section.)

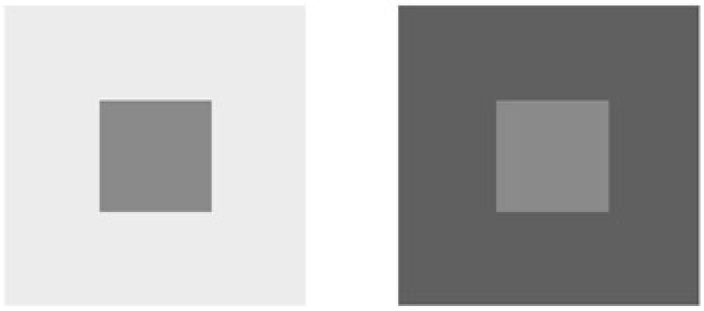


Figure 16.8: Optical illusions are common with complementary colors.

If the previous image didn't seem to vibrate, the next certainly will! Figure 16.9 illustrates how readability of type can be affected by its presentation against a background color. In this case, the two hues are both high intensity and pure—your eye has a difficult time focusing on the type in the image.

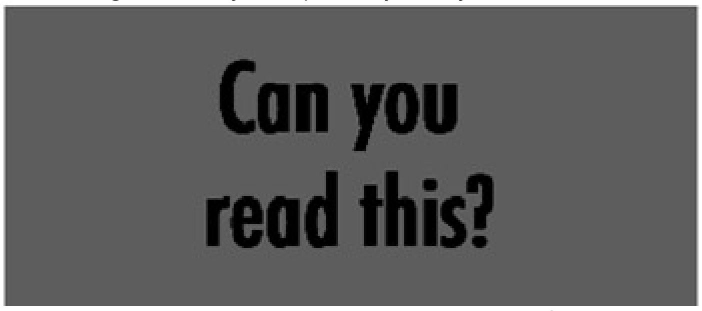


Figure 16.9: The background color competes for attention with the text color.

A complementary color scheme grabs your visitor's attention, but it is also hard on the visitor's eyes. If you use a complementary color scheme, do so sparingly in small areas. Overdosing your visitor with vibrating color is likely to be confusing and annoying.

Figure 16.10 illustrates another way background color can affect the view of the foreground color. In this example, the red- violet lines and boxes in the graph are on a graduated background with colors from blue-violet to red. The red-violet line appears to change color as a result of its relationship with the graduated background; however, the line and the boxes are the identical color.

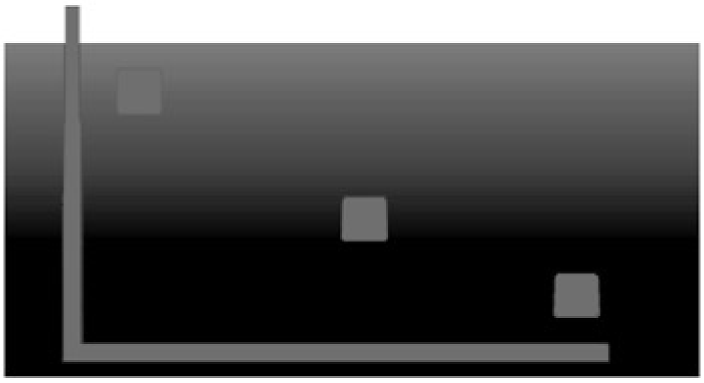


Figure 16.10: Background color can impact how other objects are seen.

Using high-contrast hues can improve the readability of type on a color background. Figure 16.11 compares white text against a light green and a dark green background. The only difference in the colors is the value of the green. The type is much easier to read against the darker green because of the high contrast.

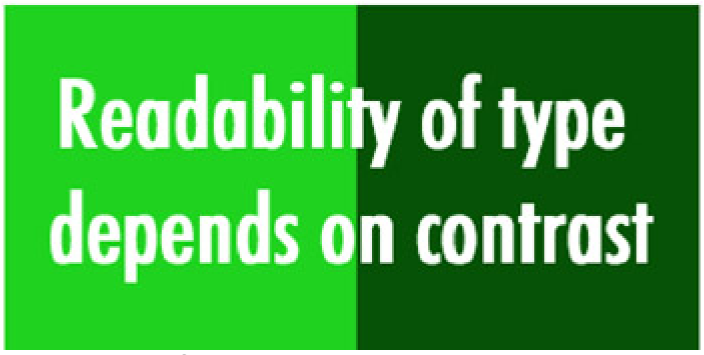


Figure 16.11: Choosing high contrast between text and background colors enhances readability.

The Psychological Impact of Color

Color has a strong impact on people emotionally. Consider the following phrases:

- They're seeing red right now.
- I'm having a blue day.
- She has a yellow streak.
- He's still rather green.
- She told him a little white lie.

Each of these statements is immediately understandable without a description of the situation or the emotions involved. Color has the same effect on the viewer—it sets a mood or a feeling.
People commonly refer to colors as *warm* and *cool*. Figure 16.12 shows where these "temperatures" of color fall on the color wheel.

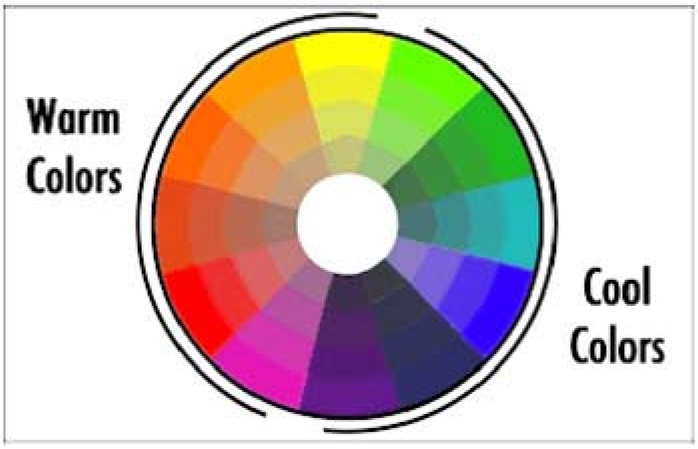


Figure 16.12: Colors arranged by "temperature"

Warm Colors

Reds, oranges, and yellows are warm colors. Warm colors appear to move forward in an image, and they convey a sense of energy.

A viewer's response to color depends on the saturation or intensity level of the hue. Bright red is known to attract attention—often unwelcome attention if you happen to be the owner/driver of a sporty cherry-red car. Some studies have shown that red cars tend to be driven faster and are stopped more often by the police for speeding than any other color car. Consider that red is often used for emergency vehicles, for flashing lights on ambulances, and for stop signs—all situations that require prompt attention.
Figure 16.13 illustrates a use of warm colors; this time, the hues are much less intense than the proverbial cherry-red automobile and have less energy overall. The sunset colors of yellow, pinks, and oranges still have the ability to evoke energetic emotions in the viewer. Does it feel romantic to you? You have a strong urge to watch it and see the sun drop below the horizon. The average viewer finds it hard to withdraw from this image.

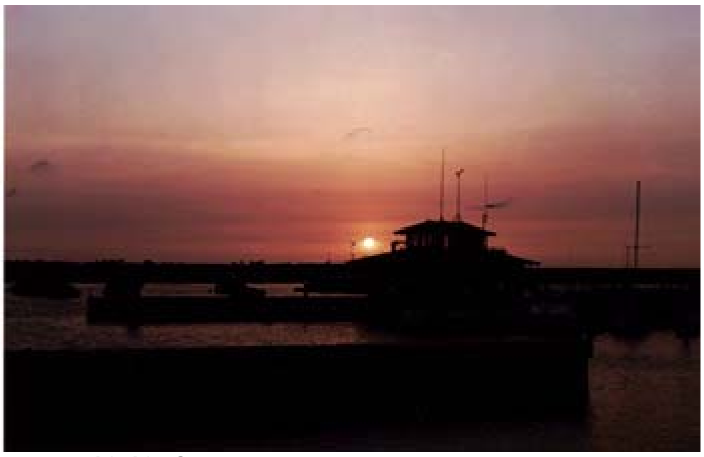


Figure 16.13: Sunsets and their colors inspire romantic thoughts.

Cool Colors

The cool colors, purples, blues, and greens, appear to recede into the background and evoke a feeling of released tension.

Figure 16.14 shows a scene at Yosemite National Park. The blues of the skies and water, the blue-white of the snow, and the greens of the landscape are low-intensity, cool colors. Your eye finds these images soothing. Do you feel a sigh of relaxation as you look at this image? Do you feel an urge to kick off your shoes, pull up a hammock, and relax in the peace of this landscape?

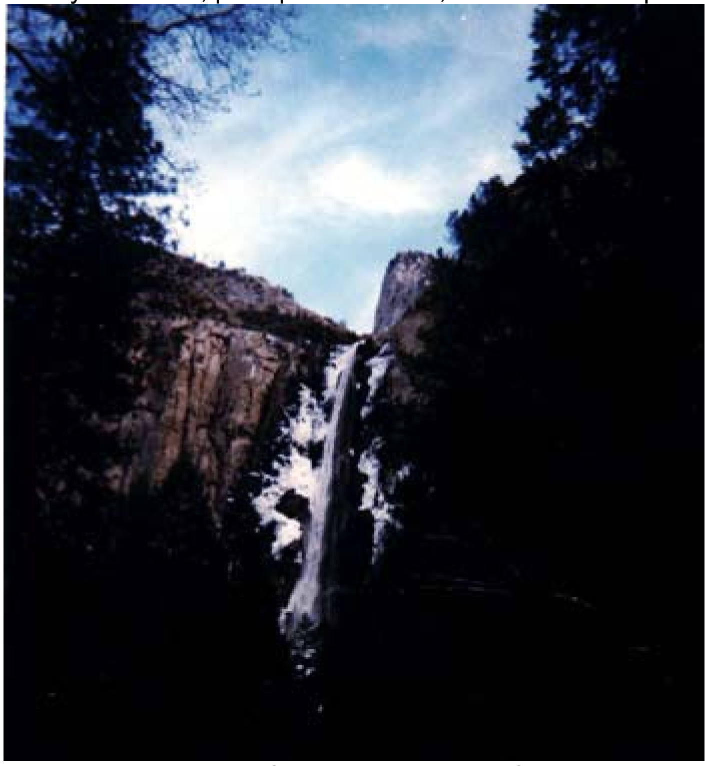


Figure 16.14: Do you find the cool colors of this scenic spot relaxing?

In Figure 16.15, another example of cool colors, the greens are higher intensity than the hues in the previous photo, but they still evoke a feeling of restfulness. While your eye travels around the image to take in the various details, you can feel the relaxation. Would you believe that this tropical retreat is in the middle of Manhattan?

Figure 16.15: A lush green set of palm trees

Choosing Colors

So, now that you've read about color and its emotional aspects, what good is it? How does this apply to Web-page design?
The impact of color in graphic design is easy to see with a little bit of study. Designers use color to draw your attention to specific elements, in the same way that they use the basic art principles outlined in Chapter 14. By combining those principles and the tips for using color in this chapter, you can create attention-grabbing pages that will keep your visitors at your Web site, hungry for more!

Tip Once you've read through this chapter, take some time to look at information and advertising sources both in print and on the Web to see if you can understand how color is being used. By looking at other designs with an analytical eye, you will begin to see patterns in the use of color that you can utilize for your own designs.

For the beginning designer, choosing colors feels like an overwhelming task. There are so many choices! Keep your color schemes simple in the beginning. Experiment with the color wheel, apply the color basics I have provided here, and study the color schemes used by other designers. As you experiment, you will begin to get a feel for working with color.

Color is subjective for most people. Although the rules may tell you that yellow is a happy color, all the rules in the world aren't going to make yellow feel happy for you if you personally don't like yellow. Start with colors that you like and trust your gut reactions. After all, when you design your first Web site, you are likely looking to please yourself—not follow a rulebook!
Figure 16.16 shows a fictitious company's Web site done in a one-color scheme with varying tints of violet and black for the text. This is a simple color scheme that can be surprisingly versatile and easy for the novice designer to work with. Graphical text is always white, and buttons are one of two tints of violet. The background is a very light violet. You can easily make color decisions as you add features to the site.

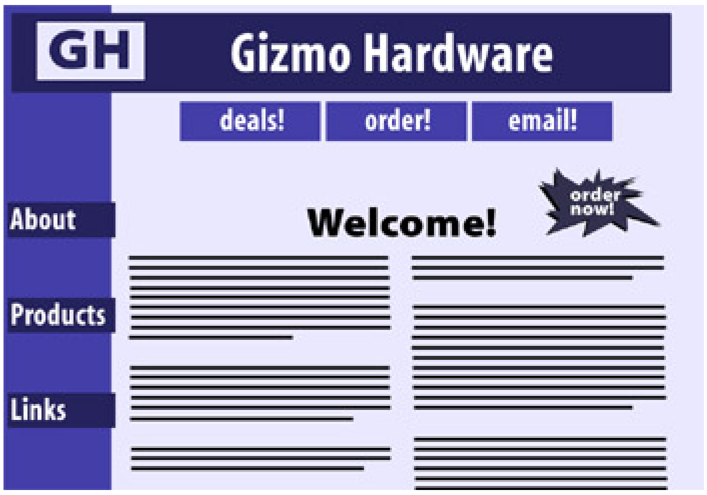


Figure 16.16: A simple single-color design scheme

Figure 16.17 shows the look you get when you add violet's complementary color—orange. The graphical text really jumps out at you now, doesn't it? The complementary color is used for text, and a very light tint is used for the background. For zing, you could also add orange bullets to this page.

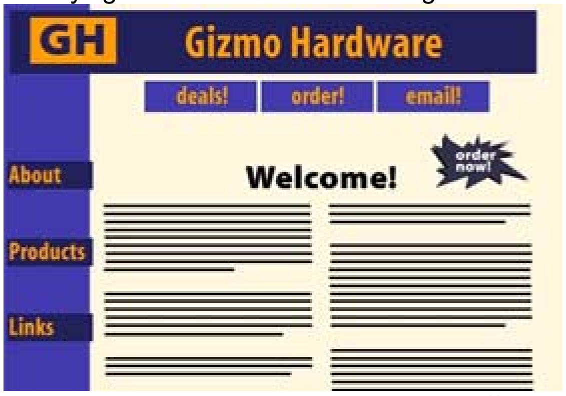


Figure 16.17: The orange really jumps off the screen.

Tip Earlier, I warned against using complementary colors inappropriately. The example in Figure 16.17 illustrates the use of complementary colors; however, the violet is not a fully saturated hue. The toned-down violet tint reduces the potential for a "vibrating" image, which is hard to look at, particularly on a computer monitor. The vibration effect is most often seen with two very pure hues of complementary colors. Let your eye be the judge if you are considering a complementary color scheme.

In Figure 16.18, I've used a color triad made up of tints of the primary colors (red, yellow, and blue) and white as a background. This color scheme is still relatively easy to work with. The stripe on the left side of the page is a less saturated tint of the blue.

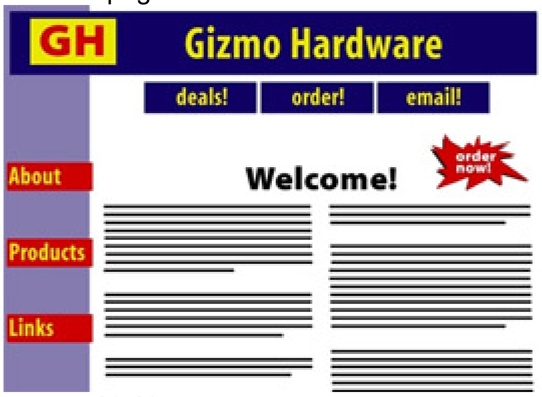


Figure 16.18: Incorporating a color triad into the design

As a designer, you will need to choose which tints of the hues to use primarily based on what feels good to you. You may notice that the blue chosen for the banner and buttons is not a pure hue. Pure hues are intense and are hard on the eyes. Experiment with hue intensity levels until you find a combination that has sufficient contrast to be readable but isn't difficult to look at on your computer monitor.

Figure 16.19 shows an adjacent-hues color scheme—red-orange through yellow. Again, the designer has used varying tints and saturation levels to achieve a pleasing combination of colors and sufficient contrast for legibility of graphical text. When you use this color scheme, you'll need to experiment to find the right colors.

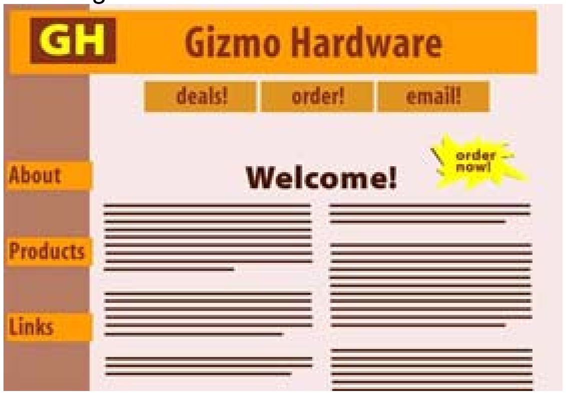


Figure 16.19: The same design using an adjacent palette

For a rather intense and dramatic experience, take a look at Figure 16.20. It uses nearly all colors on the wheel at full intensity, plus black and white. This color scheme can work; however, the process for deciding which color to use for each element takes careful consideration and a great deal of experimenting. The site is bright and has a great deal of energy. Subsequent pages will be a serious challenge. Which color should dominate throughout the site? Should secondary pages be as dramatic or perhaps toned down a bit?

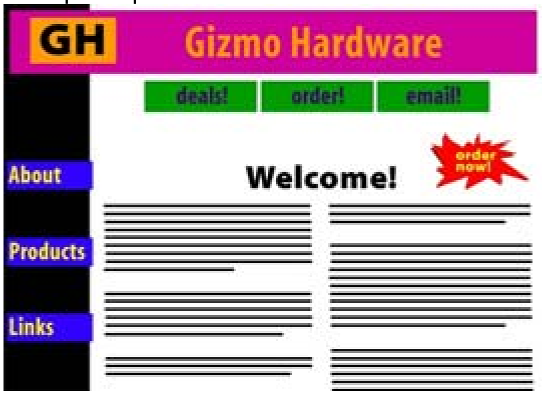


Figure 16.20: Bright colors are eye catching, though care must be taken not to overwhelm.

Using Color in Your Designs

Using color effectively depends on understanding and following some simple rules about how eyes perceive color and how color affects people. Now that you understand the vocabulary, I'll dive into some choices that you will have to make. What are your options when choosing colors and combinations of colors to present your message?

Color Relationships

Colors affect the way you see other colors. This phenomenon is used over and over in advertising, design, and decoration. You choose colors to enhance and emphasize other colors much as you would select a mat or a frame for a painting or a print.

Have you ever heard people describe themselves as a "winter" or an "autumn"? They are talking about their personal coloration—their skin tones, hair color, and eye color. When you purchase clothing, you automatically sense this—often unconsciously. Have you ever tried on a sweater that looked great on the shelf but appeared to make you look pale or ill? Do some colors of clothes make you look good? The reason is a color relationship between your personal coloration and color of clothes.
Figure 16.7 shows my first example of color illusions. The pink squares in the center of each of the two larger squares are the identical shade of pink; but they sure don't look the same, do they? That's because of the background color on which the pink square rests. The yellow background appears to increase the vibrancy or intensity of the pink, and the gray background seems to cause the pink to appear less intense or less saturated.

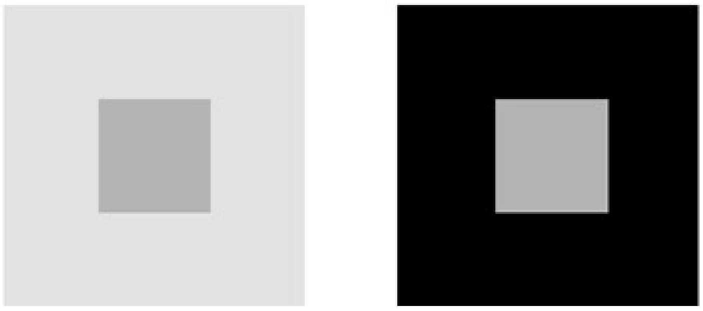


Figure 16.7: Do both pink squares have the same saturation?

Figure 16.8 illustrates a similar principle. The green in the smallest boxes is identical in both images. However, concentrate on the green on red graphic for a few seconds—does the small box appear to be vibrating? Does it appear to move toward you and then away from you? This is an optical illusion common when two complementary colors are next to each other. (Complementary colors are opposite each other on the color wheel, as you read in the previous section.)

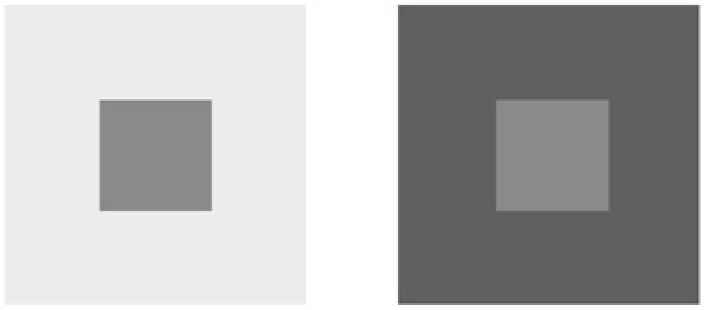


Figure 16.8: Optical illusions are common with complementary colors.

If the previous image didn't seem to vibrate, the next certainly will! Figure 16.9 illustrates how readability of type can be affected by its presentation against a background color. In this case, the two hues are both high intensity and pure—your eye has a difficult time focusing on the type in the image.

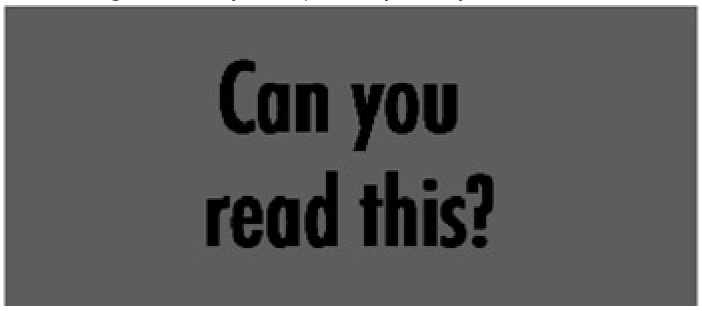


Figure 16.9: The background color competes for attention with the text color.

A complementary color scheme grabs your visitor's attention, but it is also hard on the visitor's eyes. If you use a complementary color scheme, do so sparingly in small areas. Overdosing your visitor with vibrating color is likely to be confusing and annoying.

Figure 16.10 illustrates another way background color can affect the view of the foreground color. In this example, the red- violet lines and boxes in the graph are on a graduated background with colors from blue-violet to red. The red-violet line appears to change color as a result of its relationship with the graduated background; however, the line and the boxes are the identical color.

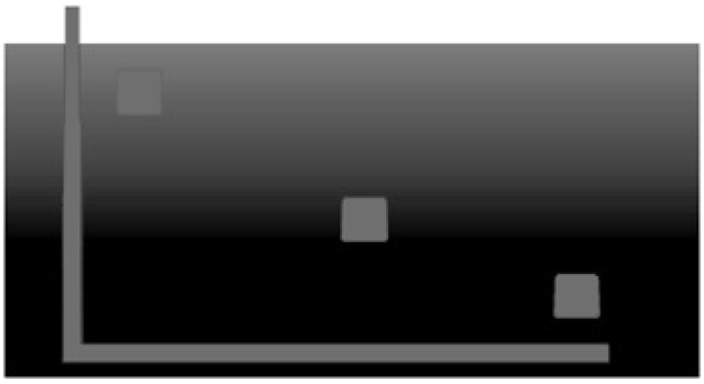


Figure 16.10: Background color can impact how other objects are seen.

Using high-contrast hues can improve the readability of type on a color background. Figure 16.11 compares white text against a light green and a dark green background. The only difference in the colors is the value of the green. The type is much easier to read against the darker green because of the high contrast.

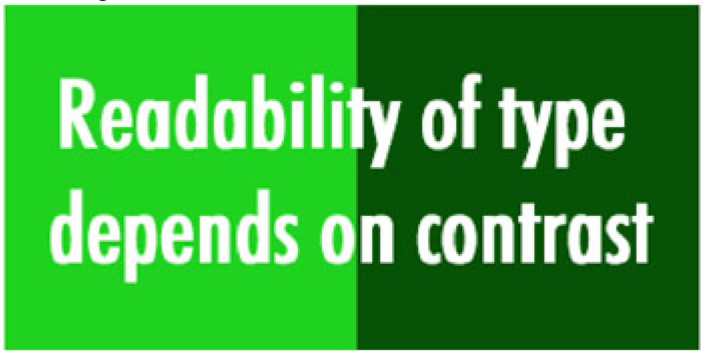


Figure 16.11: Choosing high contrast between text and background colors enhances readability.

The Psychological Impact of Color

Color has a strong impact on people emotionally. Consider the following phrases:

- They're seeing red right now.
- I'm having a blue day.
- She has a yellow streak.
- He's still rather green.
- She told him a little white lie.

Each of these statements is immediately understandable without a description of the situation or the emotions involved. Color has the same effect on the viewer—it sets a mood or a feeling.
People commonly refer to colors as *warm* and *cool*. Figure 16.12 shows where these "temperatures" of color fall on the color wheel.

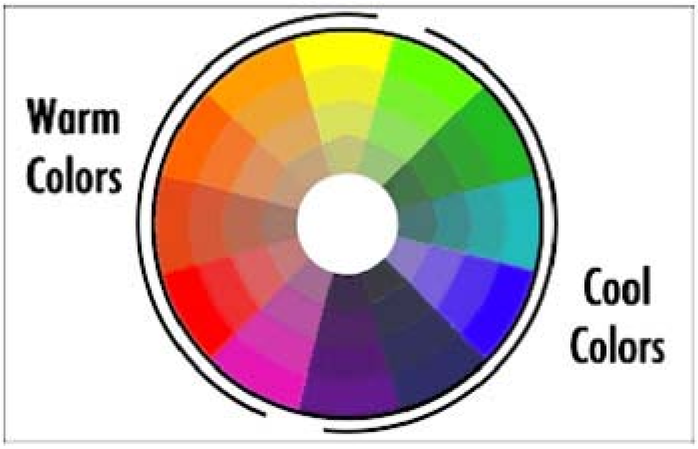


Figure 16.12: Colors arranged by "temperature"

Warm Colors

Reds, oranges, and yellows are warm colors. Warm colors appear to move forward in an image, and they convey a sense of energy.

A viewer's response to color depends on the saturation or intensity level of the hue. Bright red is known to attract attention—often unwelcome attention if you happen to be the owner/driver of a sporty cherry-red car. Some studies have shown that red cars tend to be driven faster and are stopped more often by the police for speeding than any other color car. Consider that red is often used for emergency vehicles, for flashing lights on ambulances, and for stop signs—all situations that require prompt attention.
Figure 16.13 illustrates a use of warm colors; this time, the hues are much less intense than the proverbial cherry-red automobile and have less energy overall. The sunset colors of yellow, pinks, and oranges still have the ability to evoke energetic emotions in the viewer. Does it feel romantic to you? You have a strong urge to watch it and see the sun drop below the horizon. The average viewer finds it hard to withdraw from this image.

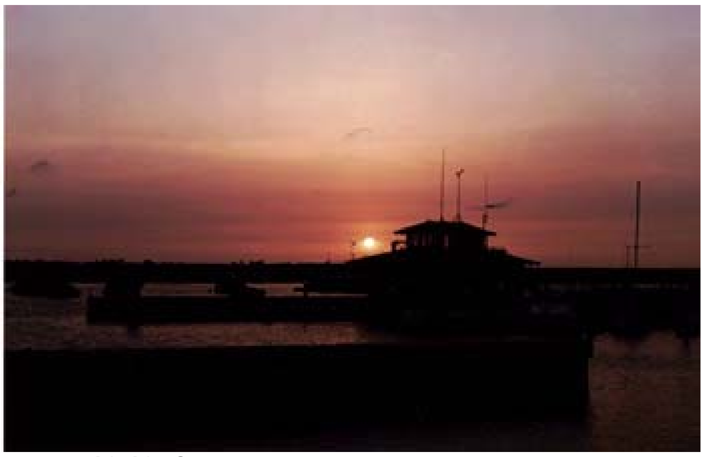


Figure 16.13: Sunsets and their colors inspire romantic thoughts.

Cool Colors

The cool colors, purples, blues, and greens, appear to recede into the background and evoke a feeling of released tension.
Figure 16.14 shows a scene at Yosemite National Park. The blues of the skies and water, the blue-white of the snow, and the greens of the landscape are low-intensity, cool colors. Your eye finds these images soothing. Do you feel a sigh of relaxation as you look at this image? Do you feel an urge to kick off your shoes, pull up a hammock, and relax in the peace of this landscape?

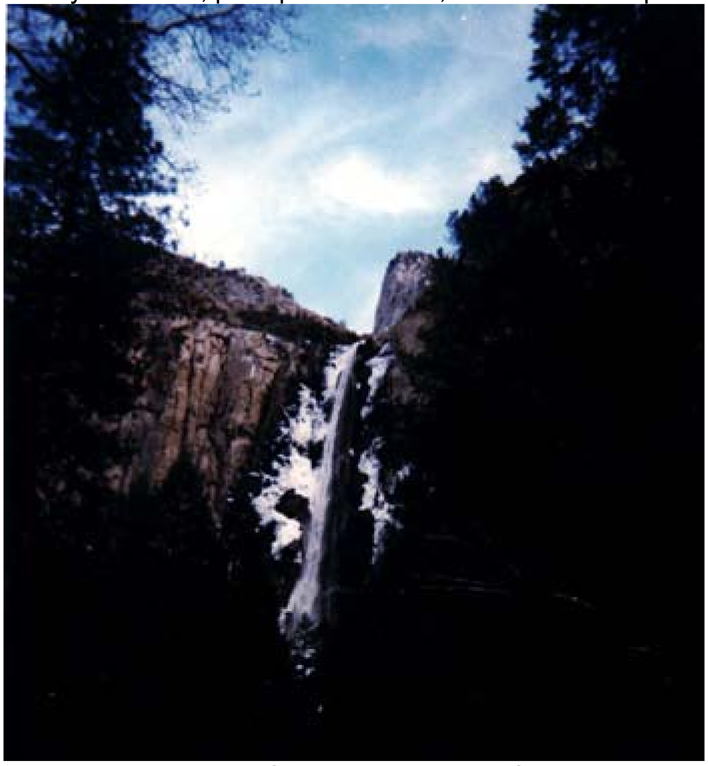


Figure 16.14: Do you find the cool colors of this scenic spot relaxing?
In Figure 16.15, another example of cool colors, the greens are higher intensity than the hues in the previous photo, but they still evoke a feeling of restfulness. While your eye travels around the image to take in the various details, you can feel the relaxation. Would you believe that this tropical retreat is in the middle of Manhattan?

Figure 16.15: A lush green set of palm trees

Choosing Colors

So, now that you've read about color and its emotional aspects, what good is it? How does this apply to Web-page design?
The impact of color in graphic design is easy to see with a little bit of study. Designers use color to draw your attention to specific elements, in the same way that they use the basic art principles outlined in Chapter 14. By combining those principles and the tips for using color in this chapter, you can create attention-grabbing pages that will keep your visitors at your Web site, hungry for more!

Tip Once you've read through this chapter, take some time to look at information and advertising sources both in print and on the Web to see if you can understand how color is being used. By looking at other designs with an analytical eye, you will begin to see patterns in the use of color that you can utilize for your own designs.

For the beginning designer, choosing colors feels like an overwhelming task. There are so many choices! Keep your color schemes simple in the beginning. Experiment with the color wheel, apply the color basics I have provided here, and study the color schemes used by other designers. As you experiment, you will begin to get a feel for working with color.

Color is subjective for most people. Although the rules may tell you that yellow is a happy color, all the rules in the world aren't going to make yellow feel happy for you if you personally don't like yellow. Start with colors that you like and trust your gut reactions. After all, when you design your first Web site, you are likely looking to please yourself—not follow a rulebook!
Figure 16.16 shows a fictitious company's Web site done in a one-color scheme with varying tints of violet and black for the text. This is a simple color scheme that can be surprisingly versatile and easy for the novice designer to work with. Graphical text is always white, and buttons are one of two tints of violet. The background is a very light violet. You can easily make color decisions as you add features to the site.

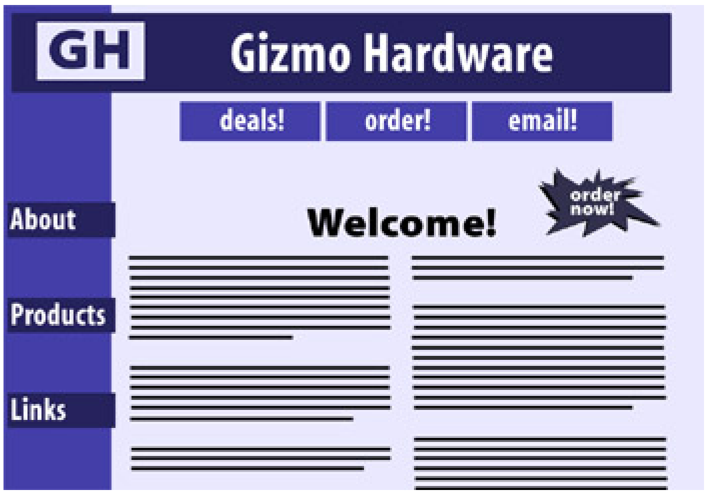


Figure 16.16: A simple single-color design scheme

Figure 16.17 shows the look you get when you add violet's complementary color—orange. The graphical text really jumps out at you now, doesn't it? The complementary color is used for text, and a very light tint is used for the background. For zing, you could also add orange bullets to this page.

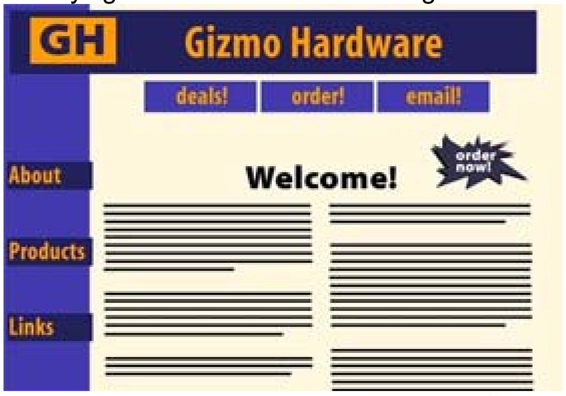


Figure 16.17: The orange really jumps off the screen.

Tip Earlier, I warned against using complementary colors inappropriately. The example in Figure 16.17 illustrates the use of complementary colors; however, the violet is not a fully saturated hue. The toned-down violet tint reduces the potential for a "vibrating" image, which is hard to look at, particularly on a computer monitor. The vibration effect is most often seen with two very pure hues of complementary colors. Let your eye be the judge if you are considering a complementary color scheme.

In Figure 16.18, I've used a color triad made up of tints of the primary colors (red, yellow, and blue) and white as a background. This color scheme is still relatively easy to work with. The stripe on the left side of the page is a less saturated tint of the blue.

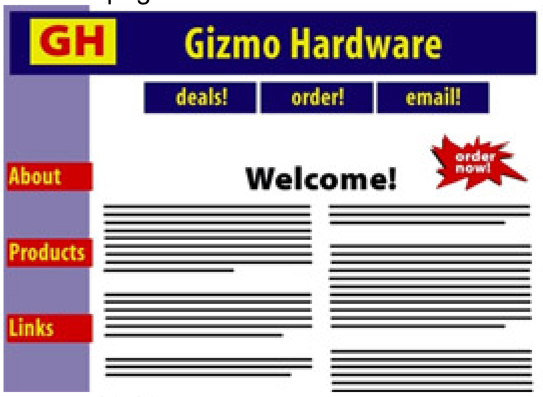


Figure 16.18: Incorporating a color triad into the design

As a designer, you will need to choose which tints of the hues to use primarily based on what feels good to you. You may notice that the blue chosen for the banner and buttons is not a pure hue. Pure hues are intense and are hard on the eyes. Experiment with hue intensity levels until you find a combination that has sufficient contrast to be readable but isn't difficult to look at on your computer monitor.

Figure 16.19 shows an adjacent-hues color scheme—red-orange through yellow. Again, the designer has used varying tints and saturation levels to achieve a pleasing combination of colors and sufficient contrast for legibility of graphical text. When you use this color scheme, you'll need to experiment to find the right colors.

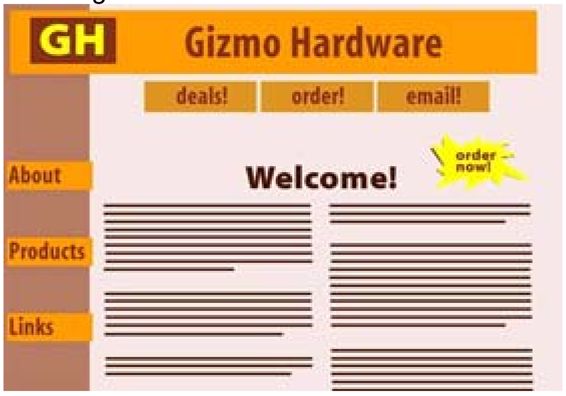


Figure 16.19: The same design using an adjacent palette

For a rather intense and dramatic experience, take a look at Figure 16.20. It uses nearly all colors on the wheel at full intensity, plus black and white. This color scheme can work; however, the process for deciding which color to use for each element takes careful consideration and a great deal of experimenting. The site is bright and has a great deal of energy. Subsequent pages will be a serious challenge. Which color should dominate throughout the site? Should secondary pages be as dramatic or perhaps toned down a bit?

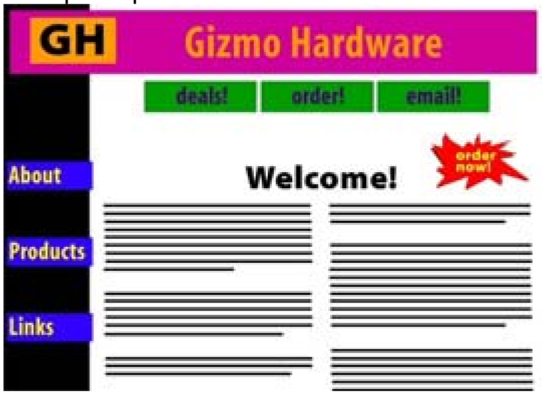


Figure 16.20: Bright colors are eye catching, though care must be taken not to overwhelm.

Summary

Designing colorful Web pages isn't difficult when you understand color theory. By following basic guidelines and experimenting, you can create vibrant and effective designs that will enhance your message and attract visitors to your pages again and again.

Chapter 17: Creating Professional Graphics

Overview

Digital images, whether graphics or photographs, provide much of the appeal for the Web. They add color, excitement, information, illustration, and fun to Web pages when they are done well. However it's not enough to design attractive images; you also need to produce technically good images that will load quickly and be consistent across computer platforms.

Have you seen graphics that just don't work for one reason or another? Huge images as the first element on a page, photos with "off" colors, strangely dithered or spotted graphics, or palette shifting? A number of common problems can occur with Web images.

In this chapter, I will provide you with the information you need to create professional graphics for your Web site—graphics that will load quickly, look good consistently across platforms, and enhance your site's image.

This chapter covers the following topics:

- Digital-image basics
- Creating professional GIFs and JPEGs
- Tools of the trade
- How those who can't draw survive

Digital-Image Basics

Before diving into the details of producing Web images, I'll take some time to explain some basics about digital images. Digital graphics come in two primary types: vector and raster.

Raster Images

Raster images are the type of images that Web designers are most familiar with. They include GIFs, JPEGs, and TIFs. Raster formats are always used for photographic or *continuous-tone* images. By looking at a magnified raster image such as the one in Figure 17.1, you can see that a raster image is made up of individual colored pixels. Each of the individual little rectangles you can see in this image is a pixel.

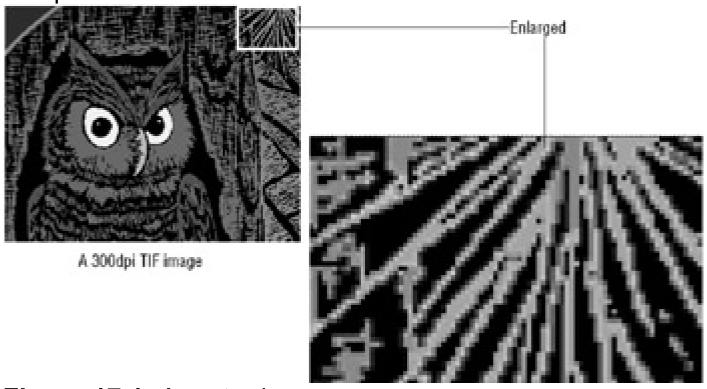


Figure 17.1: A raster image

Raster images are *resolution dependent*, meaning that they are not easily stretched or scaled to a larger size without losing image quality. As you can see in Figure 17.2, the rocket on the left is sharp and clean. This image was created for the Web as a top-of-page button. It was saved at 72ppi (pixels per inch). The image on the right is a scaled-up version of the original; it was resized by 200 percent in an image-editing program. The larger version is fuzzy and blurry looking, with some jaggy edges on the diagonal lines forming the top of the rocket. The loss of sharpness is what happens when you attempt to scale or enlarge raster images. The image is dependent on the resolution at which it was created.

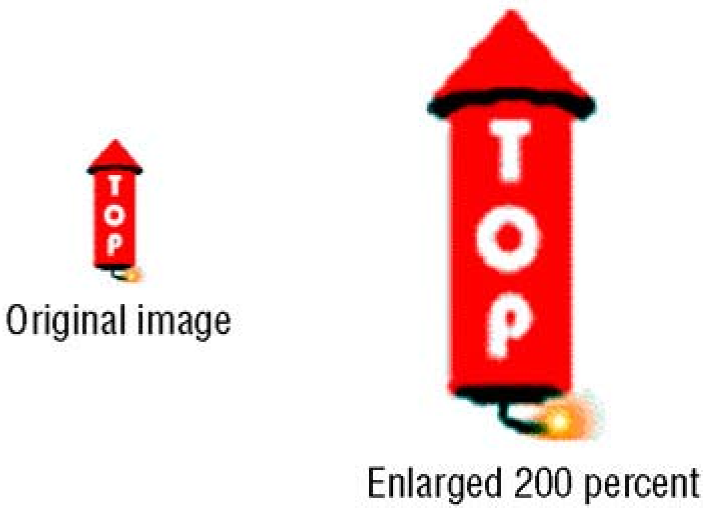


Figure 17.2: A raster image scaled up 200 percent

Raster images are often referred to as bitmap images, which isn't always completely accurate. Technically, the term *bitmap* describes a black-and-white raster graphic. They do not contain any color data.

Vector images

Vector graphics files contain a series of mathematical instructions for drawing a picture in a particular language. Postscript is the most commonly used language for generating vector graphics. The most common vector-graphics file formats are EPS, AI, and CDR. Vector graphics are most commonly used for print-media images.

Vector graphics are easily scaleable or stretchable and as a result are resolution independent. The scalability provides you with plenty of advantages—you can easily resize a vector image without loss of quality, whether you want to fit it on a business card or on a billboard. As you can see in Figure 17.3, the images on the left and the right are the same. The only difference between them is that the right-hand image is 200 percent larger than the left-hand image.

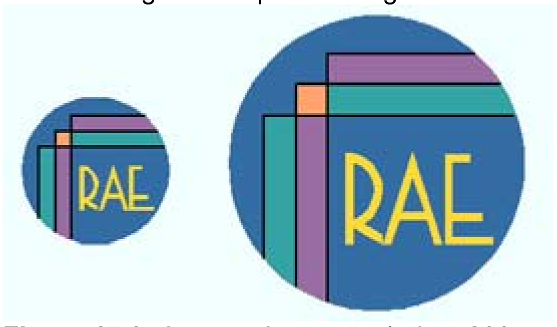


Figure 17.3: A vector image scaled up 200 percent

The problems encountered in the resizing of the raster image are not obvious here; the jaggy edges are absent. Vector images work especially well for logos, branding graphics, symbols, and other images that you may need to resize for various uses.

Vector graphics seem like the ideal type of image to use; however, you cannot directly use one on a Web page. Vector graphics can, though, be converted to raster images using graphics-conversion software.

Graphic File Formats

Although two primary types of digital images exist, you will see seemingly endless flavors of digital images, technically known as file formats. The format of a graphic file refers to the specific method of storing the image data within the file on your disk. The format tells your software how to open the image and display it on your monitor.

Web File Formats

Two primary file formats are commonly in use for Web graphics: GIF (Graphical Interchange Format) and JPEG (Joint Picture Experts Group). These two formats are supported by most graphical Web browsers and are therefore the file formats of choice for most Web development. A recent introduction, PNG (Portable Network Graphic), may eventually replace GIF; however, it is just now slowly gaining support of most Web browsers.

GIF

GIF is the most common graphic file format used on the Web. GIF files are usually small in file size and are therefore quick to download and display in a Web browser. They may contain a maximum of 256 colors, or eight-bit color. GIF is the file format of choice for Web graphics containing line art, large areas of flat color such as logos or banners, small icons, and other nonphotographic images.

In most cases, a GIF image is created from another raster file format such as TIF. The artist begins with an image created in a raster-image-editing program such as Photoshop and uses tools within the program to convert the image to a GIF file. When a file is converted to GIF, it is much smaller than the original and the file would then be displayable by Web browsers. (Later in this chapter, I will look at how to create quality GIF images using the Netscape palette.)

Note If you expect that your audience will be primarily using computer equipment that can only display 256 colors, you may want to experiment with converting your continuous-tone images to GIF format using an adaptive palette as described later in this chapter.

JPEG

JPEG is the file format of choice for displaying photographs or continuous-tone images on the Web. JPEG files use a *lossy* method of compression for full-color RGB files, which are 24 bit or 16.7 million colors. If the viewer has a monitor capable of displaying 24-bit color, JPEG images can capture the full color range of a continuous-tone image and display it in a graphical Web browser.

Lossy compression means that in reducing file size, some image data is removed from the original file. The impact of the removal of data is governed by the choices made by the user of the graphic-conversion software. Most raster-image-editing programs can convert an image to a JPEG file format. The artist begins with a 24-bit image and saves it to JPEG format after choosing image-quality levels. (Later in this chapter, I will address choosing image-quality levels appropriate to your image.) Once the file is converted to JPEG, the file size will be much smaller than the original image.

Although JPEG files are usually smaller than GIF files, JPEG does not work very well for line art, logos, or other images that contain large areas of flat color. One of the side effects of the lossy-compression algorithm is a speckling or appearance of other "artifacts" in the final image. These artifacts are usually not noticeable in continuous-tone images; however, speckles in your logo typically do not improve its appearance!

PNG

The World Wide Web Consortium (W3C), the Internet-specification organization, has endorsed PNG as the newest of graphic file formats for the Web. Although PNG is likely the best way to produce consistent Web graphics, only relatively recent versions of browsers can display PNG files. Current versions of Netscape Navigator, Microsoft Internet Explorer, and Opera support this format.

PNG offers the file-size reduction of GIF and JPEG files. It also works well across computer platforms, as it allows the graphic creator to include image information such as the palette used within the file, thus insuring consistent appearance on all systems. PNG files are expected to be about 30 percent smaller in size than most GIFS, which provides another advantage in the creation of Web graphics. The hope has been that when Web browsers better support PNG files, PNG will become the favored file format for the Web. The jury is still out on that hope, primarily because of backward-compatibility issues for those not using the most recent browsers.

Non-Web File Formats

You will create some images yourself for the Web site; for others, you will scan photographs or other printed materials. Others still will already be in digital form. The ones in digital form may not be Web ready; in other words, they may be in a file format other than GIF or JPEG.
When you have a digital image in an unfamiliar file format, you need to know how to open it and convert it to a usable file format for the Web. Table 17.1 shows some of the most common file formats, whether they are vector or raster, and what software creates the formats. Armed with this information and the appropriate software, you can tackle the task of getting your image converted and ready for the Web.

Table 17.1: Common File Formats

FILE FORMAT	FILE TYPE	NOTES
GIF	Raster	A common Web format; many programs can read it.
JPEG	Raster	A common Web format; many programs can read it.
TIF	Raster	A common format. Many programs can read it.
PSD	Raster	A file type created in Adobe Photoshop.
PCD	Raster	A file type created in Kodak PhotoCD, supported by many programs.
CPT	Raster	A file type created in Corel Photo-Paint. This file type is not widely supported.
BMP	Raster	A file type created in Microsoft Paintbrush. Many programs can read this format.
PCX	Raster	A file type created in Microsoft Paintbrush. Many programs can read this format.
TARGA	Raster	A file type usually created by video-industry software. Many programs can read this format.
PICT	Vector	A file type created in MacPaint. This file type is platform dependent.
WMF	Vector	A file type created in Microsoft Draw. Others software can read this format.
CGM	Vector	A file type created in various software programs. Other software can read this format.
DXF	Vector	A file type created in AutoCad. This file type has some compatibility with other software.
AI, EPS	Vector	A file type created in Adobe Illustrator, CorelDRAW, and others. Many programs can read this format.
CDR	Vector	CorelDRAW creates this file type. Some software programs can read it.

Once you have identified the file format of your image, you will need to either use the software it was created in to open it or work with a graphics-conversion program such as Paint Shop Pro, CorelDRAW, Hijaak, or Lview Pro. I have provided additional information about conversion software later in this chapter.

Creating Professional Images

Now that you have a good basic understanding of Web-image file formats and when to use which one, you need to develop Web image-production skills. Next, I will look at the different ways to prepare images and some of the choices that you will need to make, depending on your Web site and goals.

GIFs

In Chapter 16, you learned about the problems of 256-color monitors and cross-platform inconsistency in displaying images. Netscape came to the rescue of surfers everywhere with the Netscape palette. The palette, containing 216 "browser-safe" colors, is the solution for Web designers around the world. By using the palette, you can provide cross-platform consistency for displaying Web images.

WWW Of course, by now you have a copy of the palette ready to go with your paint program, right? If not, download a copy (Victor Engel has provided a very nice one at `http://the-light.com/netcol.html`, as has Lynda Weinman at `http://www.lynda.com/hex.html`) and then dive into learning when to use the Netscape palette and when not to.

Flat Color Images

You almost always want to use the Netscape palette with line art or flat color images like logos. The ideal situation is to have created the image using the Netscape palette colors in your paint program. If the image was not created using the Netscape palette, the next best scenario is to load the image into a paint program such as Photoshop and save the image as a GIF file using the Netscape palette. Figure 17.4 illustrates the fictitious Tropical Escapes logo in its original RGB file. You can see several areas of flat color in the palm tree and in the squiggly blue line. The rest of the name is a smooth gradient from a hot red to a bright yellow. The color change in the text is gradual and shows no *banding* or stripes of color. The original file size for this image in TIF format is 46,900 bytes.

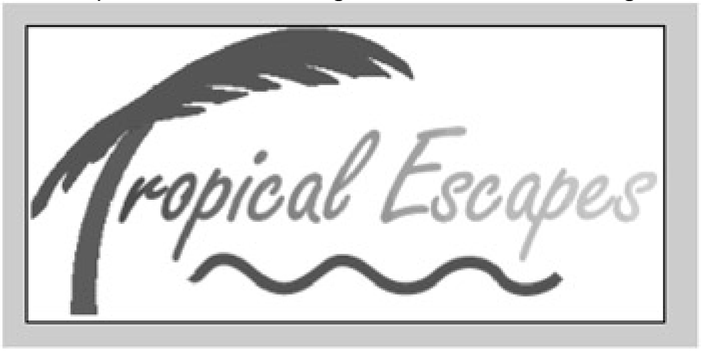


Figure 17.4: The original version of the Tropical logo

Figure 17.5 shows the same logo after it has been converted in Photoshop to a GIF file using the Netscape palette. The file was saved using the No Diffusion option. If you look at the image carefully, you can see that the flat areas haven't changed a great deal; the original colors have only been converted to Netscape palette colors. The area that shows the most difference is the gradient in the text; if you were to view the image in a Web browser, you may notice some slight banding. The GIF file saved with no diffusion is 7,880 bytes.

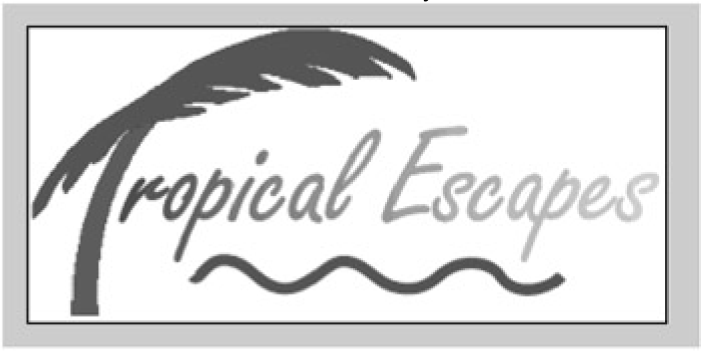


Figure 17.5: The Tropical Escapes logo using the Netscape palette with no diffusion

Figure 17.6 illustrates the logo once again, but this time the original logo was converted to the Netscape palette with diffusion turned on. Careful examination of the image shows that it doesn't have a great deal of change overall. Though it's difficult to notice, the banding problem is gone. Under careful examination, you may be able to see some diffusion of colors—a few pixels of each color mixed together to make the change in color less obvious—in the areas of color change in the text. The diffusion generally makes for a better-looking image. The GIF file saved with diffusion turned on is 6,974 bytes.

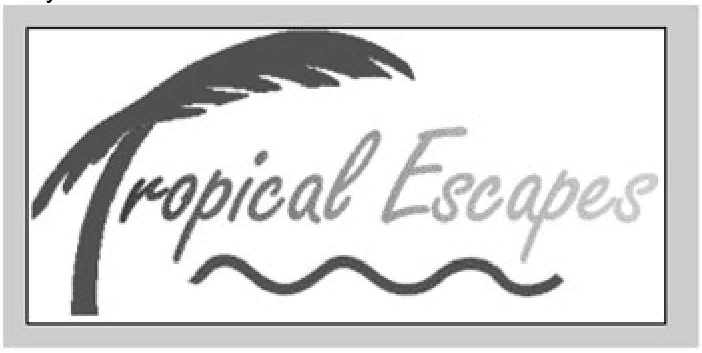


Figure 17.6: The Tropical Escapes logo using the Netscape palette with diffusion

Tip If you have an image that contains a few solid areas of color, it's worth experimenting with converting those colors to the Netscape palette by hand. Doing so allows you to choose the color combinations that look best instead of relying on your paint program's conversion algorithm. Just use the eyedropper tool to select the Netscape colors from your swatch palette or your Netscape color file and "pour" those colors into the solid areas of your image. You may discover that you can make a more pleasing image than expected!

Photographic-Type Images

I've said that JPEG is the format of choice for continuous tone and photographs. However, if you are a control freak (and many Web designers are) and you expect that most of your audience will be viewing your Web images on a 256-color monitor, you may present your photos as GIFs. You will thus insure that your audience will see the image just as you intended, rather than as the Web browser chooses to render the image.

This is where it gets a bit trickier to decide whether to use the Netscape palette in your file conversion to GIF. In some cases, the choice will be very obvious; the image degradation that can occur by using the GIF format for a photograph containing a wide variety of colors can be truly appalling. (Try saving in GIF a photograph of an image with many levels of light and shadow and many different colors. Ick!) No hard and fast rule on whether to use the palette exists; what counts is your personal preference in the images that result from experimenting.
Figure 17.7 shows the original RGB photograph of a happy soccer player and her father (Linda and Dave Navarro). As you can see, this image doesn't contain a very wide spectrum of color but does have many changes in value and intensity of color in the light and shadows of the image. Notice especially the changes in tone of the clothing and skin tones. The file size of the original tiff file is 325,062 bytes.

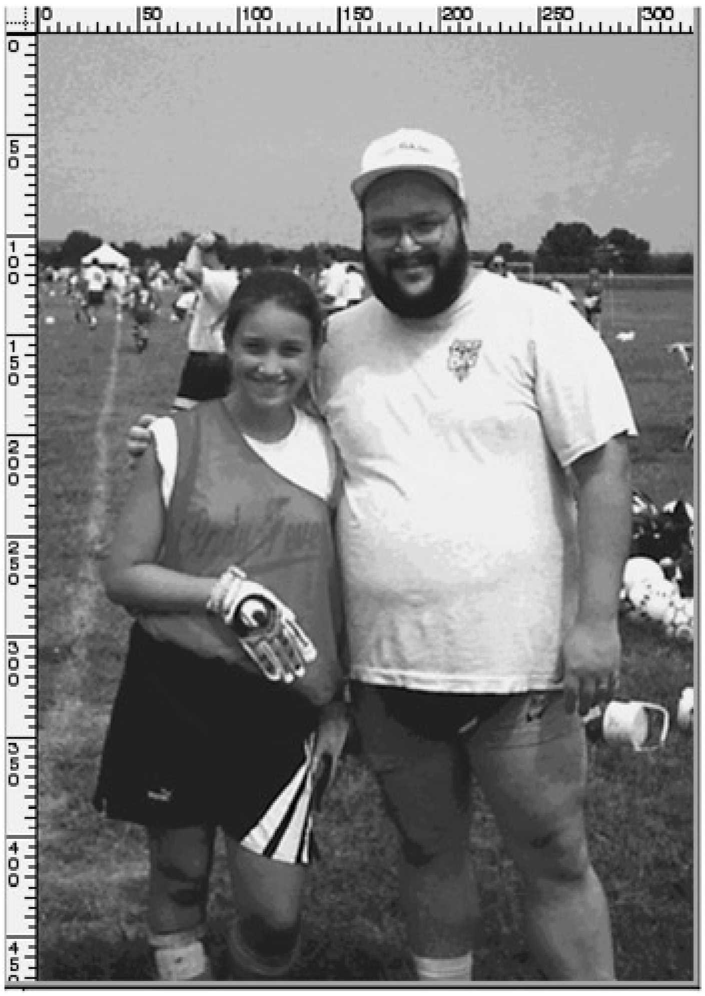


Figure 17.7: The original version of the soccer photo

Figure 17.8 is a prime example of what happens when diffusion is turned off when converting a photographic image to GIF using the 216-color Netscape palette. Well, this is alarming! You can see the loss of detail in the image. It has been *posterized* very badly, meaning that the color conversion has flattened out all of the colors, making them appear like a flat poster treatment. The photo has lost all depth, and the colors of skin were converted especially poorly. Skin tones have turned green, red, or gray in the highlights and shadows. The file size of the GIF image is 35,872 bytes.

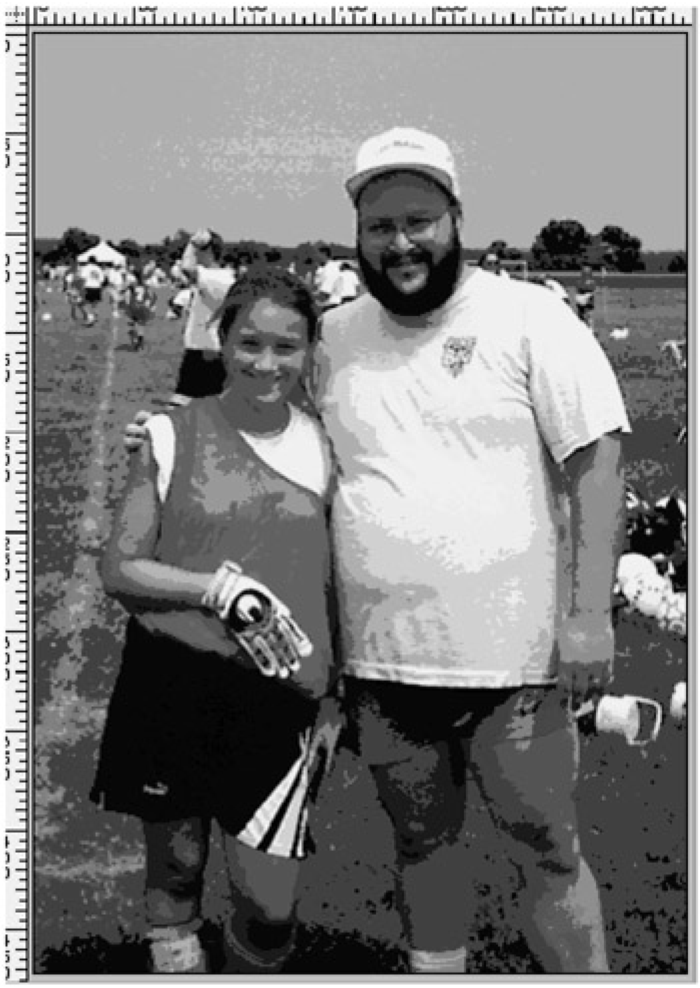


Figure 17.8: The soccer photo using the Netscape palette with no diffusion

Figure 17.9 shows the same photo with the 216-color Netscape palette and diffusion turned on. Although it is quite an improvement over the previous version, the image still isn't very good. The colors are still not right and the diffusion causes a great deal of speckling in the image that is distracting to look at. Do you find yourself staring at the spots in Linda's soccer shorts or perhaps at the way Dave's leg blends into the green of the grass behind him? The file size of the GIF image is 65,721 bytes.

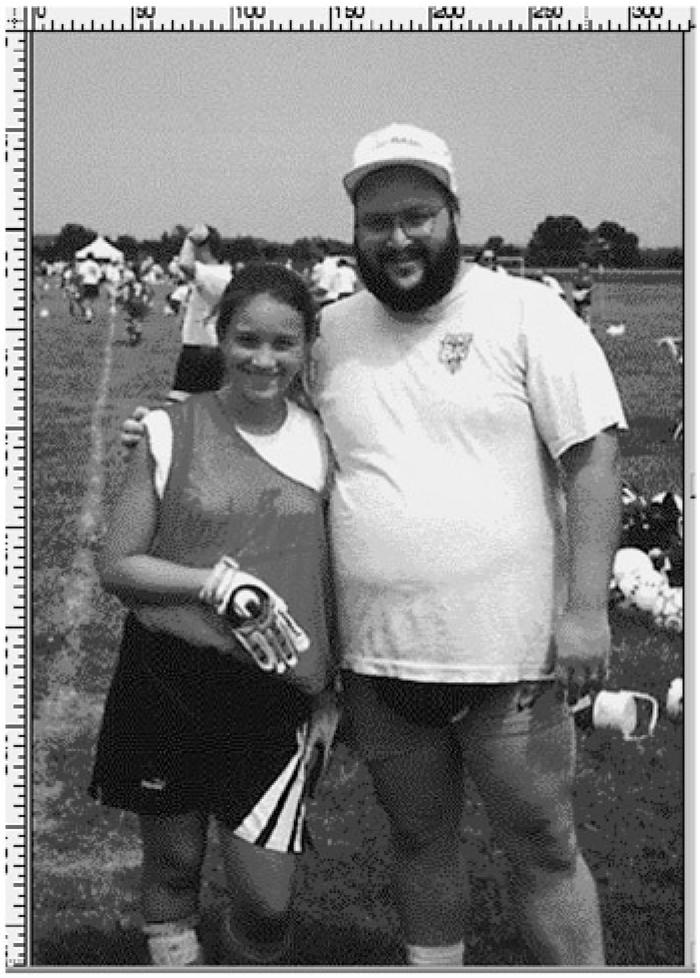


Figure 17.9: The soccer photo using the Netscape palette with diffusion

In Figure 17.10, you can see the effect of using an *adaptive palette* with diffusion. The adaptive palette is the paint software's algorithm for selecting the most common colors in the original image to create a 256-color palette for the GIF conversion. Depending on your software program, the adaptive algorithm is different and more or less effective. This image was converted in Adobe Photoshop.

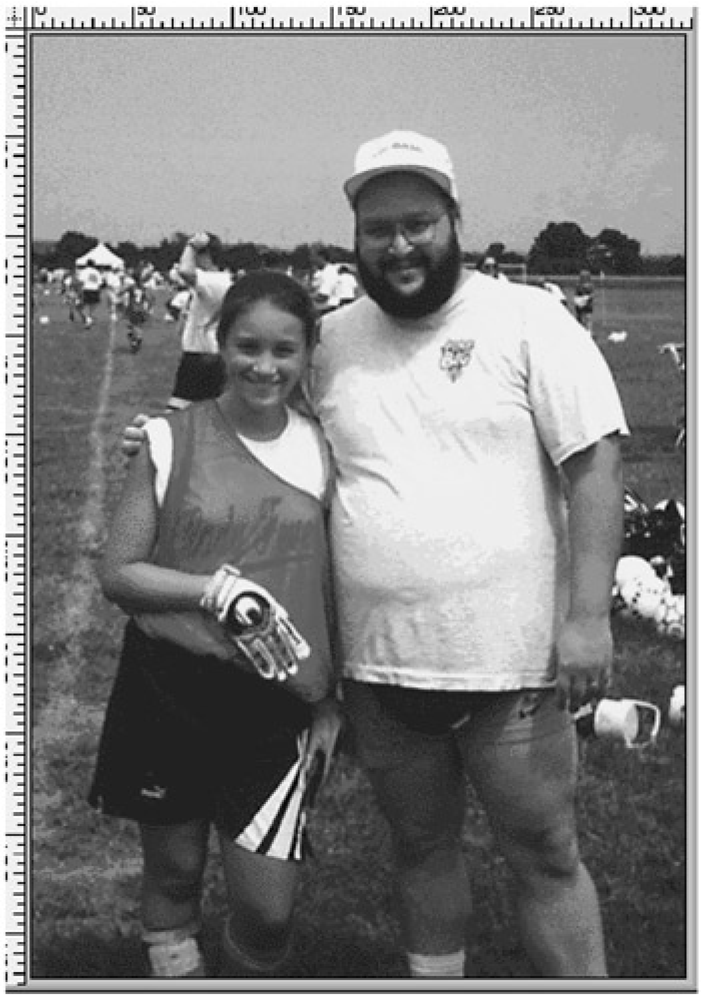


Figure 17.10: The soccer photo using a 256-color adaptive with diffusion

The image in Figure 17.10 shows less degradation than the previous examples. The skin tones aren't ideal, but they are closer to real skin colors. The image suffers from some posterizing, but not so much as in the previous examples. There are also fewer speckles. Although this image is acceptable for Web publishing, it *will* dither on a 256-color monitor, as it does not use the Netscape palette. The file size of the GIF image is 85,511 bytes.

Figure 17.11 shows the last of the GIF examples of photographs. In this image, the adaptive palette was once again used, this time with a limited palette of 64 colors. Again, there is a significant loss of detail in the image; but the colors are better than in the Netscape-palette examples. This image is marginally acceptable, but it has the same problems as the previous adaptive-palette image: It will dither in Web browsers if the user is viewing it on a 256-color monitor. The file size of the GIF image is 60,022 bytes.

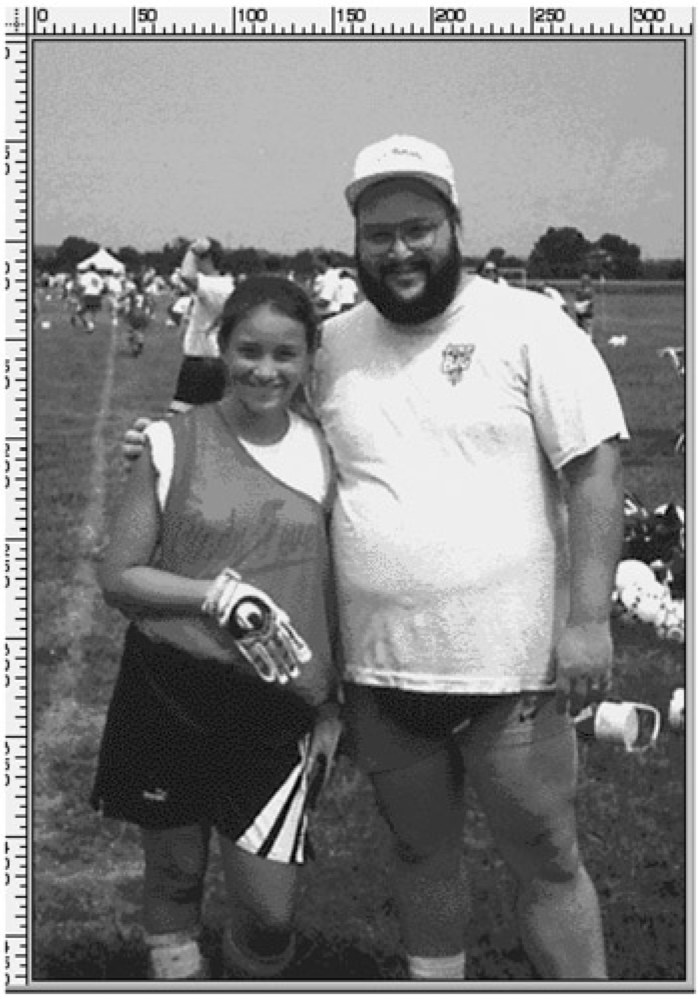


Figure 17.11: The soccer photo using a 64-color adaptive palette with diffusion

Sizing Files

Did you note the file-size information for each of the previous images? In the case of the Tropical Escapes logo, the best looking GIF image actually turned out to be the smallest image: 6,974 bytes, as shown in Table 17.2.

Table 17.2: Logo File Sizes

FILE	SIZE
Original tiff image	46,900 bytes
GIF Netscape palette without diffusion	7,880 bytes
GIF Netscape palette with diffusion	6,974 bytes

The Soccer photograph turned out somewhat differently. The smallest image (35,872 bytes) was the worst-looking image; the Netscape palette without diffusion gave people purple legs. The second smallest image (60,022 bytes) was the 64-color adaptive palette. This image was acceptable for publishing; however, not ideal. Table 17.3 shows the file sizes for the versions of the soccer photo.

Table 17.3: Soccer-Photo File Sizes

FILE	SIZE
Original tiff image	325,062 bytes
GIF Netscape palette without diffusion	35,872 bytes

Table 17.3: Soccer-Photo File Sizes

FILE	SIZE
GIF Netscape palette with diffusion	65,721 bytes
GIF 256 Adaptive palette with diffusion	85,511 bytes
GIF 64 Adaptive palette with diffusion	60,022 bytes

I'll now look at JPEG versions of the same soccer photograph for comparison of file sizes and image quality.

JPEGs

JPEG is the file format of choice for continuous-tone images on the Web. It allows you to provide a file with full 24-bit (RGB) color at greatly reduced file size. However, 24-bit images will only display as 8-bit images on a 256-color monitor. So your JPEG image will be force-dithered into the computer's operating-system palette if displayed on a low-color monitor.

That said, I'll look at the choices you will need to make for creating professional JPEG images. JPEG conversion is much less complicated than GIF conversion; no palette issues need to be considered. The only decision that needs to be made is one of image quality versus the final size of the file.

Most paint programs offer three or four levels of compression for JPEG images, ranging from maximum-quality image (lowest amount of file compression) to low-quality image (highest amount of compression). How much image quality are you willing to trade for file size? Your answer is dependent on the specific image, so the only real way to decide is to once again experiment with the images. Figure 17.12 shows a maximum-quality JPEG version of the familiar soccer photo. (You can compare the image to the original shown in Figure 17.7.) Very little difference in the overall image's appearance exists between the two. The place to look for image degradation (posterizing) is in the shadows. In Figure 17.12, the shadow cast by the man's arm hanging at his side is a good place to look for changes in the levels of shadow colors. Not much difference exists between this image and the original. This JPEG image is 163,859 bytes.

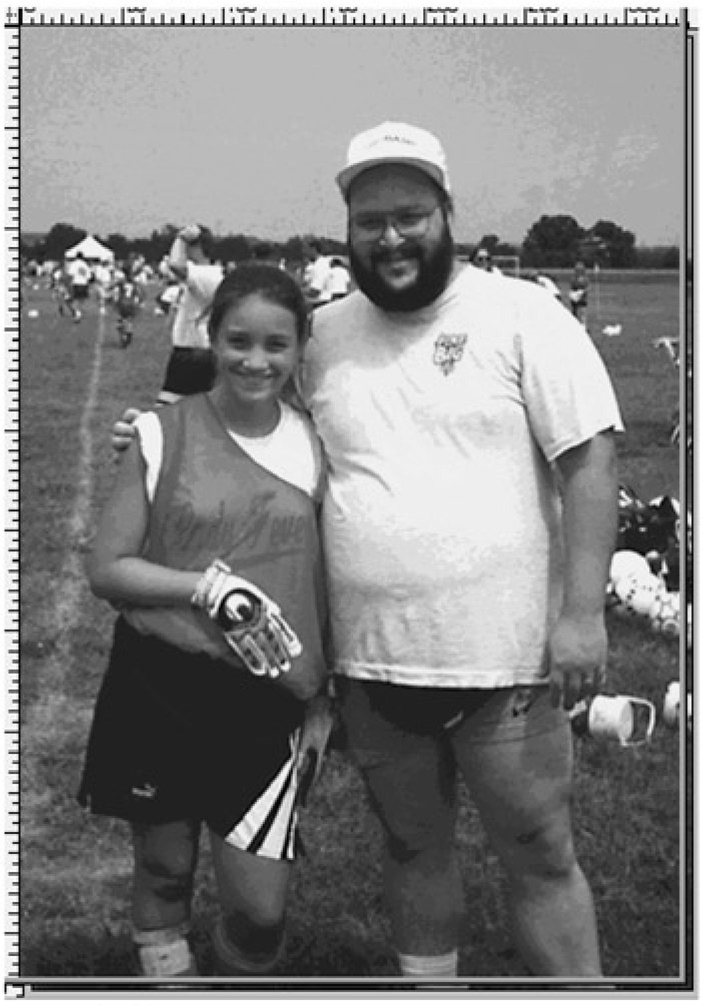


Figure 17.12: The soccer photo saved as a maximum-quality JPEG

In Figure 17.13, you can see the medium-quality JPEG image. The image begins to show the effects of compression: A bit of deterioration is in the shadows and JPEG compression has introduced a few "artifacts" or speckles. If you look carefully at Linda's forehead, you can see some areas of light speckles that don't appear in the original image. This JPEG image is 28,210 bytes.

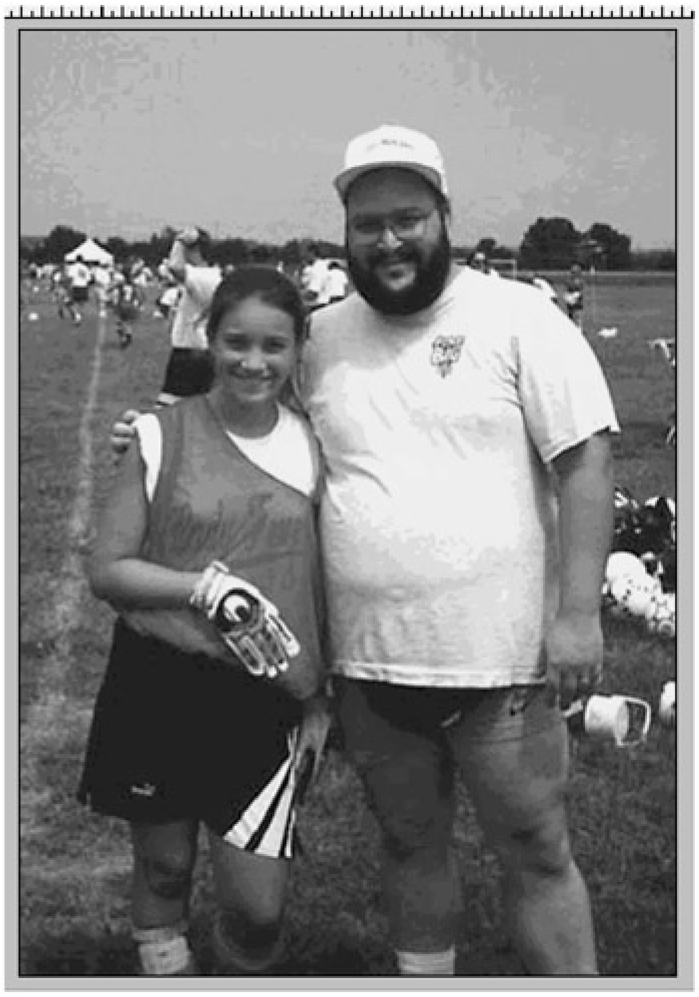


Figure 17.13: The soccer photo saves as a medium-quality JPEG

Figure 17.14 shows the lowest quality of jpeg. This image shows more posterization in the shadows than either of the previous images. You can also see some additional JPEG-induced artifacts. However, the overall image quality is fine for Web publishing. This JPEG image is 21,265 bytes.

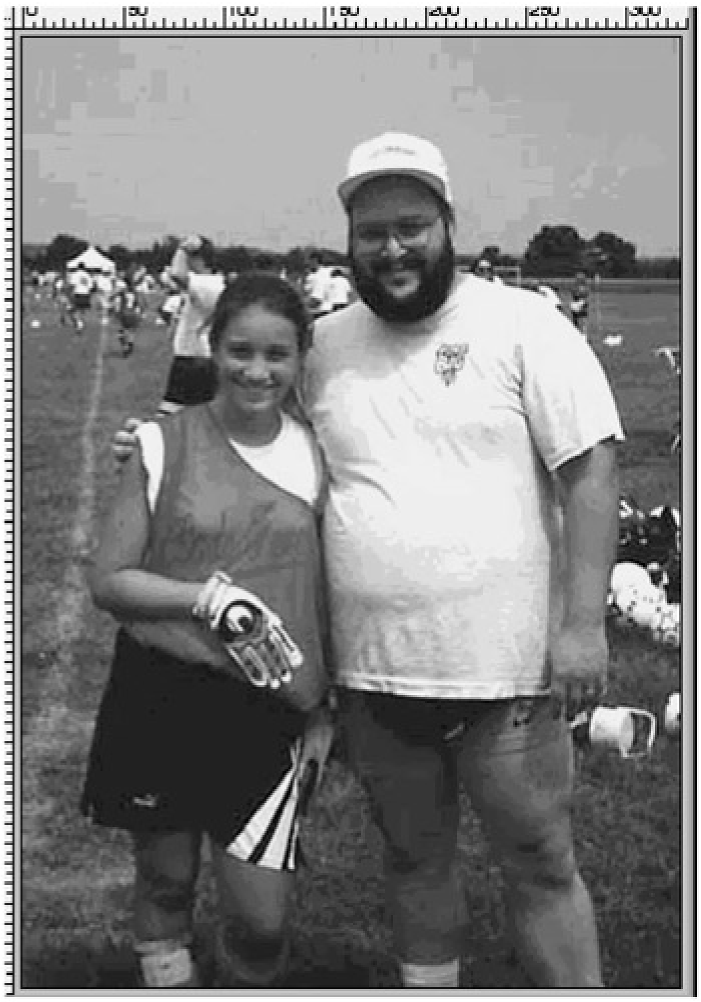


Figure 17.14: The soccer photo saved as a low-quality JPEG

JPEG file sizes are almost always much smaller than comparable GIF files sizes. Based on visual inspection and the file sizes, the low-quality JPEG would be the best bang for the buck on the Web. The maximum-quality image is a beautiful photo and about half of the original image; however, few visitors will wait for 163K photo to download. While a 21K image requires some download time, the quality of the image is worth the wait. Table 17.4 shows the file sizes of the three JPEG images.

Table 17.4: File Sizes for JPEG Images at Varying Qualities

FILE	SIZE
Original TIF image	325,062 bytes
JPEG maximum quality	163,859 bytes
JPEG medium quality	28,210 bytes
JPEG low quality	21,265 bytes

Tip Providing small thumbnail images linked to larger photographs is an excellent way to let your visitor preview an image without being forced to download the entire file. Thumbnail images work especially well when a Web site has a large number of images; such sites could be a travel site or photographer's portfolio page. Consider using this presentation, especially if your audience is primarily connected to the Net via modems rather than T1 lines. Your visitors will thank you for it!

Resources

I've covered some important points about working with the various types and formats of digital images, but I've really only scratched the surface. For more information about digital file formats and Web imaging, check out URLs listed in Table 17.5.

Table 17.5: WWW Web Imaging and Digital File Format Resources

WEB PAGE	DESCRIPTION

Table 17.5: WWW Web Imaging and Digital File Format Resources

WEB PAGE	DESCRIPTION
Center for Innovative Computer Applications http://www.cica.indiana.edu/graphics/image.formats.html	The article here provides a comprehensive list of image formats, including in what programs many proprietary formats originated.
Graphics File Formats FAQ Home Page http://www.ora.com/centers/gff/gff-faq/index.htm	The Web page is about the best resource, despite its age, for technical (and not so technical) details of file formats.
Usenet's version of File Formats FAQ http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/top.html	The Usenet version is another good collection of information.
WC3's PNG (PortableNetwork Graphics) http://www.w3.org/Graphics/PNG/	W3C's site provides a good description of PNG and information about the future.

Tools of the Trade

Now that you have a basic understanding of digital image types and formats, you're probably wondering how to actually work with those images. What imaging program will dependably produce professional graphics for your Web site?

A wide variety of tools are available for image creation and manipulation. Which you choose to use will depend on several factors:

- Your graphic-design needs
- Your software budget
- Your patience for and commitment to learning to use graphics software

A brief overview of the most popular graphic-design software products follows.

Adobe Photoshop

Adobe Photoshop is indisputably the prime product for most professional graphic designers working with Web images. Photoshop has a breadth of tools and image-manipulation facilities that other software manufacturers try to emulate. Photoshop is also one of the most expensive imaging software products and has one of the steepest learning curves for the graphic-design novice.
Whether you want to scan a photo and retouch it, create an image of a "painting" (digital art that appears to be watercolor, impressionist, etc.), or create the standard buttons, bars, and backgrounds, Photoshop can do it. Photoshop's interface is fairly intuitive once you become familiar with where the various tools are located. Figure 17.15 shows Photoshop's standard palettes.

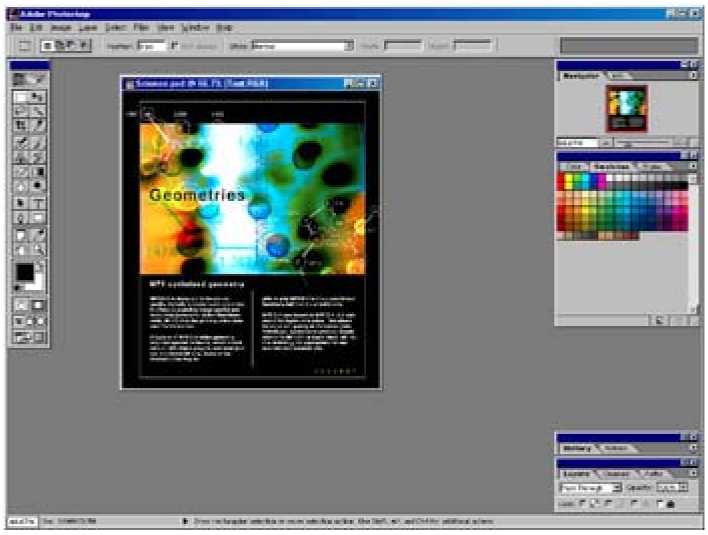


Figure 17.15: Adobe Photoshop 6

The term *palette* here refers to the individual tool windows offered by Photoshop. The palette system is user-configurable and very convenient. The Help menus and tutorials are well designed and fairly comprehensive technically. However, most Photoshop novices would be well advised to purchase one of the third-party reference books, such as *Mastering Photoshop for the Web* by Matt Straznitzkas, as neither the tutorial nor documentation is geared toward those who are not graphic designers. In other words, if you think a *screen* is what keeps out the flies in summer then visit your bookstore.

Photoshop creates raster image files but is quite capable of working with a number of vector file formats, particularly AI (Adobe Illustrator) and EPS (Encapsulated Postscript). It also is capable of working with and converting most raster-image file formats to GIF and JPEG files. Photoshop will allow you to work with almost all of the images that you may come across.

Photoshop will also allow you to manipulate palettes for GIFs, including palettes for animation files. It will not create animations directly; however, it will create the individual animation frames that you may then assemble in an animation tool.

WWW For more information about Photoshop, visit Adobe's Web site at `http://www.adobe.com/prodindex/photoshop/main.html`.

CorelDRAW

CorelDRAW is one of the best tools available for working with vector images. CorelDRAW can handle almost any vector image that you can find, and, most importantly, it will allow you to edit the image and convert it to a Web image. CorelDRAW isn't cheap, but it includes raster-image-handling software, a 3D-image-creation package, and a variety of other graphics tools. If you are working with a variety of file formats, CorelDRAW is one of the best software products you can buy.

Figure 17.16 shows CorelDRAW's main window and *rolldowns*—Corel's version of the Adobe's palettes discussed previously.

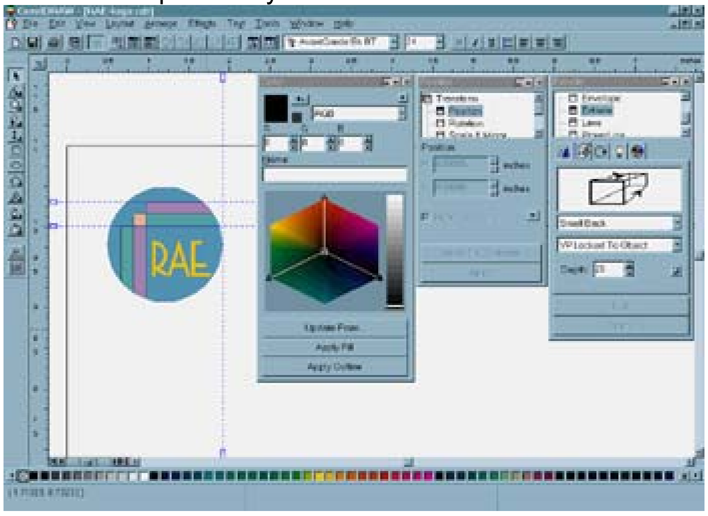


Figure 17.16: CorelDRAW's main window

CorelDRAW is configurable and offers an excellent help menu and tutorials. The documentation is written for the novice user and can actually teach the novice how to use the software effectively. The user interface is easy to understand, and the Help menus are also good for the novice user.

Perhaps the best feature of CorelDRAW is the clipart and photo-CD collection included in the package. The clipart and symbols include well over 30,000 graphics, and the photos number more than 1,000. This library is an excellent way to get started providing your own graphics.

Included in the CorelDRAW package is PhotoPaint, Corel's raster-image-manipulation software. PhotoPaint performs many of the same functions as Adobe Photoshop but is not nearly as robust.
WWW For more information about Corel products, visit its Web site at http://www.corel.com/.

Macromedia Fireworks
Following the popularity of its Dreamweaver HTML editor and Flash animation products, Macromedia scored another hit with Fireworks, its new graphics-editing package. Figure 17.17 shows Fireworks' primary work area, with the default palettes and tools selected.

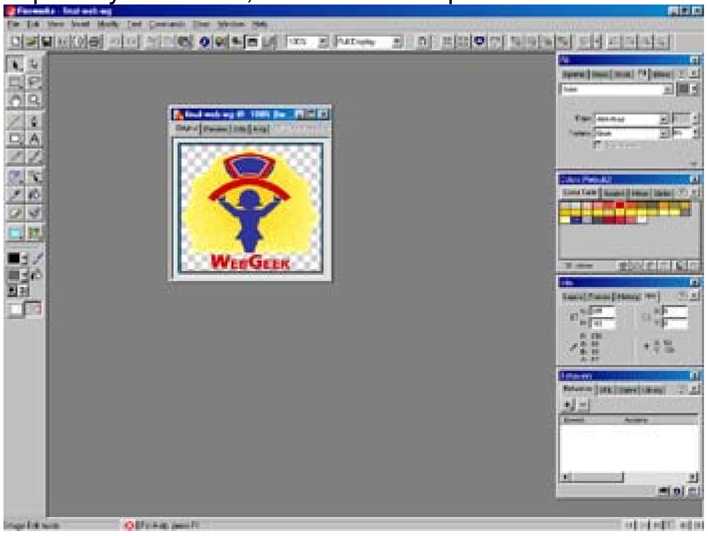


Figure 17.17: Fireworks' main window

A major selling point for Fireworks is its easy integration with other Macromedia products. Dreamweaver users can invoke Fireworks directly from within the Dreamweaver interface. Fireworks also provides assistance with rollover scripting, image slicing, and other activities that might traditionally be thought of as HTML-editor tasks.
WWW Macromedia offers a 30-day trial of the product, available for download at http://www.macromedia.com/software/fireworks/trial/. At the time of this writing, Fireworks version 4 was available in trial format.

Pricing for Fireworks as a stand-alone product is $199 for an electronic copy (activated from within the trial download), or $399 as a bundle with Dreamweaver.

Jasc Paint Shop Pro
WWW What started as a popular shareware tool, Paint Shop Pro (PSP) from Jasc is now competing in the "big leagues" as a fully viable graphics-editing suite for the serious Web designer. Staying true to its shareware roots, the price of Paint Shop Pro can't be beat—$99 for the downloadable version 7 ($109 boxed). A fully functional trial is available from the Jasc Web site, located at http://www.jasc.com//product.asp?pf%5Fid=001. Figure 17.18 shows the PSP user interface in action, editing a GIF image.

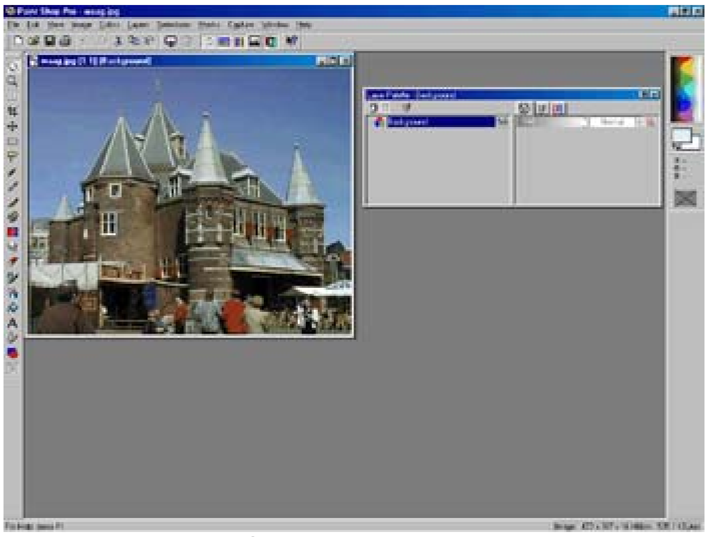


Figure 17.18: Paint Shop Pro

My favorite handy feature in Paint Shop Pro is the ability to import screen captures directly into the editor. Many freelance designers use such a technique to make thumbnail images for a portfolio of Web sites they've created.

How Those Who Can't Draw Survive

Want to know a closely kept secret? Not every Web designer is an artist. Not everyone can draw; in fact, many Web-site designers can't draw a straight *or* curved line. How can you produce a professional Web site without being an artist? Read on, as several solutions are easily available for the nonartist Web designer.

Stock Photos

If you're not a whiz at photography, don't worry. Many photographic resources are just waiting for you to come to them! Using stock photos is an excellent way to provide images for your Web site. Whether you need an illustration of particular topics or just some decoration, stock photography can provide color and look at a reasonable price.

Stock photographs are available on nearly any subject, in a wide variety of media. You can purchase stock photos in a number of formats, such as:

- CD-ROM image collections
- Individual images downloaded from the Web
- Slides
- Individual high-resolution scans on digital media of your choice

For Web use, you typically won't need high-resolution images, as your Web output is likely to be at 72ppi. Getting individual stock photos from the Web or as a part of a topical-collection CD is likely to be the Web designer's most cost-effective way to purchase images.

Warning Be sure to check what rights you are purchasing when you buy stock photos. Stock photos are sold with varying usage rights; sometimes they have only one-time-use rights, which are inappropriate for the Web. Also, be sure to find out whether you must use the image as is or if you may edit it and add the altered image to your Web site.

The following will give you a start in your search for stock photography:

- **WWW** **PhotoDisc** PhotoDisc, at http://www.photodisc.com/, offers a wide variety of images via CD-ROM and on the Web. Some images include *clipping paths*—that is, basically, sets of instructions on how to handle cutting and pasting of an image, most useful for putting part of the image "behind" another part while the rest of it is "in front."
- **Comstock Stock Photography** Comstock, http://www.comstock.com/, offers both one-time use photos and a royalty-free collection through their Comstock Klips collections.

- **Digital Vision** Digital Vision, at http://www.imageclub.com/digitalvision/, sells CD-ROM collections of photographs by topic. These images include clipping paths.
- **Artville** Artville, http://www.artville.com/, offers a collection of royalty-free photographs on CD-ROM.

Clipart

Clipart is another must-have resource for Web designers, artists, and nonartists alike. Although they won't always confess to it, most graphic designers keep a collection of clipart catalogs and clipart CD-ROMs for use in their own designs or in idea files. Whereas clipart can have a "same old, same old" sort of look to it, plenty of creative methods can make clipart fresh in your designs.

Clipart usually comes on CD-ROM in either raster or vector format. (Vector format is the most common.) Vector clipart lends itself to customization by the user. The clipart can be modified by changing its color, removing elements from the image, adding other clipart elements, or by processing the image (in a raster-image editor such as Photoshop) with various filters.
As you can see in Figure 17.19, a little creativity and some image editing goes a long way toward customizing clipart. The original vector images, the safari hat and the construction guy, were combined in a drawing program to create the safari guy. His original hardhat was removed and replaced with a safari hat. The shirt was recolored in shades of green to give him a jungle-like feel. Of course, he's really not quite ready for a safari Web page yet—it would be a good idea to replace his sledgehammer with a camera or other suitable tool.

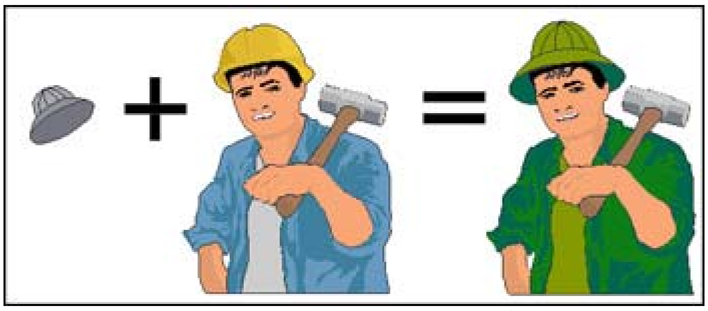


Figure 17.19: From construction guy to safari guy

Once the changes to the original vector art were made, the image was exported as a TIF file and opened in Adobe Photoshop for some more image editing and eventual conversion to a GIF image. Figure 17.20 shows safari guy changed to a more textured image that will display nicely on the Web. After I used some filters in Adobe Photoshop to add noise and alter his colors a bit, safari guy doesn't look much like a standard piece of clipart. Once you have experimented with clipart and done some customization, you'll never see clipart in quite the same way.

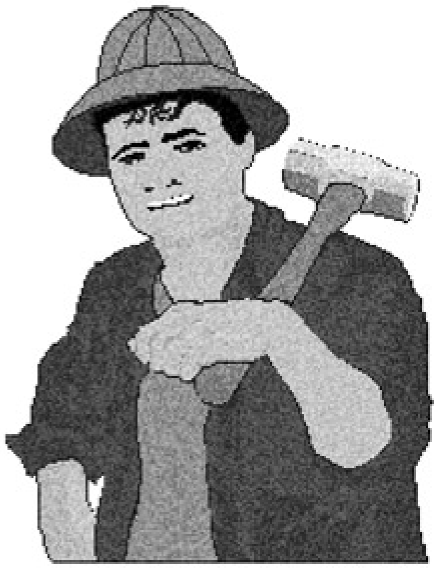


Figure 17.20: From smooth vector image to textured Web image

As with stock photos, clipart comes with differing usage rights. Always verify that you can edit and use the art on a Web site before you add it to your Web page.

The following list will give you a start in your search for clipart:

- WWW **Artville** Artville, at http://www.artville.com/, offers an excellent collection of unusual clipart on CD-ROM.
- **Artworks** Artworks, http://www.dgusa.com/, has a wide variety of clipart images in varying styles on CD-ROM.
- **Image Club** Image Club, a division of Adobe located at http://www.imageclub.com/, has one of the more comprehensive collections of clipart, photos, tools, and a terrific monthly catalog.

Subcontracting

Many Web designers subcontract their graphics work to an outside graphic designer. Whether you need assistance with the creation of a couple of images or the overall look and feel of your Web site, an outside designer will be willing to provide their expertise.

Finding a good match with a graphic designer is critical to the success of your Web-site project. Forget all the stories you've heard about temperamental artists—they aren't that difficult to get along with. Finding the right designer is quite possible with a bit of research and some planning on your part. Roll up your sleeves and get out that notepad again. It's time for some analysis of the task.

Define the Task

Perhaps the most important part of working with a graphic designer is being able to clearly articulate what you want them to do. You will need to honestly assess your Web-site plan and determine which pieces of the visual presentation you need help with. Then think about your budget and develop a list of the discrete tasks for your graphic designer to accomplish.

Contract Information

When you work for a company or subcontract work to another individual or firm, always have a contract. Many obvious reasons exist for having a formal written contract that outlines exactly who will do what for what price and when it will be accomplished. It is extremely important to have an attorney look over any contract that you personally write or are considering.

The following URLs may be of some help for more information about contracting, running a Web-site-design business, and other business issues:

- The U.S. Business Advisor provides a comprehensive resource at http://www.business.gov/.
- A Business-resource company and newsletter for people in creative fields may be accessed at http://www.creativebusiness.com/.
- Business resources are never complete without visiting the IRS. Find it at http://www.irs.ustreas.gov/.
- This site is a gem with many resources for those in need of Web contract-writing help: http://provider.com/contracts.htm.

Tip Depending on your personal inclination and the complexity of your needs, you may wish to develop a formal RFP (Request for Proposal). An RFP solicits a proposal or estimate from individuals or design firms that you will consider hiring as a subcontractor.

Choosing a Graphic Designer

Choosing a graphic designer isn't that complicated, but some important issues need to be addressed. Finding a compatible style and developing a good working relationship is crucial to the success of subcontracting.

You can start your search on the Web. In your travels, you've seen graphics at Web sites that impressed you; go back to those sites and find out who created the images. Another good source for finding graphic designers is to talk to your online friends. Ask people whom they would recommend for graphic design.

Tip If the artist isn't listed directly on the Web page, try sending e-mail to the site's Webmaster. Most people are delighted to receive a compliment. Tell them you like the site, and they will happily tell you who designed their images.

Once you have compiled a list of designers, contact them and ask if you can view their portfolios. Most designers who create Web graphics will have an online portfolio for you to view or will be happy to send you a list of URLs that contain their work. As you review portfolios, be sure to take notes on what images you liked, what you didn't like, and whether the images are technically well done.

Tip When reviewing Web images on a portfolio page, see if the images are created with technical know-how. Do they load quickly? Present well at 256 colors? Succeed at accomplishing their purpose on the Web site?

Once you have narrowed your list of designers, talk with them via e-mail or telephone. You will need to get a sense of working with the person. Do you have compatible styles and can you communicate? Do your budget expectations match? Be up-front about any issues that you anticipate will be a problem. Always include a copy of your task list or RFP so you're talking about the same issues!

Summary

Producing professional Web images isn't hard for a novice Web designer with the right tools and resources. With a basic understanding of digital imaging for the Web and knowing what makes a quality Web graphic, you're well on your way.

Chapter 18: Multimedia on the Web

Overview

The Web is evolving constantly into a sophisticated multimedia experience. Many Web users got their first taste with animated graphics. Sound could be embedded in the page using proprietary HTML markup in each major browser, as could short movie clips by using *plug-ins*, the helper applications that work within the browser. Then came JavaScript, and graphical menus could change when the mouse passed over them. Java applets allowed more advanced animation, along with interactive applications such as games, calculators, or other tasks that require computation after user input.

Next, a company called Macromedia provided a giant leap forward in the multimedia experience. Its Shockwave and Flash player plug-ins allowed users to see complex animations, movies, and interactive entertainment. Indeed, learning to create Flash animations and Shockwave games is one of the most desirable skills in many Web-design circles.

In this chapter, I'll tour the multimedia tools available to Web developers today and some of the exciting new markup languages coming out of the W3C, which deal with graphics and animation in a nonproprietary format.

This chapter covers the following topics:

- A quick tour of Macromedia development tools
- Producing mouseover effects with JavaScript
- Mouseovers using style properties: a simple DHTML example
- Audio and video on the Web
- SMIL: an XML-based multimedia solution

Macromedia Shockwave

At the time this book went to print, Macromedia was celebrating the five-year anniversary of the Shockwave player. That any given Internet technology has survived that long speaks volumes for the utility and creativity that it can bring to Web sites.
Shockwave animations are created using the Macromedia Director Studio authoring tool. Currently in version 8 of its release, Director Studio not only creates Shockwave animations and applications for the Web but also nearly any multimedia experience on a computer platform, including interactive CDs, the introductions and flashy menus you sometimes see when installing new software products from CD, and kiosk applications. Figure 18.1 shows the Director Studio interface.

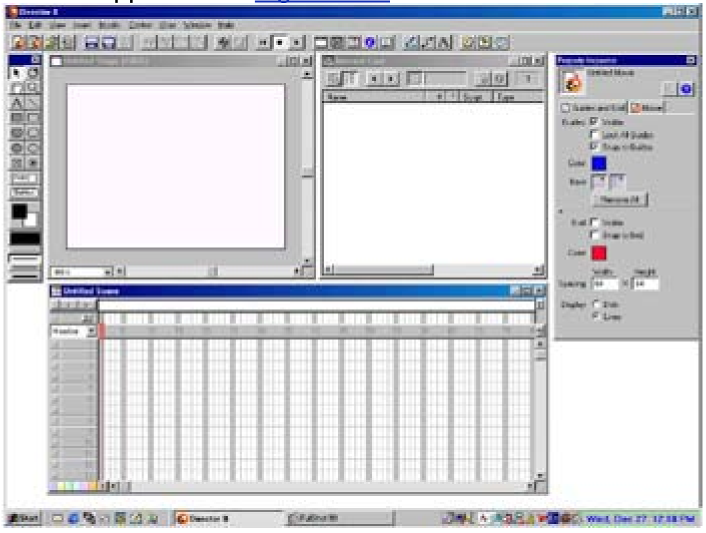


Figure 18.1: Macromedia Director Studio 8

WWW Although much of the Director Studio interface makes use of drag-and-drop and a myriad of palettes that you would find in other Macromedia applications, a large segment of the work is done in a scripting language known as Lingo. The details of working in Lingo are beyond the scope of this book. Resources for learning more are available online at the Macromedia University (`http://macromedia.elementk.com/`).

Tip For detailed study of Director and Lingo, consider *Director in a Nutshell* and *Lingo in a Nutshell* by Bruce A. Epstein, published by O'Reilly & Associates.

You can explore the possibilities of developing with Director by visiting `http://www.shockwave.com`, where you can solve puzzles, play games, and watch movies, all produced using the Director Studio. A current favorite is the South Park Snowballs game (see Figure 18.2).

Figure 18.2: A Tetris-like game featuring South Park characters, at Shockwave.com

Macromedia Flash

Macromedia Flash, currently in version 5, has a very similar look and feel when compared to Director Studio. In some ways, Flash can be described as "Director Lite" in that it concentrates on animations and reactions to user input (objects move or change based on mouse clicks), rather than the more complex interactive components possible using Director and Lingo scripting for Shockwave.
A simple example of Flash animation ships with the Flash 5 product, allowing the user to roll a set of animated dice (see Figure 18.3).

Figure 18.3: A Flash animation with a simple user interaction

Flash animations have been particularly popular recently for creating a splash page—a cool, wow-inducing introduction to a site. Though perhaps visually stunning (depending on the talent of the artist!), they can have the effect opposite of grabbing the visitor's attention and making them stay. If a splash page contains animation, which by its nature results in a longer download, and provides no other information, many users will move on to sites that give them what they're looking for on the very first page.

Warning After designers learn Flash animations, they typically may feel the urge to show off their new talent with large or lengthy animations. But resist the urge to add content to a site simply because you can. Remember the purpose of your site; if it's anything other than to entertain or for personal expression, give the visitor the goods up front.

Animation through JavaScript

JavaScript is a programming language included in the Netscape Navigator (and several other) browser executable program. It's an *interpreted* language, meaning the browser processes the program instructions, or *scripts*, at the moment they are run—a process known as a *run-time* event. JavaScript is also a basic *object-based* programming language. An object-based system of programming development views each portion of the Web page and browser as an object—the browser window itself is an object, as is an image within the XHTML file being displayed. Object based simply means that the programming instructions identify individual objects and then direct instructions to them specifically.

Tip Despite its name, JavaScript is a wholly different language from *Java*, the programming language developed by Sun Microsystems. Unfortunately, the similarity in names has led to some confusion within the Web community. At this point, you only need to be aware that they are two different languages that produce different functions for Web designers.

Mastering the JavaScript language is a topic worthy of a book all its own. However, in this section you'll learn the basics necessary to include simple animations based on responses to *events*—actions that are perceived by the browser, such as placing a mouse pointer over a hyperlink. The browser recognizes that event and changes the pointer into a hand. You'll learn to handle these *mouseovers* yourself.

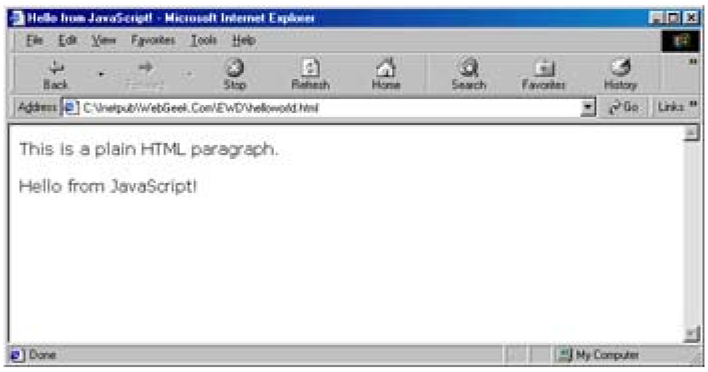
Adding JavaScript to Your XHTML File

To introduce JavaScript in your pages, you first need to let the browser know you're about to do so, using the `script` element. This element contains the script to be introduced into your page. The element takes a `type` attribute that defines which language you'll be using. You can place this container just about anywhere within your Web-page file, though it's often inserted into the head element or just after the `body` element opens.

To get started, I'll create a short script that mimics the traditional first program completed by those learning almost any programming language—the Hello World! program. Listing 18.1 (also on the CD-ROM) shows the script, and Figure 18.4 shows the display of this file in IE 5.5.

Listing 18.1: HelloJavaScript.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Hello from JavaScript!</title>

</head>

<body>

<p>This is a plain HTML paragraph.</p>

<p>

<script type="text/JavaScript">

document.write ("Hello from JavaScript! ")

</script>

</p>

</body>

</html>
```


Hello from JavaScript! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History
Address C:\Inetpub\WebGeek.Com\EWD\helloworld.html
Go Links
Done
My Computer

This is a plain HTML paragraph.

Hello from JavaScript!

Figure 18.4: Hello from JavaScript!

Hiding JavaScript from Noncompliant Browsers

According to XHTML specifications, a browser that doesn't understand an element is supposed to ignore it. Unfortunately, that depends greatly on the programming processes used to create the browser. Not all behave properly when encountering the `script` element and may display the contents of that element in the page as plain text. To prevent this, you can use an XHTML comment to hide the script from these browsers.

As discussed in Chapter 7, comments have a slightly different structure than what you're used to working with. A comment looks like this:

```
<!-- this is an HTML comment -->
```

In order not to confuse browsers that *do* understand JavaScript, you use the JavaScript comment notation of a double slash (//) before the end of the HTML comment. So the comment now looks like this:

```
<!-- this is an HTML comment with JavaScript notation added //-->
```

Warning Although the browser won't display the comments when the page is viewed normally, they are visible if someone chooses to view the source file. Never place sensitive information in comments.

Listing 18.2, here and on the CD-ROM, shows the file when the comment is incorporated.

Listing 18.2: HelloJavaScript2.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>Hello from JavaScript!</title>

</head>

<body>

<p>This is a plain HTML paragraph.</p>

<p>

<script type="text/JavaScript">

<!--

document.write ("Hello from JavaScript! ")

 // -->

</script>
```

```
</p>

</body>

</html>
```

Tip You can spread comments over more than one line, as seen in HelloJavaScript2.html. Doing so allows for improved readability and lets you immediately locate the beginning and end of each comment.

Ready to get into the animation? Keep reading!

Animation through the Mouseover Event

Your script first needs to determine whether the visitor's browser supports not only JavaScript but also the image object within JavaScript. The major browsers have done so since Netscape Navigator 3 and Internet Explorer 4.

Tip One of the frustrations of Web development is the slightly different implementations of browser features. JavaScript encounters this on two fronts: First, Netscape's JavaScript has two versions, 1 and 1.1. Second, Microsoft's implementation—known as JScript—is more a dialect than a true implementation. It's similar to JavaScript but different enough to make it a distinct language (like U.S. English vs. British English). Most of the time, the "reader" (the browser) understands what you mean, but occasionally it scratches its electronic head!

To test for support, the first portion of the script needs to include the conditional if statement. A conditional statement operates like it sounds here: If this condition is met, proceed with these directions. In JavaScript, it appears as:

```
if (document.images) { ... }
```

This statement says that *if* the browser can process `document.images` (determined by its true/false value), then continue with the actions enclosed in the curly braces (represented by the ellipse for example purposes only).

Defining Variables

I now need to define several *variables*. A variable is a word or *string*—a set of alphanumeric characters undivided by spaces—that is assigned a value. This concept is probably familiar to you from basic algebra class. For example, in the following equation, `x` is a variable:

$$x + 3 = 10$$

The letter `x` holds the value of 7.

In JavaScript, a variable can hold a numeric value as in this example, or it can hold words or phrases, file names, even complex mathematical operations, or XHTML markup.
In the script written here, the variables will hold file names—the names of the image files to be used in the animation. I'll be animating three images, so I'll create six variables—one each for the on and off state of each variable. The variables are for three images that serve as links to the services, portfolio, and "about us" section of my WebGeek, Inc. Web site, at www.webgeek.com.

Variables are named with the following syntax:

```
var ServicesOn = new Image()

        ServicesOn.src = "serviceson.gif"
```

This code says, "I've created a variable named `ServicesOn`, which is being assigned the value of a new image." The second line defines the source (.SRC) of the image, referenced by the variable `ServicesOn`, as the file `serviceson.gif`.

Tip Wondering about the variable name chosen here? Programmers have long found that by naming variables with words or phrases that have real meaning, rather than nonsense words or simple alphabet letters, it's easier to read through the code and know what the program is doing. In this case, ServicesOn refers to an image and link combination for services, with *on* indicating the on state of the animation vs. the off state.

Warning In JavaScript, variable names are case sensitive. That is, ServicesOn in the previous code is distinct from serviceson. You must maintain the same case throughout when referencing variables.

Here is the entire list of variable definitions:

```
if (document.images) {
var ServicesOn = new Image()
        ServicesOn.src = "serviceson.gif"
            var ServicesOff = new Image()
        ServicesOff.src = "servicesoff.gif"
            var PortfolioOn = new Image()
            PortfolioOn.src = "portfolioon.gif"
        var PortfolioOff = new Image()
            PortfolioOff.src = "portfoliooff.gif"
            var AboutOn = new Image()
        AboutOn.src = "abouton.gif"
            var AboutOff = new Image()
            AboutOff.src = "aboutoff.gif"
}
```

Defining Functions

Next, I need to define two functions. A *function* is a piece of JavaScript code that gives an action instruction; that is, it says, "Do this!" to the browser.

The first function tells the browser to pull the "on" image file into the image source. Specifically, I'm creating a function named on that will operate on the object `imgName` (I'll talk about this in a moment). If the browser understands the document image object, it will concatenate (add together) the image name and the string `On.src` and equate it to `document[imgName].src`. If that still sounds confusing, stay with me for another minute here, and it should become clearer. Here's the code:

```
function on (imgName) {
            if (document.images)
              document[imgName].src = eval(imgName + 'On.src')
          }
```

This second function is the same, except that it deals with the off-state images:

```
function off (imgName) {
  if (document.images)
    document[imgName].src = eval(imgName + 'Off.src')
}
```

Now all that's left is closing out the `script` element. Here is the entire element with its contents:

```
<script type="text/JavaScript">
          <!--
if (document.images) {
var ServicesOn = new Image()
        ServicesOn.src = "serviceson.gif"
            var ServicesOff = new Image()
        ServicesOff.src = "servicesoff.gif"
            var PortfolioOn = new Image()
```

```
    PortfolioOn.src = "portfolioon.gif"
  var PortfolioOff = new Image()
    PortfolioOff.src = "portfoliooff.gif"
    var AboutOn = new Image()
  AboutOn.src = "abouton.gif"
    var AboutOff = new Image()
    AboutOff.src = "aboutoff.gif"
}
function on (imgName) {
    if (document.images)
     document[imgName].src = eval(imgName + 'On.src')
   }
function off (imgName) {
    if (document.images)
  document[imgName].src = eval(imgName + 'Off.src')
     }
// -->
   </script>
```

Now all I need to do is add a few new attributes to XHTML elements that you're already familiar with—anchor and image tags.

Adding Event Handlers to Your XHTML

An *event handler* is a function defined in the JavaScript that is assigned to a particular event. I'll be dealing with the `onmouseover` and onmouseout events.

Tip JavaScript event names all begin with on, indicating that they are activated when the event occurs.

The following XHTML snippet includes the anchor and image elements for the first animation:

```
<a href="services.html"><img src="serv.gif" width="129" height="59"
 alt="services" border="0" align="left">
```

When I add the information necessary to trigger the JavaScript event, the elements look like this:

```
<a href="services.html"
onmouseover="on('Services') "
onmouseout="off('Services') ">
<img src="servicesoff.gif"
width="129"
height="59"
name="Services"
alt="services"
border="0"
align="left" />
```

Tip I've shown these new examples with each attribute on a new line in order to allow you to read and digest them more easily. Once you're comfortable working with them, you can compose them on single lines if you want.

Warning While viewing the source code of other Web sites, you may have noticed that the event names onmouseover and onmouseout are more often seen written in "camel case," as onMouseOver and onMouseOut. With XHTML, all element and attribute names must be lowercase, so the adjustment is made here. Note that the lowercase requirement only applies to the event-handler attribute names. Variables and functions can use "camel case."

These two newest attributes are these event handlers:

```
onmouseover="on('services') "
onmouseout="off('services') "
```

They say, "When the mouse is over this linked area (defined as the space occupied by the image), perform the function on, operating on the image named *services*. When the mouse moves out of the linked area, perform the function `off` on the image named *services*.

Looking back at the function on, I can now insert the pieces missing earlier:

```
function on (imgName) {
        if (document.images)
        document[imgName].src = eval(imgName + 'On.src')
      }
```

This function, when called from this anchor and image element set, now says, "If you're a browser that understands the `document.images` object, the source file for `document.services.src` should be `ServicesOn.src` (the combination of the `imgName`, or `Services`, and the string `On.src`), which was previously defined in the variables section of this script (`serviceson.gif`)."

The entire set of anchor and image tags now looks like the following:

```
<a href="services.html"
onmouseover="on('Services') "
onmouseout="off('Services') ">
<img src="servicesoff.gif"
width="129"
height="59"
name="Services"
alt="services"
border="0"
align="left" /></a>

<a href="portfolio.htmla"
onmouseover="on('Portfolio') "
onmouseout="off('Portfolio') ">
<img src="portfoliooff.gif"
width="129"
height="59"
name="Portfolio"
alt="portfolio"

border="0"
align="left" /></a>

<a href="about.html"
onmouseover="on('About') "
onmouseout="off('About') ">
<img src="aboutoff.gif"
width="129"
```

height="59"

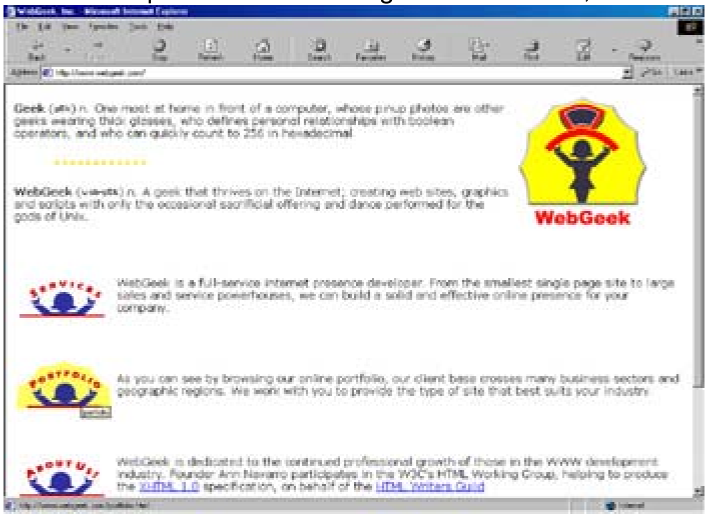
name="About"

alt="about"

border="0"

align="left" /></a>

Figure 18.5 shows the Portfolio image in the on state, mimicking the main WebGeek logo when the mouse is placed over the image. To see it "live," view this page at `http://www.webgeek.com`.



Geek (gēk) n. One most at home in front of a computer, whose pinup photos are other geeks wearing thick glasses, who defines personal relationships with boolean operators, and who can quickly count to 256 in hexadecimal.

WebGeek (web-gēk) n. A geek that thrives on the Internet; creating web sites, graphics and scripts with only the occasional sacrificial offering and dance performed for the gods of Unix.

WebGeek is a full-service internet presence developer. From the smallest single page site to large sales and service powerhouses, we can build a solid and effective online presence for your company.

As you can see by browsing our online portfolio, our client base crosses many business sectors and geographic regions. We work with you to provide the type of site that best suits your industry.

WebGeek is dedicated to the continued professional growth of those in the WWW development industry. Founder Ann Navarro participates in the W3C's HTML Working Group, helping to produce the XHTML 1.0 specification, on behalf of the HTML Writers Guild

Figure 18.5: The `onmouseover` event in action

Online Resources for JavaScript

A wealth of JavaScript information and tutorials are available on the Web. Here are some of the better sites to check:

- JavaScript Tip of the Week is an archive of 30 JavaScript coding tips, complete with source code. Find it at `http://webreference.com/javascript`. (This site is no longer actively maintained, but it still is a gold mine of information.)
- WebCoder.com is a reference on JavaScript and Dynamic HTML. It contains a useful JavaScript support chart. Find it at `http://www.webcoder.com`.
- JavaScript.com, provided by the internet.com information conglomerate, can be found at `http://www.javascript.com`.

Visual Effects with CSS

You can achieve a similar mouseover effect using style properties when combined with a little bit of JavaScript. The combination of all three technologies, (X)HTML, JavaScript, and CSS, is commonly referred to as DHTML, or Dynamic HTML. DHTML has never formally been codified into a specification; instead, it's a marketing term coined to reflect the use of all three pieces to create dynamic effects in a Web page.

In Listing 18.3, I'll look at what's needed to create a simple color change when the user moves the mouse over a portion of text. The listing is also on the CD-ROM.

Listing 18.3: Textover.html

```
<!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" xml:lang="en" lang="en">

<head>

<title>DHTML Mouse Over</title>
```

```
</head>

<body>

<p>Rollover effects can be achieved by using style properties

  combined with a mouseover and mouseout event. No script elements

  are necessary, and the text chosen doesn't even need to be a

  link.

  <font color="#000000" onMouseOver="this.style.color = '#0000FF'"

  onMouseOut="this.style.color = '#000000'"Roll your mouse over

  this sentence, and watch it change color!</font>

</body>

</html>
```

Tip After advancements over the last year, nearly all of the major browsers support the mouseover effect, including IE 4 or higher, Navigator 6, Opera 5, and NeoPlanet. You'll only run into trouble with Navigator 4.x. If you use this effect, make sure it's not a critical component of the information you're trying to convey.

Audio and Video

Thus far in the discussion about multimedia, I've focused on the animated and digitally created experience. However, *multimedia* also includes sound and traditional video footage, albeit in special digitized formats.

Movies for the Web

Video on the Web comes in a multitude of popular formats, including MPEG, AVI, and RealVideo. The most complicated aspect of providing video on the Web is the disparity in implementations between Navigator and Internet Explorer as to how they recognize and play these files. A variety of plug-ins help the situation, including the Windows Media Player, QuickTime, and of course the Real Audio/Video Player.

The most cross-platform and cross-browser compatible method of serving this content is to provide a simple link to the multimedia files. Once downloaded, any media player on the visitor's computer can play them. The link is a simple anchor:

```
<p>Watch the clip Dave made for the <a href="http://www.snorf.net/
  screensavers.avi">ScreenSavers</a> intro!</p>
```

The downside to this approach is the long wait for large video files to download, especially over 56K modem connections, which are still the majority of connections today despite the growing high-bandwidth market of DSL, ISDN, and Cable Internet service.

MP3: The Net Audio Revolution

Unless you've had your head buried in the proverbial sand over the past year, you couldn't help but have been inundated with information about the new digital music format known as MP3. MP3 refers to MPEG 1 Layer 3 and is taken from the file extension used when naming the files, .MP3. MPEG is an abbreviation for the Moving Picture Experts Group, not an organization you'd immediately connect with music; however, the "layer" in MP3 gives the clue that sound is one layer of the moving-picture specifications put out by this group. MP3 is a high-quality, compact, and now—with new devices out on the market—highly portable format for music. Results can rival that of retail CDs or even direct digital recordings.

The desirability and quality of such recordings, and the fact that software technology has advanced enough for individuals to copy purchased CDs into the older digital. WAV format and then convert those files into MP3s, has led to considerable controversy. Add to the mix the "community of friends" that share these files using a service provided by Napster, and society is on the brink of a digital music revolution. The cases filed surrounding the online sharing of music are still before the courts, but many suspect that the horses will not put up with being rounded up back into the corral this time.
Poking a little fun at the situation, while at the same time demonstrating the viability of providing streaming audio on the Web, is Bob River's Twisted Tunes Web site (`http://www.twistedtunes.com`). There, you can listen to *I Want My MP3*, a song written to mimic *I Want My MTV* by Dire Straits (see Figure 18.6).

Figure 18.6: Streaming audio at it's funniest at Bob River's Twisted Tunes

Successfully embedding audio files as the Twisted Tunes site did requires some sophisticated JavaScripting that goes beyond the scope of this book. Unfortunately, alternate methods of embedding audio are not cross-browser compatible, as you discovered is the case with video on the Web. That is, IE handles it using the W3C-approved `object` element, whereas Navigator requires use of the nonstandard `embed` element. Some designers choose to nest one within the other, betting on the "ignore it if I don't know it" dictum to which browsers are supposed to adhere. However, in a world that increasingly requires fully valid XHTML documents, such an approach is not feasible. The only remaining alternative is to provide links to the sound files that the user can activate on demand. This requires no functionality other than a simple anchor element and the user having some form of sound-player software loading on their machine (default software is provided on nearly all computing platforms).

If I were to provide a link to the theme from one of my favorite movies in MP3 format, I could write:

```
<p>Listen to the <a href="stargate.mp3">Stargate Theme</a>
```

Because my machine has MP3 files associated with the Music Match Jukebox, that program will launch and then begin playing the file as it is downloaded from the Web site.

Help from the W3C

So far here, the options for audio and video are pretty slim: Use nonstandard XHTML elements—which invalidate your documents and require plug-ins that are always a hit-or-miss proposition for the user—or provide a simple link to the file. Fortunately, relief may come, courtesy of the W3C.
The W3C recently finalized their recommendation for Synchronized Multimedia Integration Language, or SMIL (pronounced *smile*). SMIL is an XML vocabulary that manages the synchronized presentation of images, sound, video, text, and more. Presentations can be as simple as a single video clip and can be extended to include a text caption of the voice portion of the video, pauses and starts in the playback, and much more.

The only complicating factor is that like some of the other technologies discussed in this chapter, viewing a SMIL presentation requires the use of an external application. There's hope, however, that as Web browsers move into the increasingly XML-centric world, native support for SMIL will be included, making it worthy of mention here.

The creation of SMIL presentations can be done, to some extent, by hand, but the expectation is that authoring tools will produce most of them. An early entrant into that field is the Oratix GRiNS development environment and SMIL player.

Tip You can download and try out this tool for a limited time by visiting http://www.oratrix.com/GRiNS/.

I've placed two example SMIL presentations online at `http://www.webgeek.com/books/ewd/smil`, one that contains a video clip made by my husband Dave as an entry for ZDTV's Screensavers Web-cam introduction contest. The second is a slide show featuring photographs taken in Amsterdam. Figure 18.7 shows a still shot of the second.

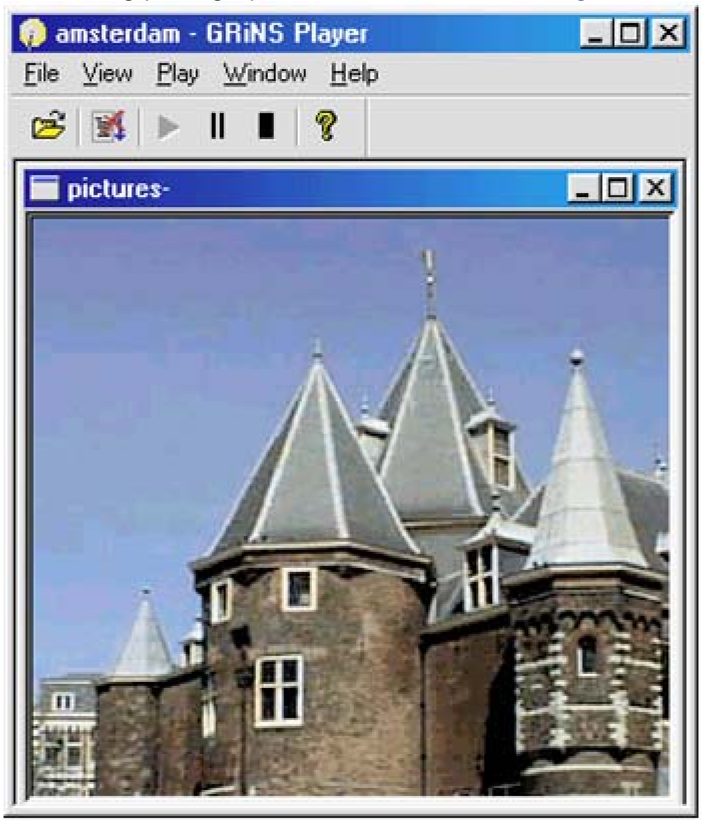


Figure 18.7: Images can be swapped in a virtual slide show using SMIL.

Tip A more thorough discussion of SMIL, including syntax examples, can be found in *Mastering XML*, also from Sybex.

Summary

Animation can be a fun way to add some flash to your Web site. Tools for creating interactive and engaging movies, games, and utilities have become very popular on the Web, though they use a proprietary technology from Macromedia. Through learning just a touch of JavaScript, you can provide instant feedback to your visitors by creating events based on the actions and location of their mouse pointers. You can find additional tutorials and resources for JavaScript in many places online. Audio and video on the Web present challenges today, though with the XML-based SMIL the hope is that browsers will natively support multimedia in the near future.

Chapter 19: Doing Business on Your Site

Overview

Congratulations! Your sales pitch was a smashing success. Your site is informative, useful, and pleasing to the eye. Your visitors can't wait to buy your products! Now what?
The key to successful online sales rests in *customer convenience*. If visitors can't easily order or obtain your goods, they aren't likely to purchase them, especially if the same items are commonly available elsewhere. Users also need to be confident that any sensitive information they transmit will be secure.

This chapter covers the following topics:

- Understanding Web-server security issues
- Introducing firewalls
- Protecting your private data
- Dealing with commerce and Internet security
- Looking at how secure servers work
- Exploring payment alternatives
- Getting a handle on tax matters

Keys to Doing Business Successfully Online

This chapter covers two complex issues in Web design, and both Web security and e-commerce have books devoted solely to them. In-depth coverage of these topics is beyond the scope of this book, but I hope to at least give you an overview of the main areas so that you have more knowledge of what is involved when you do need to learn more about these topics.

Although the elements of successful online sales are similar to those for mail-order catalog sales, important differences exist. Customers can browse a catalog at their leisure, taking time to review their choices before purchasing, and order at any time via a handy 24-hour toll-free telephone line or by mailing or faxing a printed order form—all without ever leaving home.

Online, however, the average consumer expects faster results because the Internet gives every appearance of being instantaneous. At the same time, consumers are concerned about security on the Internet and want assurance that their identity and purchasing methods during the transaction are protected.

A Primer on Web-Server Security

If you plan to do business on the Net, you must address two major security concerns in order to have a safe and secure Web presence:

- The server, which is normally at a known Internet location and may present an attractive target to "crackers" looking for information about your company or your customers
- The data for any transaction, which may be exposed to unauthorized access at any point as it travels between your server and your customers

When considering the security of your server, you need to ask the following questions:

- Is your server connected to your company's LAN? (If it is, you should put up a firewall.)
- Does it store sensitive documents, even though they aren't linked to the Web site? (If so, that's another reason to put up a firewall.)
- Can an unknown individual execute commands on the host machine that could change, damage, steal, or destroy data?
- Does your server have adequate protection against misuse by others, including denial-of-service attacks and the routing of unsolicited e-mail through your mail servers?

When considering the security of the data that is passed back and forth between your site and your customers, you need to ask the following questions:

- Can someone eavesdrop on a network connection point? Eavesdropping can occur anywhere between the customer's browser and your server, including:
    - At your customer's ISP
    - At your ISP
    - At an ISP's regional or "backbone" provider
    - Even on your own servers, if you have a full-time connection
- Have you protected against eavesdropping on commerce transactions by providing a secure server for consumer use? A secure server encrypts data passed to and from it. Having a

secure server doesn't prevent eavesdropping, it just assures that any intercepted information will be encrypted.

Firewalls: (Carefully) Bringing the Outside In

The whole process sounds simple enough: All you need is the phone company to install a new line (often called a *pipe*), connect the pipe to a router, connect the router to the Web server, and then connect the Web server to your LAN. Ta-da! Your company is on the Net!

Now, you wouldn't show off your brand-new office building with its gleaming new furniture and computer systems with a swank soiree and then leave the doors open when you go home. For the same reason, you shouldn't wire up your company and then not take measures to keep the curious and the bad guys out. Still, your employees must have access.

The most important security measure you can take is implementing a *firewall*. A firewall is a hardware and/or software system for your servers that allows approved traffic both in and out according to the constraints you impose when setting it up. It consists of two primary pieces:

- **The Screening Router** This device provides the primary connection between the "trusted" network (your LAN) and the "untrusted" network (the Internet at large). It examines incoming data packets for information such as the machine name where the message originated, the IP address, or the Internet protocol type and then either forwards the packets to the appropriate servers or blocks them.

Tip *An IP address is a numerical identifier for a specific machine or device connected to the Internet or an internal network.*

- **The Application Gateway** The screening router performs what are known as the *low-level functions*—the basic tasks that don't require much decision making beyond an approval or a denial of a request. The application gateway handles more complex tasks, such as routing incoming FTP or Telnet requests and determining where password authentication takes place before access is granted.

Because a firewall protects expensive equipment and confidential data, it must be installed and configured by experts. A poorly designed system is like a cheap dead-bolt lock: It gives you a false sense of security, while making it possible for even the most casual snooper to break in. If a firewall is not already in place at your network's Internet connection, work with your system administrator to find a qualified professional to design one for you.

Keeping the Private Information Private

Some of the ways that you provide security can become quite technical; others, however, are just plain common sense. While ensuring that you've taken all the appropriate technical steps, don't overlook the practical aspects, such as the following:

- **Store Only Public Documents on a Public Web Server** Keep items such as the following on a server that *doesn't* directly connect with the outside world:
  - A map of your company network
  - A list of hardware and software configuration
  - The company address and telephone directory
  - Internal letters and memorandums
  - Your customer database
  - Personnel files

Although a good firewall may prevent someone from rooting around where they shouldn't, don't tempt fate by storing these documents where they'd be immediately available after a security failure.

- **Keep Centralized Records of All Access Accounts** Designate one person on your staff to approve and distribute all login accounts for your Web server. Typically, the person is someone in your Information Systems division or whoever is responsible for your current LAN access.

- **Change Passwords Frequently** Require users to change passwords frequently (at least every two months). A good password is a combination of at least six letters, numbers, and characters. Never allow users to use their birthday, a family member's name, or any other readily "guessable" word or phrase. Some system administrators do not even allow any words that can be found in the dictionary.

- **Audit All Access Logs Periodically** Your Web server's access logs provide detailed information on what type of requests were made, when, and by whom. (Identification may be only by an IP address or a machine name, rather than by user ID.) Reviewing this data can alert you to potential trouble spots or enable you to identify malicious individuals by their attempted usage patterns.

Security on a Virtual Server
Not every company has the resources to maintain a full-time connection to the Internet. In those cases, most companies contract with an ISP to set up what's known as a *virtual server*—space on an ISP's Web server. Through software configuration, the outside world sees your site as its own machine; hence, the *virtual* in virtual server.

The security of your data on a virtual server is only as good as the security the ISP has in place for the entire system. Here are some questions to ask an ISP system administrator in this regard:

- What security measures have they taken against unauthorized access to Web documents?
- Do they allow users to run command-line tasks from CGI scripts or Server Side Includes? (Doing so opens a potential security hole, though it isn't always indicative of immediate danger.)

Tip Server Side Includes (SSIs) are processes that run on the server. Using SSIs, a Web designer can instruct the server to *include* material from another source in the document currently being processed. An SSI is often used to insert the current date and time or to incorporate repetitive blocks of XHTML markup, such as navigation bars or address and copyright notices. Because it allows command-line tasks, it permits a savvy user to run commands never intended.

- Who in the ISP's organization has access to your data? Is it only the Webmaster and system administrator, or can anyone in the company get into the server?
- Will their access logs be available to you?
- Will they assist you in dealing with any attempted breaches to the system?
- Do they subscribe to the CIAC-Bulletin, distributed by the U.S. Department of Energy's Computer Incident Advisory Capability group? If not, how do they keep abreast of new developments in Internet security?

On your end, you can take several steps to protect your virtual Web server:

- Don't share your account with anyone: neither the dial-up access nor your Web-server space. Internet access accounts are available for as little as $5.00 a month, so no economic benefit offsets the risk. In fact, most ISP contracts prohibit sharing an account.
- Don't use the same password for multiple accounts or services, such as your dial-up password and your FTP access password. If one is compromised, all others will be.
- Test all CGI scripts for security before uploading them to your live site. If you aren't comfortable judging this, consider hiring a professional programmer to review them.

Commerce and Internet Security

Now that you understand what's involved in protecting your servers and the data residing on them, the next step is to protect your customer's data while it's in transit to you.

Recent waves of public concern over the security of Internet transactions have ranged from the conservative to the paranoid. Unfortunately, much misinformation exists. In order to cut through it all, I'll look again at a short form that might be used on a commercial Web site and consider the risks that this example would face. (For brevity's sake, I've removed many fields.)

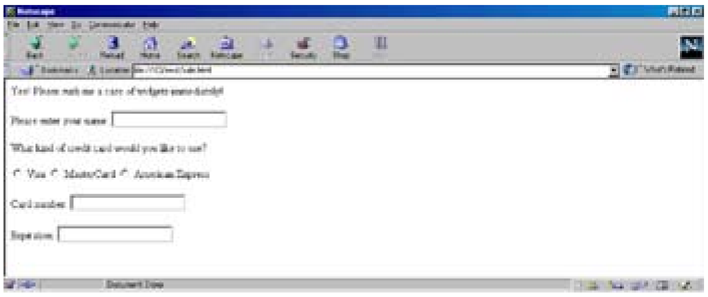
Tip For a quick review of forms and processing, see Chapter 8.

Listing 19.1, also on the CD-ROM, presents the XHTML that produced the form shown in Figure 19.1.

Listing 19.1: The XHTML for a simple order form

```
<form method="post" action="http://www.foo.com/cgi<->bin/orderform

.cgi">
```

```
<p>Yes! Please rush me a case of widgets immediately!</p>

<p>Please enter your name: <input type="text" name="name" size=

"20" />

<p>What kind of credit card would you like to use?</p>

<p><input type="radio" name="card" value="Visa" /> Visa

<input type="radio" name="hardware" value="MC" /> MasterCard

<input type="radio" name="hardware" value="AmEx" /> American

  Express</p>

<p>Card number: <input type="text" name="number" size="20" />

<p>Expiration: <input type="text" name="exp" width="6" /></p>

<p<input type="submit" value="Submit!" />

</form>
```



Yes! Please rush me a case of widgets immediately!

Please enter your name:

What kind of credit card would you like to use?

Visa MasterCard American Express

Card number:

Expiration:

Figure 19.1: A simple form for ordering on the Web

Submitting this form sends the data to the CGI script for processing in URL-encoded format. Here's how the format looks:

name=Jane+Smith&card=Visa&number=4444-4444-4444-4444&exp=02%2F03

Pretty obvious what the data is, isn't it? For the most part, no one (or nothing) but your CGI script sees the data. Although it is sent "in the open," or uncoded, it's not quite as bad as wearing a nametag with your credit-card number on it in a busy shopping mall.
To obtain the data, someone must purposefully hunt for it, a process that's usually done with *packet sniffers*, or programs designed to capture and read each packet of data sent to or through a specific Internet hub or subnet. (A *subnet* is loosely defined as a network of computers such as the LAN in your office.) Such programs have legitimate uses—they can help network administrators locate system problems or aid in security checks. Unfortunately, they can just as easily be used to snoop for account names and passwords or unencrypted credit-card data.

Some Internet users feel that the risks associated with sending personal data through e-mail or as a form transaction are minimal. In concept, I agree. You're at similar risk every time you hand your credit card to a waiter at a restaurant or give credit-card information over the telephone to a catalog salesperson. (I realize that most waiters and sales reps are honest individuals and certainly don't mean to imply otherwise, but theft does happen.) In reality, however, the common perception is that you should avoid sending such data unencrypted at all costs. If your potential customers believe in a security risk, it makes sense to demonstrate to them that you are taking reasonable precautions.

You need to consider these legitimate concerns. One way you can alleviate the concerns is to allow first-time customers to contact you offline, perhaps over the phone. You can obtain the relevant data—name, address, shipping information, and payment method—and then issue an account number. When customers are ready to make an online purchase, they need only enter the account number. You look

up the customer's data in your database (which is not connected to the Web) and then process it as you would any online sale. Although careful, this method does have the drawback of discouraging impulse purchases, which are very profitable to any Web site.
The most popular way to deal with online-transaction security issues is to use what's known as a secure server. A *secure server* is an extra layer of software that encrypts specified transactions, such as sending URL-encoded data from a form to the CGI script for processing.

Transaction Security through the Secure Sockets Layer

The Secure Sockets Layer (SSL) was developed and proposed to the W3C by Netscape Communications Corporation. It's an encryption scheme that protects transactions (the exchange of data between computers, regardless of content) for HTTP requests (Website transactions). It allows for the browser and the server to verify each other's identity, and it encrypts all data in transit between the two. Netscape Navigator and Internet Explorer, as well as most other recently released browsers, support the use of SSL.
The identity-verifying stage of the process involves a *server certificate* that is granted by a certifying authority. Several authorities exist with varying application processes and fees, including:

- VeriSign (http://www.verisign.com/), which was the first and continues to be the most widely used
- GTE CyberTrust (http://www.cybertrust.gte.com/)
- Thawte Consulting (http://www.thawte.com/)

The certificate authorities are generally considered trustworthy because if they ever failed to provide security, they would lose all credibility and be shut down by thousands of lawsuits.

Tip Before choosing a certifying authority, be sure to investigate which browsers recognize its certificates. Navigator and Internet Explorer, along with many other browsers that support SSL, recognize VeriSign and Thawte certifications, but support for others isn't universal.

Tip If you'll be hosting your site at an ISP rather than running your own Web server over a full-time Internet connection, consult with the people at the ISP before purchasing a certificate of your own. Many ISPs allow clients to use their general site certificate, saving you considerable cost and effort.

Applying for a Certificate

Once you've decided on a certifying authority, it's time to complete the online-application process. The steps that follow are for VeriSign, but other certifying authorities use a similar process. Before you begin, you need to generate a key pair—basically, a digital password—and a certificate-signing request (CSR). You'll find instructions for doing so in your server software documentation.

When you fill out the form, you'll be asked to supply a Dun & Bradstreet number (DUNS number). VeriSign must verify your company's identity, and the quickest way to do so is through your DUNS number. If you don't have one, you can apply by using a form supplied in the middle of the VeriSign application process.

Tip Dun & Bradstreet is a respected business resource that compiles background information on companies and their executives.

If you don't have a DUNS number or would rather not use your DUNS number, you can verify your company's identity by providing, via postal mail, a copy of one of the following:

- Articles of incorporation
- Partnership papers
- Business license
- Fictitious business license
- Federal Tax ID confirmation

Once you've generated your key pair and CSR, you need to fill out the enrollment form, choosing from one of several service packages offered by VeriSign at http://digitalid.verisign.com/server/index.html.

Be prepared with the following information:

- Your secure-server software vendor.
- A challenge phrase. This phrase will be used as a password of sorts between you and VeriSign, to authenticate requests to revoke a certificate.
- Name and contact information for the following:

    - **Technical Contact** This person must be employed by your organization and be authorized to run and maintain the Web server. (If you are purchasing a certificate to be used on an ISP, your ISP will need to complete the ISP online enrollment form for you.)
    - **Organizational Contact** This person must also be employed by your organization and must be someone other than the technical contact. He or she must have the authority to enter into a binding agreement with VeriSign (generally a corporate officer, business owner, or other individual that enters into contracts for your company).
    - **Billing Contact** This person will receive all invoices from VeriSign and can be the technical or organizational contact as well.
- The type of service you need. Are you applying for a new digital ID? An additional ID or a renewal?
- Payment method. VeriSign accepts credit cards, purchase orders, and checks. (Purchase orders and checks must be received before your application is processed.) For fastest service, use a credit card.

Once you've submitted the application, the technical and organizational contacts will get e-mail that confirms enrollment. Once approved, the certificate will also be forwarded via e-mail. Instructions for installing it are provided with your secure-server software documentation.

Overall, you'll find that setting up a VeriSign account is actually quite difficult and time-consuming because of the various forms of proof required. But that's the way it needs to be. The purpose of a digital ID is to authenticate your company to anyone you may do business with on the Internet. If you were a customer, you would not want to entrust your financial transactions to anyone who has not been through a rigorous authentication process.

Collecting the Data

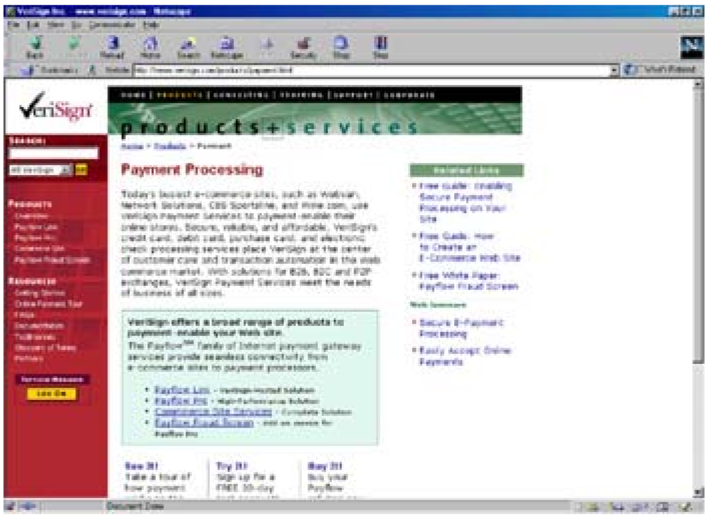
To keep your customer's data secure once the CGI script has finished processing it on the secure server, the data needs to remain on the secure server. If you're running your own Web server, that's not a problem: You can collect the data from inside your firewall and distribute it appropriately. But what if you're using a virtual server?

Downloading the data via e-mail, Telnet, or FTP subjects it to a trip across the Net in an unsecured environment—something that undoes all your earlier precautions and may well land in you trouble with the server-certificate authorities and/or the credit-card companies if the data is compromised. To maintain the required confidentiality, you'll need to access your orders on the secure server itself. You can do so by programming your CGI scripts to alert you via e-mail each time an order comes in, or you can make it a part of your routine to check the system periodically.

E-Commerce Options for those Without Dedicated Servers

At times, you may find that using your own secure server is impractical or not an available option. You can still provide your customers with an immediate online ordering experience by doing a little custom programming or by using one of several third-party solutions. Any transfers done with third parties simply should be sent over secure connections.

VeriSign Payflow
VeriSign, in addition to its secure-server products, offers real-time transaction processing for small- to medium-sized businesses that host their Web sites elsewhere. The Payflow family of services includes a Payflow Link service that is simple to set up and allows you to direct customers, for their transaction, to the secure Payflow order form behind VeriSign's secure servers. VeriSign then redirects the customer back to your site when the transaction is complete (see Figure 19.2).


products+services

Payment Processing

Today's busiest e-commerce sites, such as Walmart, Network Solutions, CBS Sportsline, and Wine.com, use VeriSign Payment Services to payment-enable their online stores. Secure, reliable, and affordable, VeriSign's credit card, debit card, purchase card, and electronic check processing services place VeriSign at the center of customer care and transaction automation in the Web commerce market. With solutions for B2B, B2C and P2P exchanges, VeriSign Payment Services meet the needs of business of all sizes.

VeriSign offers a broad range of products to payment-enable your Web site.
The Payflow℠ family of Internet payment gateway services provide seamless connectivity from e-commerce sites to payment processors.

- Payflow Link - VeriSign-hosted Solution
- Payflow Pro - High-Performance Solution
- Commerce Site Services - Complete Solution
- Payflow Fraud Screen - Add on service for Payflow Pro

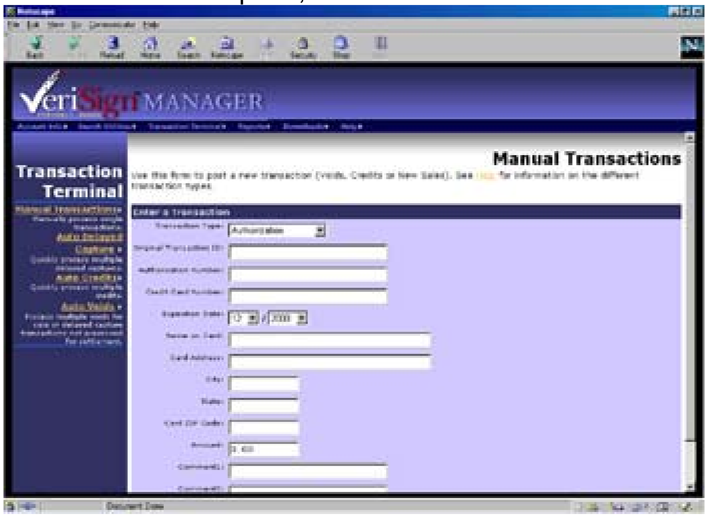
See It! Take a tour of how payment
Try It! Sign up for a FREE 30-day
Buy It! Buy your Payflow

Related Links
- Free Guide: Enabling Secure Payment Processing on Your Site
- Free Guide: How to Create an E-Commerce Web Site
- Free White Paper: Payflow Fraud Screen

Web Seminars
- Secure E-Payment Processing
- Easily Accept Online Payments

Figure 19.2: VeriSign Payflow transaction-processing options

As the site owner, you manage the page where the customer will be sent after the transaction, the look and feel of the order form, and how you will be notified of new sales. You can even customize an e-mail receipt that will be sent to the purchaser!
The VeriSign Manager Web site acts as your virtual credit-card terminal. You can enter transactions (including sales, credits, and voided transactions) manually (as shown in Figure 19.3), generate sales and settlement reports, and much more.


VeriSign MANAGER

Transaction Terminal

Manual Transactions

Use this form to post a new transaction (Voids, Credits or New Sales). See help for information on the different transaction types.

Enter a Transaction

Transaction Type:	Authorization
Original Transaction ID:	
Authorization Number:	
Credit Card Number:	
Expiration Date:	12 / 2000
Name on Card:	
Card Address:	
City:	
State:	
Card ZIP Code:	
Amount:	1.00
Comment1:	
Comment2:	

Figure 19.3: The site owner can use VeriSign Manager to manage transactions and set up order forms.

Yahoo! Store

Yahoo! was once just a Web directory; now it's a megaportal providing just about any service imaginable, including a shopping interface for business Web sites. Yahoo! charges a flat fee per month based on the number of items you have for sale. A small store, defined as 50 items or less, was priced at the time of this writing at $100 per month. Fees must still be paid to your merchant account, and Yahoo! does have a link to a favored provider.

Warning The charges on Yahoo!, in my opinion, are a little steep: a $45 basic monthly fee plus a discount rate and transaction fees. It's not unusual to see a minimum monthly fee, but it's usually less than the $45 fee and is usually offset by transaction fees and discounts collected.

Obtaining an Internet Merchant Account

In order to conduct any commerce over the Web involving credit-card transactions, you must have what's known as a *merchant account*. This is an account with a bank that processes the credit-card sales that you make, and *settles* them on a periodic basis (usually daily, or on-demand). The proceeds of the sale are then deposited in your business checking account. The merchant bank makes a profit providing this service by charging a *discount rate*, which is a percentage of each sale, and often a nominal per-transaction fee. Typical rates for small, virtual businesses are a 2.5–3.5 percent discount rate with a $.20 to $.30 per-transaction fee.

The ability to get a merchant account for small businesses that operate solely online has improved dramatically over the last two or three years. Merchant banks were very hesitant about providing accounts for online use, even for a traditional retail establishment that wanted to add online services. The banking community has become far more accepting, in part through the help of third-party processing services such as VeriSign.

If you do not currently have a merchant account, you will need to obtain one before you can conduct online commerce involving credit cards. VeriSign has a list of merchant account providers at `http://www.verisign.com/products/payflow/merchant.html`, and dozens of others can be found online. You can also try `http://www.yahoo.com/Business and Economy/Companies/Financial Services/Transaction Clearing/Credit Card Merchant Services`. When looking for a merchant account, keep the following in mind:

- Application fees should be nominal—less than $50. Avoid companies that require application fees that can range from $129 up to $399 or more.
- Unless your business is incorporated and has established business credit, the decision to open a merchant account will be made based on a review of the business owner's personal credit. It is possible to get an account with less-than-stellar credit, but a good credit rating of your own will allow you to get an account with a well-respected firm at a competitive discount rate.
- Avoid firms that require you to sign nonrevocable "leases" of transaction-processing software or hardware terminals that are three to four years in length. Software is available for purchase, or for a monthly use fee by application service providers, and terminals can be purchased on the open market and reprogrammed remotely by your provider.

Setting up your store couldn't be easier. The service guides you step by step in choosing sections, items, captions, and other necessary particulars to run the site. You can even set one up on a temporary basis for 10 days before converting it to a "real" store subject to Yahoo!'s pricing. (Temporary store transactions are scrambled and cannot be submitted for true credit-card authorization.) I've begun the process in Figure 19.4.

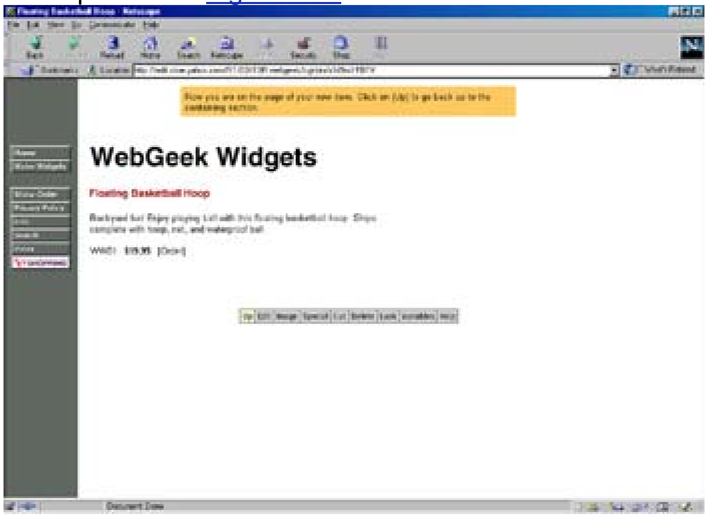


Figure 19.4: Setting up a storefront on Yahoo! Shopping

iBill

You may not qualify for a traditional merchant account, may not be willing to commit to lengthy contracts, or aren't sure of the viability of your idea. In those cases, you may wish to investigate the services that provide transaction processing *without* a merchant account. One such service is iBill (`http://www.ibill.com`). This, however, is a very expensive and time-consuming option. Services like iBill generally pay their merchants only once or twice per month. The iBill service pays for transactions occurring in the first half of the month on or after the first of the *following* month. For example, a transaction occurring on March 3rd wouldn't be paid to the merchant any earlier than April 1. The service charges 15 percent of all transactions, *and* holds 10 percent of all transactions as a reserve against returns or nonperformance by you, the merchant, which may result in a challenge of the charge by the consumer. Because iBill or any other similar service provider actually holds the merchant account in this arrangement, they require their clients to place that money into trust to cover their potential expenses incurred while handling your customers. The 10 percent reserve against returns is held a full six months and then begins to rotate out to you on the seventh month. Services like iBill are also very specific in what goods or service they will accept for sale, focusing primarily on subscriptions or one-time purchases. A list of refused products can be found at http://www.ibill.com/Services/Refuse.cfm.

Tax Issues

Setting up a virtual storefront immediately presents the new or small business owner with a unique set of commerce issues. With traditional retail stores, shop owners only need to learn about and comply with local sales or use tax laws because their customer base is primarily residents and businesses from the local area. Move that store to the Internet, and your customer base instantly becomes the global wired population. Does that really have any consequences beyond figuring out how to ship something to Sri Lanka? Unfortunately, the answer is it all depends.

The U.S. government, states, and local agencies have struggled for years with how to deal with the taxation of sales that occur on the Net. Congress finally passed the Internet Tax Freedom Act, which was recently extended in 2000 for another five years. The details can be read in full at http://www.house.gov/chriscox/press/coverage/2000/USATodayINDA.htm. In summary, it instituted a moratorium on new taxes on electronic commerce and will prevent new sales and use taxes on Internet access. Smoothing out conflicting local regulations is a key goal. Situations cited by sponsoring Senator Ron Wyden's (D-OR) office include:

- New York levies taxes on gross receipts on the "furnishing of information," but not on personal or individual information.
- Ohio taxes electronic transmissions and real-estate databases as providing objective data, but exempts news services as providing analysis.
- Texas taxes the transmission of electronic information and software in whatever form, but does not tax software sent out of state on a disk.

In some states, such as California, whether sales tax must be collected depends on whether the seller has a *nexus* within the state. The definition of a nexus is rather vague—it's a determination of whether the seller has a "significant connection" with the state, which would result in the seller being subject to a requirement to collect sales tax. Not very helpful, is it?

California has determined that the physical location of a Web server in that state does *not* constitute a nexus if that is the only "presence" the seller has in the state. Further, the U.S. Supreme Court has ruled that a seller is not obligated to collect sales tax if it does business in a state only by mail order.

California's Long Arm across the Net

While making the collection of sales tax a fairly simplistic issue, California has muddied the waters on another front. In addressing a growing fear of online fraud, a statute enacted in early 1997 applies consumer-protection provisions previously enacted for telephone and mail order sales outlets to online sales. Specifically, California requires that *any* seller that makes a sale to a California resident or business, regardless of nexus within the state, is required to provide on a Web site or via e-mail information regarding refund policies, a physical address for the business, and contact telephone numbers for complaint and dispute resolution. Additionally, vendors who don't perform within the bounds of those policies—perhaps not shipping products or not providing refunds within the prescribed time frames—are subject to prosecution.

This law remains relatively untested, as it takes a "long-arm" stance toward out-of-state vendors. California has taken the position that anyone making sales to its citizens must follow its rules—above and beyond the rules of the vendor's home state or country. Because the information the law requires is consumer-friendly, providing these details is a good idea, regardless of any statute that attempts to regulate it.

Collecting Sales Tax

According to the old cliché, death and taxes are only two sure things in life. Even then, taxes often seem as if they'll be the death of business owners! To help you wade through the widely varying regulations on the collection of sales taxes from state to state, Table 19.1 directs you to information straight from the source at each state's Web site (with notations for states that don't have sales tax).

Table 19.1: Sites for State Sales-Tax Information

STATE	WEB SITE
Alaska	No state sales tax. Contact local authorities.
Alabama	http://www.ador.state.al.us/salestax/Rules/index.html
Arizona	http://www.revenue.state.az.us/tpt/specevent/specevnt.htm
Arkansas	http://www.state.ar.us/dfa/taxes/salestax/index.html
California	http://www.boe.ca.gov/
Colorado	http://www.state.co.us/gov_dir/revenue_dir/TPS_dir/drp1002.html
Connecticut	http://www.drs.state.ct.us/index.html
Delaware	No state sales tax.
Florida	http://sun6.dms.state.fl.us/dor/taxes/sales_tax.html
Georgia	http://www.ganet.org/rules/index.cgi?base=560/12
Hawaii	http://www.state.hi.us/tax/tax.html
Idaho	http://www.idwr.state.id.us/apa/idapa35/0102.htm
Illinois	http://www.revenue.state.il.us/taxinformation/index.html#anchor48675
Indiana	http://www.ai.org/dor/pubs/bullets/salesib.html
Iowa	http://www.state.ia.us/government/drf/forms/sales.html#sales
Kansas	http://www.ink.org/public/kdor/taxinfo.html#5
Kentucky	http://www.state.ky.us/agencies/revenue/nondatedsalestaxforms.htm
Louisiana	http://www.rev.state.la.us/htmlfiles/tax_forms/tax_forms.asp
Maine	http://janus.state.me.us/revenue/RULES/intro.html
Massachusetts	http://www.dor.state.ma.us/dor/publ/pdfs/sls_use.pdf
Michigan	http://www.cis.state.mi.us/tax/
Minnesota	http://www.taxes.state.mn.us/salestax/forms/salestax.html
Mississippi	http://www.mstc.state.ms.us/taxareas/sales/main.htm
Missouri	http://dor.state.mo.us/tax/
Montana	No general sales or use tax.
Nebraska	http://www.nol.org/home/NDR/
Nevada	http://tax.state.nv.us/

Table 19.1: Sites for State Sales-Tax Information

STATE	WEB SITE
New Hampshire	No state sales tax.
New Jersey	http://www.state.nj.us/treasury/taxation/publsut.htm
New Mexico	http://www.state.nm.us/tax/
New York	http://www.tax.state.ny.us/Forms/sales_cur_forms.htm
North Carolina	http://www.dor.state.nc.us/taxes/sales/
North Dakota	http://www.state.nd.us/businessreg/oversales.htm
Ohio	http://www.state.oh.us/tax
Oklahoma	http://www.oktax.state.ok.us/btforms.html
Oregon	No state sales tax
Pennsylvania	http://www.revenue.state.pa.us/
Rhode Island	http://www.tax.state.ri.us/info/faqs/faqs&u.htm
South Carolina	http://www.sctax.org/frames/frameform.html
South Dakota	http://www.state.sd.us/state/executive/revenue/revenue.html
Tennessee	http://www.state.tn.us/revenue/faq.htm#SALES
Texas	http://www.cpa.state.tx.us/taxinfo/taxforms/01-forms.html
Utah	http://txdtm01.tax.ex.state.ut.us:80/sales/salestax.htm
Vermont	http://www.state.vt.us/tax/
Virginia	http://www.tax.state.va.us/bt_sutax.htm
Washington	http://dor.wa.gov/
West Virginia	http://www.state.wv.us/taxrev/consales.html
Wisconsin	http://badger.state.wi.us/agencies/dor/faqs/sales.html
Wyoming	http://revenue.state.wy.us/exciseframe.htm

Summary

Making the sale is an important part of many companies' Web sites. Doing it in an environment that protects both you and your customers can be difficult. A seamless shopping experience is the first sign of a successful ordering process. To complete the transaction, some form of security is required while dealing with customer's credit-card data. You can use secure servers or payment alternatives such as VeriSign Manager, Yahoo! Store, iBill, and more. Don't forget to research what jurisdiction you must collect sales tax from before you ship that product!

Chapter 20: Accessibility Issues

Overview

It has been estimated that at least 45 million Americans have some form of impairment involving the processing of visual or auditory information. When you add in the number of those with mobility or cognitive disabilities, the total is a significant percentage of the U.S. population. With more and more Americans (and persons of other nationalities) accessing the Internet every day, accessibility is becoming an increasingly important issue.

This chapter covers the following topics:

- Understanding the Web Accessibility Initiative
- How the Americans with Disabilities Act impacts the Web
- Evaluating your site's accessibility with Bobby
- WAI Quick Tips for Accessibility
- Additional Techniques

The Web Accessibility Initiative

In April 1997, the W3C announced the commencement of the Web Accessibility Initiative (WAI—pronounced *way*), which belongs to its Technology and Society domain of programs. The project is designed to promote and provide increased functionality on the Web for individuals with disabilities—from those with visual or auditory impairments to the physically handicapped.

The program moved forward with the opening of an International Program Office, a body that has coordinated with the traditional W3C Project work along with the efforts of organizations for the disabled, governmental entities, and individual companies. The targeted objectives include work in the following areas:

- **Technology** Work will continue on HTML and XHTML markup that is complementary to the existing ICADD (International Committee for Accessible Document Design) structures. Projects have included converting HTML documents to a DTD that supports Braille, large print, and voice synthesis and enhancements to forms and tables. This group has also been working on enhancing Cascading Style Sheets to include support for speech output.
- **Style Guide** An online style guide for tool developers has been developed, known as the Authoring Tools Accessibility Guidelines. It offers suggestions for the creators of HTML-authoring programs and other Web-related software. These guidelines give advice on how to incorporate accessibility features into programs; they will also prompt authors using those authoring tools to generate accessible Web sites.
- **Guidelines for Use of New Technologies** Although the W3C doesn't intend to suggest policy, it is dedicated to providing the necessary mechanisms for authors to produce accessible documents. Existing mechanisms include `alt` attributes for images, which allow the visually impaired to receive information in place of the designated image.
- **Education of Web Developers** Many developers and designers may not be aware of how their design choices affect those with disabilities. The Web Content Accessibility Guidelines, a document of recommendations, provides a discussion of issues, how they can be overcome, and techniques for coding the solutions.
- **Research and Development** This group will continue to work with the W3C host institutes—INRIA, Keio University, and MIT—on automatic certification tools for Web content and accessibility in scripting interfaces and XHTML display tools.

The Americans with Disabilities Act

In the United States, the equal dissemination of information to all citizens is one of the goals of the Americans with Disabilities Act, commonly known as the ADA. In general lay terms, this law asserts the rights of the disabled to gain equal access to public services. Equal access includes familiar accommodations, such as wheelchair ramps at public buildings, sign-language interpreters in courtrooms, and Braille versions of official documents.

Cities, states, and other public entities are required to provide reasonable alternatives if information or services are normally delivered by means that aren't accessible to an individual with a qualifying disability. As more public entities develop a Web presence, these concerns will apply to them for online content.
WWW Here are some excerpts from ADA. You can find the entire text in Section IV at http://www.usdoj.gov/crt/ada/t2hlt95.htm.

State and local governments must ensure effective communication with individuals with disabilities.

*…*Where necessary to ensure that communications with individuals with hearing, vision, or speech impairments are as effective as communications with others, the public entity must provide appropriate auxiliary aids. "Auxiliary aids" include such services or devices as qualified interpreters, assistive listening headsets, television captioning and decoders, telecommunications devices for deaf persons (TDD's), videotext displays, readers, taped texts, Brailled materials, and large print materials.

*…*Public entities are not required to provide auxiliary aids that would result in a fundamental alteration in the nature of a service, program, or activity or in undue financial and administrative burdens. However, public entities must still furnish another auxiliary aid, if available, that does not result in a fundamental alteration or undue burdens.

Most current legal interpretations also apply these requirements to online presentations, and many local governments are working hard to make their Web sites as accessible as possible.
WWW Further clarification of governmental requirements placed upon governmental entities can be found in the Rehabilitation Act section 508, referred to in accessibility circles simply as *section 508*. A summary may be found online at http://www.access-board.gov/sec508/overview.htm.
WWW In short, the proposed updates to section 508 will set specific accessibility standards for governmental Web efforts. At this time, the standards being proposed are not in line with the WAI Content Accessibility Guidelines, but work is in progress to bring the two projects in line, or at least out of major conflict. Watch the news section on the WAI home page for updates as they become available (http://www.w3.org/WAI/).

Additional Online Reading

The regulations that safeguard accessibility for disabled individuals can be complex. If you're designing sites for a public agency of any kind—from a city's informational Web site, to a publicly funded university site—you may be required to provide certain kinds of access that a private enterprise would not. Your organization probably already has an ADA compliance officer or has assigned those tasks to someone as part of a broader job description. Be sure to consult with that person early in the design phase of your Web project.

You'll find additional information at the following Web sites. Remember, interpreting these statutes and requirements is a task for specialists:

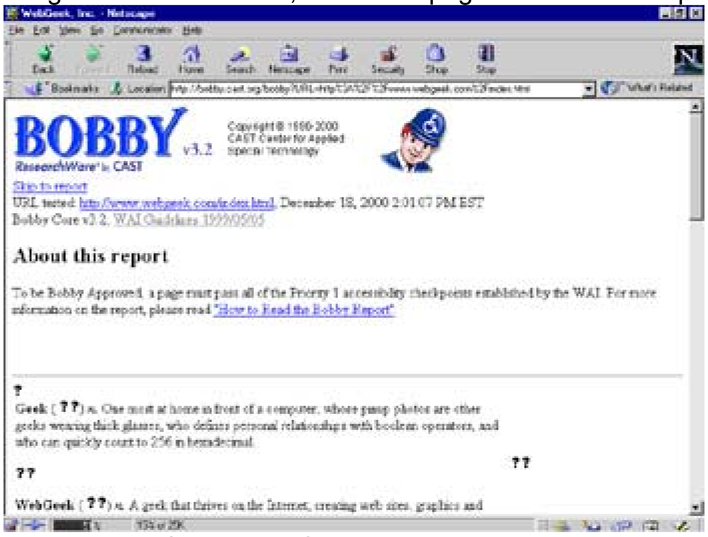
- Technology-Related Assistance for Individuals with Disabilities Act of 1988 as Amended in 1994: http://www.itpolicy.gsa.gov/coca/tech_act.htm
- The Rehabilitation Act Amendments of 1992: http://www.itpolicy.gsa.gov/coca/sect508.htm
- The U.S. Dept. of Education's Requirements for Accessible Software Design (policyversion 1.2, May 31, 2000): http://gcs.ed.gov/coninfo/clibrary/software.htm

Center for Applied Special Technology: Bobby
The Center for Applied Special Technology (CAST) was founded in 1984 as a nonprofit organization with the mission to expand opportunities for people for disabilities through the innovative use of

computer technology. Guidelines were developed for accessibility in Web pages by the Trace Research and Development Center and were quickly adapted by CAST in 1996 as the basis of a tool for Web designers to review their work for any potential problems. The tool was named *Bobby*, after the friendly British police officer.

WWW Developers can use Bobby online for a quick report or download the program to install on intranets or for use behind firewalls. The downloaded version is suitable for situations where large numbers of documents need to be checked to produce a comprehensive report about a full site. The online version of Bobby can be reached at `http://www.cast.org/bobby/`.

A simple report from Bobby can be generated using any Web page that's online. Running the program against the WebGeek, Inc. home page results in the report in Figure 20.1.



BOBBY v3.2 ResearchWare™ by CAST

Copyright © 1996-2000 CAST Center for Applied Special Technology

Skip to report

URL tested: http://www.webgeek.com/index.html, December 18, 2000 2:01:07 PM EST

Bobby Core v3.2, WAI Guidelines 1999/05/05

About this report

To be Bobby Approved, a page must pass all of the Priority 1 accessibility checkpoints established by the WAI. For more information on the report, please read "How to Read the Bobby Report"

?

Geek (??) n. One most at home in front of a computer, whose pinup photos are other geeks wearing thick glasses, who defines personal relationships with boolean operators, and who can quickly count to 256 in hexadecimal.

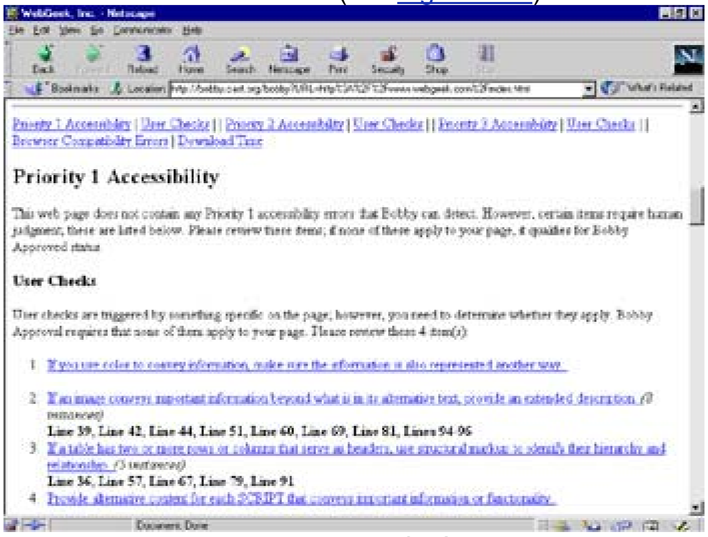
??

??

WebGeek (??) n. A geek that thrives on the Internet, creating web sites, graphics and

Figure 20.1: Checking a file using the Bobby utility

At first glance, the question marks may make it seem as if there are many problems. But as you read further into the report, you'll find that the site does meet Level 1 of the Web Content Accessibility Guidelines from the W3C (see Figure 20.2).



Priority 1 Accessibility | User Checks || Priority 2 Accessibility | User Checks || Priority 3 Accessibility | User Checks || Browser Compatibility Errors | Download Time

Priority 1 Accessibility

This web page does not contain any Priority 1 accessibility errors that Bobby can detect. However, certain items require human judgment; these are listed below. Please review these items; if none of these apply to your page, it qualifies for Bobby Approved status.

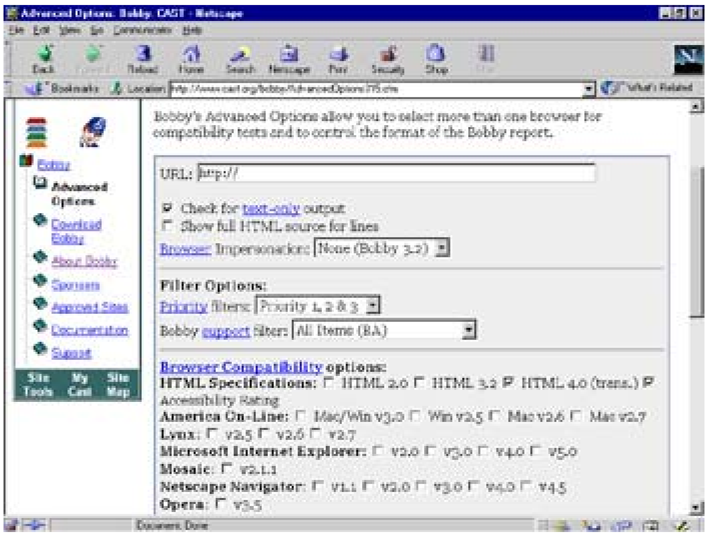
User Checks

User checks are triggered by something specific on the page; however, you need to determine whether they apply. Bobby Approval requires that none of them apply to your page. Please review these 4 item(s):

1. If you use color to convey information, make sure the information is also represented another way.
2. If an image conveys important information beyond what is in its alternative text, provide an extended description. (8 instances)
   Line 39, Line 42, Line 44, Line 51, Line 60, Line 69, Line 81, Lines 94-96
3. If a table has two or more rows or columns that serve as headers, use structural markup to identify their hierarchy and relationships. (5 instances)
   Line 36, Line 57, Line 67, Line 79, Line 91
4. Provide alternative content for each SCRIPT that conveys important information or functionality.

Figure 20.2: The page passes WCAG Level 1.

You can have more advanced control over compatibility testing by choosing the Advanced Options link (Figure 20.3). While Bobby does not yet provide direct support for XHTML, setting it to HTML 4 will bring reliable results, provided you also review your page for the differences between HTML 4 and XHTML 1.



Bobby's Advanced Options allow you to select more than one browser for compatibility tests and to control the format of the Bobby report.

URL: http://

☑ Check for text-only output
☐ Show full HTML source for lines
Browser Impersonation: None (Bobby 3.2)

Filter Options:
Priority filters: Priority 1, 2 & 3
Bobby support filter: All Items (BA)

Browser Compatibility options:
HTML Specifications: ☐ HTML 2.0 ☐ HTML 3.2 ☑ HTML 4.0 (trans.) ☑ Accessibility Rating
America On-Line: ☐ Mac/Win v3.0 ☐ Win v2.5 ☐ Mac v2.6 ☐ Mac v2.7
Lynx: ☐ v2.5 ☐ v2.6 ☐ v2.7
Microsoft Internet Explorer: ☐ v2.0 ☐ v3.0 ☐ v4.0 ☐ v5.0
Mosaic: ☐ v2.1.1
Netscape Navigator: ☐ v1.1 ☐ v2.0 ☐ v3.0 ☐ v4.0 ☐ v4.5
Opera: ☐ v3.5

Figure 20.3: Advanced options available using Bobby

For those sites that pass muster, you can add a Bobby Approved! icon to your page, as shown here.

Techniques for Accessible Design
If you've been working your way through this book, you have probably already incorporated many accessibility features in your design. It's no accident that *effective* Web design is frequently *accessible* Web design. Although in previous chapters I've briefly mentioned features that improve accessibility, I'm restating them here to reinforce the need to incorporate these techniques in your XHTML repertoire and to provide a concise and consolidated reference that you can use as a checklist when evaluating existing designs. In fact, these 10 points are found on a reference card, known as "Quick Tips," published by the W3C WAI program.

Quick Tips for Web Accessibility

Each of these items can be verified by a quick review of your Web pages. If you've been following the examples given in previous chapters, you'll find that you meet nearly all of them without any further editing!

Images and Animation

Tip Use the `alt` attribute to describe the function of each visual.

The `alt` attribute must be used for each `img` element and should be used for any other visual object included in a Web page. These items may be animations, applets, or video clips. Anywhere the `alt` attribute is permitted, accessibility is increased when it is used.

Image Maps

Tip Use client-side `map` and text for hotspots.

By processing the user action on the user's machine, client-side image maps are more accessible. They allow `alt` attributes to be added to area elements giving explicit linking information and the map to be wholly identified with its own `alt` attribute.

Multimedia

Tip Provide captioning and transcripts of audio and descriptions of video.

Audio segments are one of the easiest to make accessible; simply have them transcribed into text that can be linked near where the original audio files are linked. When making such transcripts, individual speakers should be clearly identified, as should a change in speaker.

Tip When creating transcripts of audio events, don't give in to the temptation to edit out such "extras" as sound effects or "off-camera" responses. In order for readers to experience the event as everyone else did, they need to be presented with the same information. If, for instance, a speaker's comments are received with a chuckle from the audience, a notation in the transcript similar to "<audience laughs>" can reinforce the effect and tone of the content with little effort on your part.

Video can be a little more difficult. Transcripts can be made of the audio portion of the video, but even that doesn't necessarily give the complete picture. Consider what it would be like to only listen to a popular movie or television show. A good portion of the information would be missed. Therefore, if you transcribe a video presentation, be sure to include in as much detail as possible segments describing what is happening on screen.

Hypertext Links

Tip Use text that makes sense when read out of context. For example, avoid *click here*.

The *click here* phenomenon is rather easy to understand. Most people access the Web using a visual screen and a mouse and make a clicking action with that mouse to follow a link. But consider that instruction when presented to a blind user. Where exactly is *here*? What if someone with a physical disability doesn't have the dexterity to use a mouse and therefore uses only keyboard shortcuts to activate links or perhaps uses a sip-and-puff tool, common among those with paralysis in the hands and arms? They don't "click," do they?

Instead, when linking, choose a meaningful yet small portion of your text which will stand up when taken out of context. For instance, where would you place the link to Quick Tips card ordering information in this next sentence?

The W3C's Web Accessibility Initiative has made a Quick Tips card

available for use by designers.
The most accessible answer is the direct one, around the words *Quick Tips card*.

Page Orientation

Tip Use headings, lists, and consistent structure for page organization. Use CSS for layout and style where possible.

On the surface, this tip may be rather vague. However, its meaning has been discussed throughout the earlier chapters of this book. Designers should use XHTML elements for their intended purpose: providing structure to documents. They should *not* be used for coincidental byproduct effects such as indentation, italic rendering, or other visual effects. The visual portion of the page display should be accomplished through CSS whenever possible.

Graphs and Charts

Tip Summarize or use the `longdesc` attribute.

Graphs and charts are two forms of images that contain significant information not practical to store in the `alt` attribute. A second attribute, `longdesc`, is available for images just for this purpose. The value of this attribute is a URI, leading to a page that contains the full description. Consider the following:

```
<img src="budget.gif" alt="2001 budget proposal"
  longdesc="budget.html" />
```

In the `budget.html` file, a full text-based reading of the chart would be given, with supporting numbers, description, and any other required data.

Scripts, Applets, and Plug-ins

Tip Provide alternative content in case active features are inaccessible or unsupported.

A presentation should never rely solely on a technology that may not be accessible. Such technology includes Shockwave and Flash animations, Java Applets, and other effects requiring special browser plug-ins (360-degree-view pictures, for instance). The solution is similar to other techniques described so far. Captioning, transcripts, or, in the case of content presented using the `object` element, fallback content (that is, content presented when a visitor's browser can't handle a particular technology) can be included in traditional XHTML presentation (headings, text, lists, etc.) within the `object` element.

Frames

Tip Use `noframes` and meaningful titles.

This tip is pretty self-explanatory. If you use frames, always use a `noframes` element to provide the same content in a nonframed environment. Doing this well can result in extra work, which is a downside to frames (rather than a downside to providing the `noframes` alternative).
The `frame` element can take a `title` attribute. Be sure to use this attribute on each frame, making the title meaningful, as you would when working with links.

Tables

Tip Make line-by-line reading sensible. Summarize.

Some renderings of tables on alternative devices, such as speech readers or Braille output, will render text across tables by reading the first line of each column across the page, rather than reading fully within a column before moving on to the next. You can help avoid this situation by avoiding columnar layout for text. Although it may be fun to try and mimic a newspaper-style layout on the Web, it's not a practical method of presentation. That's not only for accessibility reasons—most people aren't used to reading in that fashion on a computer screen. Such layouts are confusing.
You can enhance tables by using the `summary` attribute with the `table` element. The attribute should contain a short, concise description of the table's contents. The description goes beyond the title-type information provided in the `caption` element, but it doesn't account for every detail in a large, data-intensive table, either.

Check Your Work

Tip **WWW** Validate. Use tools, the checklist, and guidelines at http://www.w3.org/TR/WCAG.

Chapter 11 of this book is dedicated to the process of validation, and you should be quite familiar with it by now. A valid site, by definition, has a higher success rate for being displayed as intended on any browser.

The WAI Web site has significant additional suggestions for design techniques and checklists—far more than could be covered in a single chapter in this book. Visit the URL indicated for more information.

Additional Accessibility Techniques

The WAI Quick Tips cover most of the issues that should be addressed in every document. However, I find a few more important enough to discuss here.

Insert Some New Assistance Methods for Tables

As mentioned earlier in the discussion of the WAI Quick Tips, tabular formatting can really confuse speech-readers and browsers that aren't table compliant. Although bypassing the problem is difficult without providing an alternative nontables page, you can provide some assistance by always inserting a line-break tag at the end of each table cell, such as:

<td>*Cell Contents*<br /></td>

For browser clients that don't fully render tables but do display the contents of the cells, the code will insert a line break between each cell's contents, which at least can bring some clarity.
HTML 4 included some valuable tools to assist speech readers that are compliant with the HTML 4 standard, and XHTML incorporates them as well. Several new attributes for table elements were defined, as outlined in Table 20.1 and put into action in the code that follows.

Table 20.1: New Attributes for Table Elements

ATTRIBUTE	ELEMENT	USAGE
`summary`	`table`	Summarizes the table contents.
`id`	`th`	Sets an identifier for each table heading. Similar to the name attribute in forms.
`headers`	`td`	Identifies with which heading the cell

Table 20.1: New Attributes for Table Elements

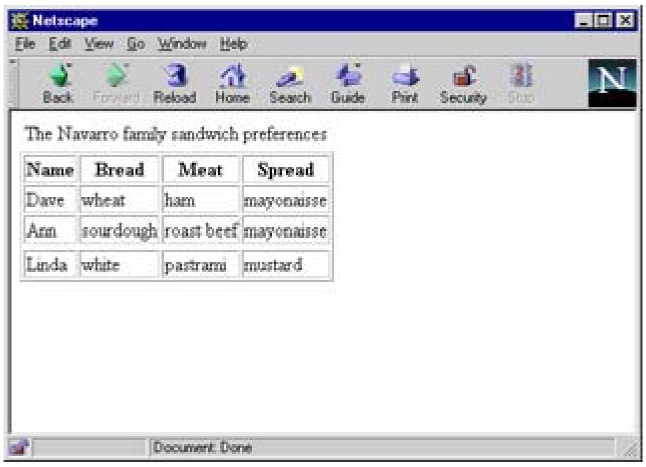
ATTRIBUTE	ELEMENT	USAGE
		should be logically aligned.

```
<table border="1" summary="This table charts what kind of
  sandwiches the Navarro family prefers, including bread,
  meat and spread choices.">
<caption>The Navarro family sandwich preferences</caption>
<tr>
<th id="t1">Name</th>
<th id="t2">Bread</th>
<th id="t3" abbr="Type">Meat</th>
<th id="t4">Spread</th>
</tr>
<tr>
<td headers="t1">Dave</td>
<td headers="t2">wheat</td>
<td headers="t3">ham</td>
<td headers="t4">mayonaisse</td>
</tr>
<tr>
<td headers="t1">Ann</td>
<td headers="t2">sourdough</td>
<td headers="t3">roast beef</td>
<td headers="t4">mayonaisse</td>
<tr>
</tr>
<td headers="t1">Linda</td>
<td headers="t2">white</td>
<td headers="t3">pastrami</td>
<td headers="t4">mustard</td>
</tr>
</table>
```

The `summary` attribute describes the contents of the entire table. Consequently, it's used within the `table` element.

The id attribute functions much like the name attribute in forms (see Chapter 8 to refresh your memory). When each heading is defined in the first table row, it's assigned a unique identifier—for purposes here, `t1` through `t4`.

When filling the main table data cells, each one is assigned to a heading with the `headers` attribute. For instance, the cell that contains the name `Dave` is assigned to the heading `Name`, which was given the ID of `t1`. Therefore, the value of the `headers` attribute for that cell becomes `t1`. Figure 20.4 shows the file displayed in Navigator 4.76 on Windows 98.


Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Document: Done

The Navarro family sandwich preferences

Name	Bread	Meat	Spread
Dave	wheat	ham	mayonaisse
Ann	sourdough	roast beef	mayonaisse
Linda	white	pastrami	mustard

Figure 20.4: Tables with accessibility features appear normal in visual browsers.

Doesn't look much different from the other examples, does it? The key here is what a speech reader can do with the additional attributes. Like graphical browsers, speech readers aren't required to read (display) these features in one strict manner; the programmers who write them do have some leeway. However, the intended presentation would be similar to what I've presented in Table 20.2.

Table 20.2: "Display" from a Speech Reader

TABLE CONTENTS			
Caption: The Navarro family sandwich preferences			
Summary: This table charts what kind of sandwiches the Navarro family prefers, including bread, meat, and spread choices.			
Name: Dave	Bread: wheat	Meat: ham	Spread: mayonnaise
Name: Ann	Bread: sourdough	Meat: roast beef	Spread: mayonnaise
Name: Linda	Bread: white	Meat: pastrami	Spread: mustard

This presentation lessens the chance of confusion by allowing the listener to immediately associate each cell's contents with the proper category.

Tip Additional attributes for more complex situations, along with sample tables, can be found on the W3C Web site, at http://www.w3.org/TR/PR-html40/struct/tables.html#h-11.4. (Though the page deals with HTML specifications, XHTML 1 refers to it, rather than having the specifications written over again.)

Provide a Substitute for Online Forms

Provide a link to a text-based form—one that can be printed and filled out by hand and mailed to you, rather than submitted online. If doing so isn't practical, perhaps for forms that don't support ordering or important consumer feedback issues, at least provide a telephone number for voice contact and an e-mail link for additional online communication.

Tip Don't rely on your XHTML forms for printing visitor responses. In most cases, browsers don't pass along the contents of the input fields when sending the data to the printer. Your visitors will wind up with blank spaces where their responses were. A better solution is to provide a form with lines where you want the user to make an entry, just like traditional printed forms.

Document Formats Other Than XHTML

Providing important content in formats other than XHTML presents challenges to the general Web population, not just to those with disabilities. For example, if you provide a link for visitors to download a white paper on your latest product, but you upload the file in Microsoft Word 2000 format, only a limited number of people will be able to open and read it. By providing an ASCII-text alternative copy, users who don't have the required program can nevertheless open the file. Though formatting and other special details may be lost, critical information will still be delivered.

Consider the ASCII-text approach for documents formatted in word-processor formats, spreadsheets, Adobe PDF files, Quark or Publisher files, and any other format that can't be displayed directly in a Web browser without a special plug-in.

Enhancing Readability

Several design choices can decrease the readability of your site for visitors that aren't generally classified as disabled. Many adults over the age of 40 experience some difficulty in reading small text or text that's set against a background that has little contrast compared with the text color.

Choose High-Contrast Background and Text Colors

In Chapter 16, you learned about color theory—how to choose complimentary colors and contrasting colors and how to match hues and densities. You can enhance readability by choosing high-contrast colors for your background and text. Not only does the contrast serve visitors with poor but not completely failed vision, but it also helps those who suffer from color blindness (it's estimated that about a third of the male population is color blind).

Warning Did you know that light text against a dark background might render your pages unprintable? With many browser/printer combinations, the background color is removed from the print data, yet the text color is interpreted in its appropriate gray-scale classification. The result is white or nearly white text printed on white paper—in other words, a blank page!

Use Relative Font Sizing

Web designers have more font-manipulation options available than ever before, with the features in Cascading Style Sheets and the font element in XHTML. When taking advantage of these styles, choose relative font sizing over absolute font sizing. I addressed this concept in both Chapters 7 and 15; it's an important issue, besides for accessibility for the disabled. Predicting what font sizes will be functional for your visitors is next to impossible with the variety of monitors and supported screen resolutions available.

Style Sheets

Rather than being an impediment to accessibility, style sheets can actually help improve the odds that all your audience will be able to view your site as you intended. In Chapter 7, I discussed how to provide links to alternative style sheets. By supplying sheets that do away with elements that can complicate the experience, visitors with CSS-compliant browsers can choose the display that will serve them best.

Summary

Accessibility is as much a design philosophy as it is an accommodation to those with disabilities. By following easy-to-remember methods, your sites can be experienced by as many individuals as possible.

Part III: Moving Forward with XHTML

Chapter List

Chapter 21: XHTML 1.1 and XHTML Modularization

Overview

When the W3C released the first working drafts of XHTML 1.1, many developers looked at the details and scratched their heads, wondering where the differences were from XHTML 1. It looked remarkably similar to XHTML 1 Strict. On the surface, it is nearly identical; what's changed, however, is strictly under the hood. XHTML 1.1 was created using an entirely new technique for building DTDs—Modularization.

This chapter covers the following topics:

- What has changed between XHTML 1 and 1.1
- How Modularization provides a framework for XHTML
- How to define abstract modules

XHTML 1.1: Back to the Structure

In the beginning of this book, I talked about the desire to return to the structural roots of HTML by reformulating it into XML syntax. XHTML was designed as the bridge between the two worlds. With XHTML 1.1, a little more than half of that bridge has been crossed by the removal of presentational elements and attributes from XHTML when the same results can be achieved through other means (such as the use of CSS or XSL style sheets).

As this book went to print, XHTML 1.1 had completed a Last Call review of the Working Draft, meaning the HTML Working Group felt it was ready to go to the next step: Candidate Recommendation or Proposed Recommendation. The transition to the higher status was expected to occur imminently.

Tip You'll always find the latest version of XHTML 1.1 by visiting this stable URL: http://www.w3.org/TR/xhtml11/.

The first question about XHTML 1.1 that comes to most people's minds is "Why did it follow so soon on the heels of XHTML 1?" After all, XHTML 1.1 was in the early Working Draft stage when XHTML 1 was made a Recommendation just before Christmas 1999. Why wouldn't the Working Group simply put what they wanted into XHTML 1, rather than save it for XHTML 1.1?

The answer goes back to the bridge metaphor. The first foray over the bridge took HTML 4.01 into XML syntax, which resulted in XHTML 1. Though the language was in XHTML syntax, it still existed in three variants: XHTML 1 Strict, XHTML 1 Transitional, and XHTML 1 Frameset. The object behind producing XHTML 1.1 was to narrow those three down to a single, uniform dialect.
For most users, the XHTML 1.1 DTD is functionally identical to XHTML 1 Strict. However, some minor changes exist that are outlined in Table 21.1. If an element wasn't impacted, it is excluded from this table.

Table 21.1: Minor Changes from XHTML 1 Strict to XHTML 1.1

ELEMENT	CHANGES FROM XHTML 1 STRICT
All elements	lang attribute removed
a	accesskey, name, tabindex, target attributes removed
area	target attribute removed
base	element removed entirely
basefont	element removed entirely
body	background, bgcolor, text, link, vlink, alink attributes removed
br	clear attribute removed
caption	align attribute removed
center	element removed entirely
div	align attribute removed
font	element removed entirely
frame	element removed entirely
frameset	element removed entirely
h1 through h6	align attribute removed from each

Table 21.1: Minor Changes from XHTML 1 Strict to XHTML 1.1

ELEMENT	CHANGES FROM XHTML 1 STRICT
hr	align, noshade, size, and width attributes removed
iframe	element removed entirely
img	align, border, hspace, vspace attributes removed
input	align attribute removed
isindex	element removed entirely
legend	align attribute removed
li	type and value attributes removed
link	target attribute removed
map	name attribute removed
menu	element removed entirely
noframes	element removed entirely
ol	compact, start, and type attributes removed
p	align attribute removed
pre	width attribute removed
script	language attribute removed
strike	element removed entirely
table	align, bgcolor attributes removed
td	bgcolor, height, nowrap, width attributes removed
th	bgcolor, height, nowrap, width attributes removed
tr	bgcolor attribute removed
u	element removed entirely
ul	compact, type attributes removed

After reviewing Table 21.1, you should appreciate that essentially all presentational attributes and elements have been removed from XHTML. Designers must use style sheets—whether CSS, XSL, or any other style language that may develop—in order to bind presentation to the structural elements that remain.

Most designers have at least some exposure to CSS. If you skipped over Chapter 7, "Style Sheets," you might want to set aside some time to review it after reading this chapter. The techniques learned there will be essential to your success in working with XHTML 1.1.

I'll review a file created earlier in Chapter 5, columngroup.html, and see how it needs to be changed to go from XHTML 1 Transitional to XHTML 1.1. The file is shown in Listing 21.1 and is on the CD-ROM.

Listing 21.1: columngroup.html

```
<!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" xml:lang="en"

  lang="en">

<head>

<title>Column Grouping</title>

</head>

<body>

<div align="center">
```

```
<table border="1" >

<caption>My Pets</caption>

<colgroup span="1">

<colgroup span="2" bgcolor="yellow">

<tr>

<td><strong>Name</strong></td>

<td>Dizzy</td>

<td>Bubba</td>

</tr>

<tr>

<td><strong>Type</strong></td>

<td>Cat</td>

<td>Dog</td>

</tr>

<tr>

<td><strong>Breed</strong></td>

<td>Grey Tabby</td>

<td>Pembroke Welsh Corgi</td>

</tr>

<tr>

<td><strong>Traits</td>

<td>quiet, loves to lay on paper</td>

<td>hyperactive, attention span of a gnat, loves to play

  fetch with his Koosh ball, drinks water out of the

  swimming pool</td>

</table>

</div>

</body>

</html>
```

Referring back to Table 21.1, you should be able to identify two attributes that are no longer allowed in XHTML 1.1: the align attribute on div, and the border attribute on table. The colgroup element (not listed in the table) also has an attribute left over from XHTML 1 Transitional that you should have also been able to pick out: bgcolor. Each of these properties will need to be placed into a style sheet. Since I'm only working with three attributes, I'll use the style element inside the head element rather than using an external style sheet:

```
<style type="text/css">
table {border-width:1px}
div {text-align:center}
.yellow {bgcolor:yellow}
</style>
```

Notice that the third style declaration is a class named *yellow*. This was created because I have more than one instance of the colgroup element, yet only one of them needs a background color change. By adding a class attribute to that element, the style property will be invoked on the one colgroup element I want to modify.

The new XHTML 1.1-compliant file (also on the CD-ROM) now looks like Listing 21.2.

Listing 21.2: columngroup.html, compliant with XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xml:lang="en"

  lang="en">

<head>

<title>Column Grouping</title>

<style type="text/css">

table {border-width:1px}

div {text-align:center}

.yellow {bgcolor:yellow}

</style>

</head>

<body>

<div>

<table>

<caption>My Pets</caption>

<colgroup span="1">

<colgroup span="2" class="yellow">

<tr>

<td><strong>Name</strong></td>

<td>Dizzy</td>

<td>Bubba</td>
```

```
</tr>

<tr>

<td><strong>Type</strong></td>

<td>Cat</td>

<td>Dog</td>

</tr>

<tr>

<td><strong>Breed</strong></td>

<td>Grey Tabby</td>

<td>Pembroke Welsh Corgi</td>

</tr>

<tr>

<td><strong>Traits</td>

<td>quiet, loves to lay on paper</td>

<td>hyperactive, attention span of a gnat, loves to play

  fetch with his Koosh ball, drinks water out of the

  swimming pool</td>

</table>

</div>

</body>

</html>
```

XHTML Modularization: The Framework that Supports XHTML 1.1

XHTML 1.1 is the first full version of XHTML that was built using the technique known as XHTML Modularization. The goal, you'll remember, in crossing the bridge from HTML to XML is to allow document authors to create their own elements and attributes yet still retain the comfortable framework of HTML's headings, paragraphs, tables, and other basic document structures.
It would have been fairly easy for the W3C to provide a method for authors to add a tag or two to, say, the full version of XHTML 1 Transitional, but what if the author didn't want to support everything that one of the standardized XHTML versions required? For instance, cell phones are hard-pressed to present tabular data, (at least in the manner in which you're used to seeing it), and they certainly would have trouble with frames. A better option is a building-block approach. The W3C grouped elements with similar semantics or functionality together and allowed authors to build XHTML documents based on only the blocks they choose plus their own custom blocks. Figure 21.1 illustrates the concept.

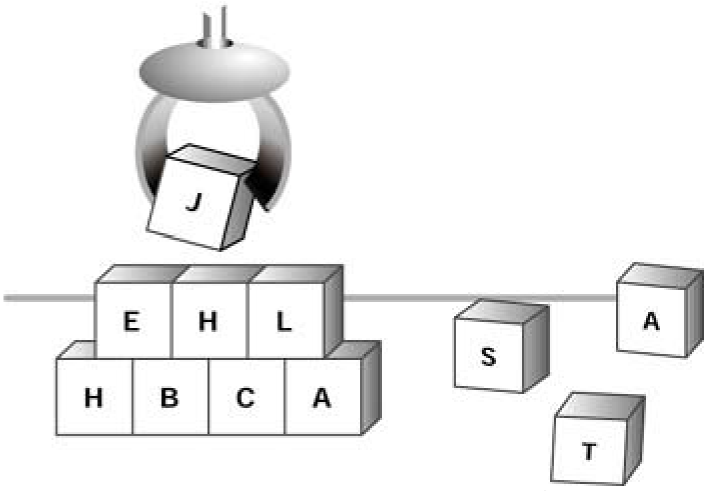


Figure 21.1: An XHTML document built out of logical blocks of elements

The W3C chose this building-block route and called each group of elements a *module*. When modules are combined into a single DTD, using a file known as a *DTD driver* that imports and refers to each individual module, the result is a *modularized* DTD.

To begin building DTDs with modules, the individual modules must first be defined. Module definition is a two-part process: First the elements and/or attributes are grouped together in an *abstract collection*, and then a DTD segment is written to define each element in the collection, creating the module.

Abstract Collections

An abstract collection is simply a grouping of elements that belong together based on semantics, function, or any other reason the document author may choose. The abstract collection should be thought of more as a description than as a concrete item. You're familiar with one abstract collection: the English alphabet. It has 26 objects in the collection, each a different letter. Any processor that deals with the English language, including humans, will have predefined knowledge of this collection, either through hard coding by programmers or through the memorization and learning that people do.

XHTML-aware processors, such as today's Web browsers, have hard-coded knowledge of the W3C-defined modules. For instance, IE 5.5 knows what unordered, ordered, and definition lists are. The abstract module definition is the human-readable form of the module. The list module is presented here in Table 21.2.

Table 21.2: The XHTML List Module

ELEMENTS	ATTRIBUTES	MINIMAL CONTENT MODEL
`dl`	Common	(dt | dd)+
`dt`	Common	(PCDATA | Inline)*
`dd`	Common	(PCDATA | Inline)*
`ol`	Common	li+
`ul`	Common	li+
`li`	Common	(PCDATA | Inline)*

The minimal-content-model column in the abstract definition contains XML syntax for the content that each element may contain. I'll cover that syntax in further detail in Chapter 22.

Rather than this easy-to-read form of the module, the XHTML processor knows the DTD for these elements. The DTD implementation of this module appears in Listing 21.3 and is also on the CD-ROM.

Listing 21.3: XHTML list module DTD implementation

```
<!-- ...................................................... -->
```

```
        <!-- XHTML Lists Module
.................................................. -->
        <!-- file: xhtml-list-1.mod

     This is XHTML, a reformulation of HTML as a modular
       XML application.
     Copyright 1998-2000 W3C (MIT, INRIA, Keio), All
       Rights Reserved.
     Revision: $Id: dtd_module_defs.html,v 1.1
       2000/10/20 16:45:13 dom Exp $ SMI

     This DTD module is identified by the PUBLIC and
       SYSTEM identifiers:

       PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
       SYSTEM "http://www.w3.org/TR/xhtml-modulatization/
       DTD/xhtml-list-1.mod"

       Revisions:
       (none)

.......................................................... -->

         <!-- Lists

        dl, dt, dd, ol, ul, li

       This module declares the list-oriented element types
         and their attributes.
          -->
```

```
<!ENTITY % dl.qname "dl" >

<!ENTITY % dt.qname "dt" >

<!ENTITY % dd.qname "dd" >

<!ENTITY % ol.qname "ol" >

<!ENTITY % ul.qname "ul" >

<!ENTITY % li.qname "li" >


<!-- dl: Definition List ............................... -->


<!ENTITY % dl.element "INCLUDE" >

<![%dl.element;[

<!ENTITY % dl.content "( %dt.qname; | %dd.qname; )+" >

<!ELEMENT %dl.qname; %dl.content; >

<!-- end of dl.element -->]]>


<!ENTITY % dl.attlist "INCLUDE" >

<![%dl.attlist;[

<!ATTLIST %dl.qname;

    %Common.attrib;

>

<!-- end of dl.attlist -->]]>


<!-- dt: Definition Term .............................. -->


<!ENTITY % dt.element "INCLUDE" >

<![%dt.element;[


<!ENTITY % dt.content

    "( #PCDATA | %Inline.mix; )*"

>

<!ELEMENT %dt.qname; %dt.content; >
```

```
<!-- end of dt.element -->]]>

<!ENTITY % dt.attlist "INCLUDE" >
<![%dt.attlist;[
<!ATTLIST %dt.qname;
    %Common.attrib;
>
<!-- end of dt.attlist -->]]>

<!-- dd: Definition Description ........................ -->

<!ENTITY % dd.element "INCLUDE" >
<![%dd.element;[
<!ENTITY % dd.content
   "( #PCDATA | %Flow.mix; )*"
>
<!ELEMENT %dd.qname; %dd.content; >
<!-- end of dd.element -->]]>

<!ENTITY % dd.attlist "INCLUDE" >
<![%dd.attlist;[
<!ATTLIST %dd.qname;
    %Common.attrib;
>
<!-- end of dd.attlist -->]]>

<!-- ol: Ordered List (numbered styles) ............ -->

<!ENTITY % ol.element "INCLUDE" >
<![%ol.element;[
<!ENTITY % ol.content "( %li.qname; )+" >
```

```
<!ELEMENT %ol.qname; %ol.content; >
<!-- end of ol.element -->]]>

<!ENTITY % ol.attlist "INCLUDE" >
<![%ol.attlist;[

<!ATTLIST %ol.qname;
    %Common.attrib;
>
<!-- end of ol.attlist -->]]>

<!-- ul: Unordered List (bullet styles) ........... -->

<!ENTITY % ul.element "INCLUDE' >
<![%ul.element;[
<!ENTITY % ul.content "( %li.qname; )+" >
<!ELEMENT %ul.qname; %ul.content; >
<!-- end of ul.element -->]]>

<!ENTITY % ul.attlist "INCLUDE" >
<![%ul.attlist;[
<!ATTLIST %ul.qname;
    %Common.attrib;
>
<!-- end of ul.attlist -->]]>

<!-- li: List Item ............................... -->

<!ENTITY % li.element "INCLUDE' >
<![%li.element;[
<!ENTITY % li.content
```

```
    "( #PCDATA | %Flow.mix; )*"

>

<!ELEMENT %li.qname; %li.content; >

<!-- end of li.element -->]]>


<!ENTITY % li.attlist "INCLUDE" >

<![%li.attlist;[

<!ATTLIST %li.qname;

    %Common.attrib;

>

<!-- end of li.attlist -->]]>


<!-- end of xhtml-list->1.mod -->
```

This DTD is well commented, so even though you may not yet understand the syntax, you should be able to follow what's happening—that each element is defined, followed by the attributes that are allowed with it. Chapter 22 will get into more detail about the syntax.

The interesting part of Modularization comes when a custom module is introduced into the mix. Obviously, the processor can't have preprogrammed knowledge of how the module is supposed to perform. Therefore, it must learn that from the DTD if it is to produce a valid document.

Summary

In this chapter, I've explored how XHTML 1.1 is built upon a new framework for XHTML DTD design: the Modularization of XHTML. You have reviewed the specific differences between XHTML 1 Strict and XHTML 1.1, a module-based version of XHTML. Finally, the idea of abstract collections of elements allows the Modularization technique to work, teaching XHTML processors about new elements and attributes as they need to learn them, by supplying DTD fragments that can be plugged into a larger DTD in a building-block fashion.

Chapter 22: Creating Your Own XHTML Module

Overview

In order to create your own XHTML elements and attributes (as opposed to using new elements and attributes created by others), you're going to need to roll up your sleeves and get your hands a little dirty working with Document Type Definitions (DTDs). Don't worry, though; like gardening, with a little work, handling DTDs will begin to grow on you!

This chapter covers the following topics:

- Reading a DTD
- Defining elements and attributes
- Combining your module with others
- Using the resulting DTD

How to Read a DTD

HTML, XML, and now XHTML document type definitions all use an expression syntax developed originally by a pair of scientists named Backus and Naur back in the Stone Age of computing (1960). Over the years, the syntax has been refined and enlarged, and the current framework is now known as *Extended Backus-Naur Form*, or EBNF.

The formal syntax for each expression takes this form:

symbol ::= expression

The `symbol` holds the value of the expression. When reading these forms, the characters `::=` should be articulated as "is defined as." For instance, you should read the following form as "Seasons is defined as winter, spring, summer, and fall":

seasons ::= [winter spring summer fall]

Notice that each word is only separated by a space, not a comma. So each of the items is a part of the symbol being defined.

You won't want all symbols to be defined as an entire list inclusively; sometimes you'll want to choose from a group of possible values. To do so, the list within the expression should be divided by the bar character (`|`).

Tip The bar character is normally found on the same key as the backslash character, immediately above the Enter key.

For instance, a student's course grade could be defined using the following form:

grade ::= (A | B | C | D | F)

The form would then be read as "Grade is defined as A, B, C, D, or F."

Content Models and Occurrence Indicators

The next segment of syntax you'll need to become familiar with are the expressions used in defining elements. For instance, the paragraph element isn't just a set of `<p></p>` tags, it also includes the content presented in between the tags. What's allowed inside those tags is referred to as the *content model*.

The simplest content model is no content model at all; that is, the element doesn't contain any content. In XHTML, I called those *empty elements*, which is exactly how the content model is described. Consider the declaration for the horizontal-rule element `hr`:

<!ELEMENT hr EMPTY>

The expression is labeled as an element declaration by the opening `!ELEMENT` string. Next is the element name (`hr`) and then the content model (`EMPTY`). Since hr doesn't have any content, the `EMPTY` keyword is used to represent that state.

A second content model that requires only the use of a keyword is one where PCDATA, or *parsed character data*, can be found within the element. Parsed character data means that the browser has resolved any character entities, such as ampersands `(&amp;)` as discussed in Chapter 4, into the characters they represent. One element that follows this model is the `textarea` element used within forms. Its element declaration appears as the following:

<!ELEMENT textarea (#PCDATA)>

When the content model contains something more than just text, more complex notation is needed. A common occurrence happens when an element may contain one or more instances of a second element. The list elements `ol` and `ul` behave in this manner, containing one or more instances of `li`. The element declaration for `ol` reads as:

```
<!ELEMENT ol (li)+>
```

The + symbol after the content model is the *occurrence indicator*, which tells you how many times, if at all, the element(s) in the content model must appear. The `+` indicator requires one or more instances of `li` to occur within the `ol` element. Table 22.1 describes the three most common occurrence indicators.

Table 22.1: The Three Most Common Occurrence Indicators

OCCURENCE INDICATOR	USAGE
`?`	Element is optional (zero or one occurrence, but not more than one)
`*`	Element is optional, but it also may be repeated (zero, one, or more occurrences)
`+`	Element is required and may be repeated (one or more occurrences)

In some element content models, other elements are not only required to be present, but they must also be present in a defined order. This type of content-model declaration is known as a *sequence*. The declaration for the `html` element shows a simple sequence:

```
<!ELEMENT html (head, body)>
```

By delimiting the list of elements with a comma, it becomes a sequence as opposed to an option (as previously described using the | delimiter). So a head element and then a body element must appear within the `html` element. You should remember these basic requirements from when you wrote your first XHTML document back in Chapter 3.

Finally, sequences, options, and occurrence indicators can all be blended within a single content-model declaration. One of the declarations that most clearly demonstrates this is the `table` element declaration in XHTML 1 Transitional:

```
<!ELEMENT table

(caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
```

It looks a little intimidating, but you can take it apart piece by piece. It reads: A `table` can contain a `caption` element, a `thead`, and a `tfoot`. It may have either zero or more `col` elements or zero or more `colgroup` elements and must contain either one or more `tbody` or one or more `tr` elements.

Attribute List Declarations

As you know from working with XHTML, elements use more than content to provide data to your Web sites. Attributes have an important role in determining size, presentation, and behavior of nearly all XHTML elements in common use. *Attribute list declarations*, or ATTLIST for short, are written in a form similar to element declarations. I'll look at the attribute list declaration for the `title` element:

```
<!ELEMENT title (#PCDATA)>

<!ATTLIST title %i18n;>
```

Like the element declaration, the ATTLIST declaration begins with the string `!ATTLIST` and then names the *element* for which the attributes are being defined. Instead of a content model, what comes next is a list of allowed attributes and their individual content models. The string `%i18n;` seen in the `title ATTLIST` declaration is a *parameter entity*, or PE—a string short for a much more extensive list of data. You'll recognize parameter entities by the `%` character that appears before each one. Parameter entities must be declared in the DTD before they can be used. For ease of use and readability, many DTD designers declare all PEs at the beginning of the DTD to prevent errors of using one before its declaration. Here's the `i18n` declaration:

```
<!ENTITY % i18n
 "lang       %LanguageCode; #IMPLIED


  xml:lang    %LanguageCode; #IMPLIED
  dir        (ltr|rtl)     #IMPLIED"
  >
```

As with element and attribute list declarations before them, entity declarations begin with a string `!ENTITY`, followed by the entity name. Next comes the list of attributes, one per line, enclosed within quotes.
Each line has the attribute name, its potential value(s), and any necessary keywords. The first attribute is `lang`, and its potential values are represented by another parameter entity—`%LanguageCode`. The keyword `#IMPLIED` indicates that the attribute is not required and that the default value given to an attribute will be instantiated if the attribute is not explicitly included.
To understand what values the `lang` attribute may have, I need to look at the declaration for the `%LanguageCode` entity. Declared earlier in the DTD, that entity was defined as:

```
<!ENTITY % LanguageCode "NMTOKEN">
  <!-- a language code, as per [RFC1766] -->
```

The value of Language code is noted as `NMTOKEN`, one of the four major attribute value types found in XHTML. Table 22.2 reviews each of these types.

Table 22.2: The Four Major Attribute Value Types

ATTTIBUTE VALUE TYPE	DEFINITION
String Type	A string of characters
Tokenized Type	An ID, IDREF, or NMTOKEN
Enumerated Type	A list of fixed values
Entities	An entity defined elsewhere (e.g., URI)

The comment that comes after the entity declaration for language code gives the human reader a hint as to what is expected there. A language code has been defined in the document RFC 1766.

RFCs: Comments or Commandments?

The label RFC is an abbreviation for Request for Comments. It may seem strange that by today's Web standards RFC documents are referenced *normatively*—meaning the information presented there is a binding part of a specification. If the documents are requests for comments, then why are they being used as a definitive reference work?

In the early stages of Internet and Web development, people simply published ideas about how various Net functions should work. A system for collecting these ideas was established, and an informal peer-review process was initiated, bringing about the Request for Comments nomenclature.

Today, ideas are evaluated by the same peer-review process, but they retain the RFC label even after they've gained widespread acceptance. Therefore, you'll find normative references to stable RFCs in many W3C Recommendations, including the XHTML family of Recommendations.

Defining Your Own Elements

To best illustrate the process of defining your own elements, I'll go through the process of defining an entire XHTML module. This module will be used to hold information for a college course catalog. The printed version of the catalog traditionally presents the user with the following information:

- Course ID
- Course title
- Units of credit
- Course description
- Prerequisites (listed by course ID and title)

The first step in determining the abstract module definition is to decide which of these bits of information should be stored as elements or as attributes. A good guide for designating attributes is identifying which data are normally short strings, IDs, or NMTOKENs.
In this case, I want to have an enveloping element that I'll call `course`. Within `course`, the course description catches the eye as a candidate for a child element, perhaps with a PCDATA content model or even with other traditional XHTML elements allowed inside.

Prerequisites also look like a good candidate for an element, because each must store two pieces of information: ID and title, both of which could be candidates for attributes.
Course ID, course title, and units of credit look like good candidates for attributes on the root `course` element. The `course` element shouldn't contain anything other than the two child elements I've identified: description and prerequisites. Using this design, a flowchart for the module might look like Figure 22.1.

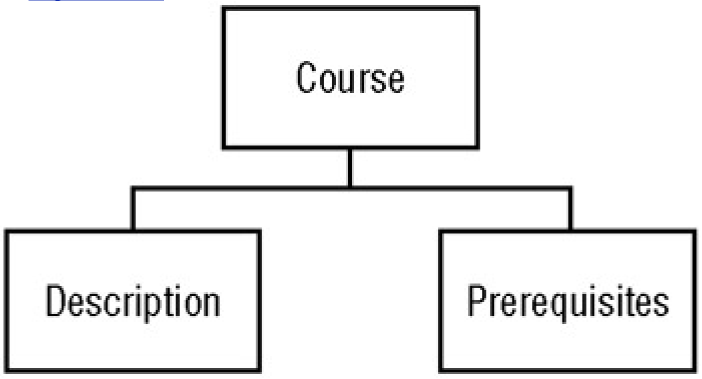


Figure 22.1: A flowchart for the course-catalog module

Before declaring the elements and attribute lists in DTD form, I should first determine the content models of each element and attribute in the abstract module definition. You'll remember from Chapter 21 that the abstract module contains the element names, their allowed attributes, and a minimal content model. Table 22.3 shows a first pass at defining the abstract module.

Table 22.3: An Abstract Module Definition for a Simple Course Catalog Entry

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`course`	`Common, number (NMTOKEN), title (CDATA), units (|"1"|"2"|"3"|"4"|"5")`	`prerequisite*, description`
`prerequisite`	`number (NMTOKEN), title (CDATA), none ("none")`	`EMPTY`
`description`	`Common`	`(PCDATA|Inline|List)+`

The table describes the `course` element as having two attributes. The `number` attribute must comply with the rules for an NMTOKEN, which is a string restricted to letters, numerals, and four punctuation marks—the colon, dash, period, and underscore. The `title` attribute will hold the title of the course and should only contain character data (no markup to be interpreted). Further, the `course` element must contain two elements: one or more instances of `prerequisite` and a `description`.
The `prerequisite` element takes the same attributes as the `course` element. The minimal content model of `EMPTY` indicates that it holds no content, only the information stored in its attributes.
Finally, `description` can take any of the attributes in the Common attribute set, which is composed of attributes available for every element type. (The core attributes of Common are `id`, `role`, and `diff`. The `id` attribute uniquely identifies an element; `role`, declared as NMTOKEN, applies an additional descriptive keyword to an element, on which a style sheet can act. The `diff` attribute describes an element that differs from its use in the last version of the document.) For content, the description can contain PCDATA (text plus items that must be interpreted such as entities) and any members of the List Content set (see Appendix A, "List Module").

In practice, an entry in the catalog might then look like the following:

<course number="SP-1A" title="Beginning Spanish" units="4">

<prerequisite none="none">

<description>For the absolute beginner or students with one

year or less of high-school level Spanish.

<strong>Please note:</strong>This is an immersive

course. No English will be spoken in

```
  class.</description>
</course>
```

Notice that the `description` element contains text and an XHTML `strong` element. The content model of `(PCDATA|Inline|List)` allows this because `strong` is a member of the Inline content set (see Appendix A).

Now that I've developed the abstract module definition and tested the module with a sample case, I can look at the resulting DTD declarations for each element and attribute.

Tip The creation of the DTD module goes a bit beyond the scope of this book and the intention of this chapter. If you want to learn in depth about DTD writing for XHTML Modularization, please refer to *Mastering XHTML*, also from Sybex.

The DTD comes in two pieces. First, a Qnames submodule, which sets up the Qnames, or Qualified Names, is used when XML namespaces are used to differentiate similarly named elements in more than one language. Listing 22.1, also on the CD-ROM, shows the submodule.

Listing 22.1: The CCML qualified-names module

```
<!-- .............................................. -->

<!-- CCML Qualified Names Module .................... -->

<!-- file: catalog-qname-1.mod



   This DTD module is identified by the PUBLIC and SYSTEM

identifiers:



     PUBLIC "-//WEBGEEK//ELEMENTS XHTML CCML Qnames 1.0//EN"

     SYSTEM "http://www.webgeek.com/dtds/catalog/catalog-qname-

1.mod"

     xmlns:CCML="http://www.webgeek.com/xmlns/catalog"



   Revisions:

   (none)

   .............................................. -->



<!ENTITY % NS.prefixed "IGNORE" >

<!ENTITY % CCML.prefixed "%NS.prefixed;" >

<!ENTITY % CCML.xmlns "http://www.webgeek.com/xmlns/catalog" >

<!ENTITY % CCML.prefix "CCML" >

<![ %CCML.prefixed;[

<!ENTITY % CCML.pfx "%CCML.prefix;:" >

<!ENTITY % CCML.xmlns.extra.attrib
```

```
 "xmlns:%CCML.prefix; %URI.datatype;

   #FIXED '%CCML.xmlns;'" >]]>

<!ENTITY % CCML.pfx "" >

<!ENTITY % CCML.xmlns.extra.attrib "" >

<!ENTITY % CCML.course.qname "%CCML.pfx;course" >

<!ENTITY % CCML.prerequisite.qname "%CCML.pfx;prerequisite" >

<!ENTITY % CCML.description.qname "%CCML.pfx;description" >


<!-- end of catalog-qname-1.mod -->
```

Next comes the element and attribute declaration submodule, as shown in Listing 22.2 and on the CD-ROM.

Listing 22.2: The CCML Element declaration module

```
<!-- ................................................ -->

<!-- WebGeek Course Catalog Module ..................... -->

<!-- file: catalog-elements-1.mod


     This DTD module is identified by the PUBLIC and SYSTEM

identifiers:


   PUBLIC "-//WEBGEEK//ELEMENTS XHTML-WebGeek Catalog 1.0//EN"

   SYSTEM "http://www.webgeek.com/dtds/catalog/catalog-1.mod"

   xmlns:CCML="http://www.webgeek.com/xmlns/catalog"


     Revisions:

     (none)

     ................................................ -->


<!-- WebGeek Course Catalog Module


       CCML:course

       CCML:prerequisite

       CCML:description
```

```
This module defines structural components of an educational
  course catalog.
-->
<![ $CCML.prefixed;[
<!ENTITY % CCML.xmlns.attrib
  "NS.prefixed.attrib;"
>
]]>
<!ENTITY % CCML.xmlns.attrib
  "NS.prefixed.attrib;
  xmlns %URI.datatype;   #FIXED      '%CCML.xmlns;'"
>
<!ELEMENT %CCML.course.qname;
  (%CCML.prerequisite.qname;*, %CCML.description.qname;) >

<!ATTLIST %CCML.course.qname;
  %Common;
  number      NMTOKEN     #REQUIRED
  title       CDATA       #REQUIRED
  units       (1|2|3|4|5)  #REQUIRED
  %CCML.ns.attrib;
>


<!ELEMENT %CCML.prerequisite.qname; EMPTY >
<!ATTLIST %CCML.prerequisite.qname;
   %Common;
   number   (NMTOKEN)    #REQUIRED
   title    CDATA        #REQUIRED
   none     ("none")
   %CCML.ns.attrib;
```

```
>


<!ELEMENT %CCML.description.qname;

  (PCDATA | Inline | List)+ >

<!ATTLIST %CCML.description.qname;

  %Common;

  %CCML.ns.attrib;

>

<!-- end of catalog-1.mod -->
```

Now I get to combine these submodules with the other required modules so that the Course Catalog Markup Language will be a conforming XHTML Host Language application. See Listing 22.3 and the CD-ROM.

Listing 22.3: The DTD driver for CCML

```
<!-- Document Structure Module (required) -->

<!ENTITY % XHTML.version "-//WEBGEEK//DTD XHTML Catalog 1.0//EN">


<!ENTITY % CCML-qname.mod SYSTEM "catalog-qname-1.mod">


%CCML-qname.mod;


<!-- end of catalog-qname-1.mod -->

<!ENTITY % NS.prefixed.extras.attrib "">


<!-- Define the Content Model file for the framework to use -->

<!ENTITY % xhtml-model-1.mod "catalogf-model-1.mod">


<!-- reserved for future use with document profiles -->

<!ENTITY % XHTML.profile "">


<!-- Bi-directional text support is not included here -->

<!ENTITY % XHTML.bidi "IGNORE">
```

```
<!-- Modular Framework Module -->
<!ENTITY % xhtml-framework.module "INCLUDE">


<![ %xhtml-framework.module; [
<!ENTITY % xhtml-framework.mod PUBLIC "-//W3C//ENTITIES XHTML
   Modular Framework 1.0//EN" "xhtml-framework-1.mod">


%xhtml-framework.mod;


<!-- end of xhtml-framework-1.mod -->
]]>
<!-- Text Module (required) -->
<!ENTITY % xhtml-text.module "INCLUDE">


<![ %xhtml-text.module; [
<!ENTITY % xhtml-text.mod PUBLIC "-//W3C//ELEMENTS XHTML Text
   1.0//EN" "xhtml-text-1.mod">


%xhtml-text.mod;


<!-- end of xhtml-text-1.mod -->
]]>
<!-- Hypertext Module (required) -->
<!ENTITY % xhtml-hypertext.module "INCLUDE">


<![ %xhtml-hypertext.module; [
<!ENTITY % xhtml-hypertext.mod PUBLIC "-//W3C//ELEMENTS XHTML
   Hypertext 1.0//EN" "xhtml-hypertext-1.mod">


%xhtml-hypertext.mod;
<!-- end of xhtml-hypertext-1.mod -->

]]>

<!-- Lists Module (required) -->

<!ENTITY % xhtml-list.module "INCLUDE">


<![ %xhtml-list.module; [

<!ENTITY % xhtml-list.mod PUBLIC "-//W3C//ELEMENTS XHTML Lists


   1.0//EN" "xhtml-list-1.mod">


%xhtml-list.mod;


<!-- end of xhtml-list-1.mod -->

]]>

<!-- Catalog Module -->

<!ENTITY % CCML-elements.mod SYSTEM "catalog-elements-1.mod">


%CCML-elements.mod;


<!ENTITY % xhtml-struct.module "INCLUDE">


<!ENTITY % xhtml-struct.mod PUBLIC "-//W3C//ELEMENTS XHTML Document

   Structure 1.0//EN" "xhtml-struct-1.mod">


%xhtml-struct.mod;


<!-- end of xhtml-struct-1.mod -->

]]>
```

You may have noticed the reference in this DTD to the content model file for the DTD framework to use. That still needs to be defined. That segment is where the content models that I defined for the catalog elements are combined and exchanged with the content models defined in the W3C-provided modules. The content model file appears in Listing 22.4 and on the CD-ROM.

Listing 22.4: The CCML content model definition

```
<!-- ............................................... -->
<!-- Catalog Content Model Module  .................. -->
<!-- file: catalog-model-1.mod


     PUBLIC "-//WEBGEEK//ELEMENTS XHTML Catalog Model 1.0//EN"
     SYSTEM "http://www.webgeek.com/dtds/catalog/
       catalog-model-1.mod"
     xmlns:CCML="http://www.webgeek.com/xmlns/catalog"


     Revisions:
     (none)
     ................................................. -->
<!-- Define the content model for Misc.extra -->
<!ENTITY % Misc.class
   "| %script.qname; | %noscript.qname; ">


<!-- Inline Elements -->
<!ENTITY % Head-opts.mix
   "( %style.qname; | %meta.qname; ">
<!ENTITY % I18n.class "" >
<!ENTITY % Inlstruct.class "%br.qname; | %span.qname;" >
<!ENTITY % Inlphras.class
   "| %em.qname; | %strong.qname; | %dfn.qname; |
   %code.qname; | %samp.qname; | %kbd.qname; | %var.qname;
   | %cite.qname; | %abbr.qname; | $acronym.qname; |
   %q.qname;" >
<!ENTITY % Inlpres.class
   "| %tt.qname; | %i.qname; | %b.qname; | %big.qname; |
   %small.qname; | %sub.qname; | %sup.qname;" >
<!ENTITY % Anchor.class "| %a.qname;" >
<!ENTITY % Inlspecial.class "| %img.qname; " >
```

```
<!ENTITY % Inline.extra "" >

<!-- %Inline.class; includes all inline elements, and used

  as a component in mixes -->

<!ENTITY % Inline.class

  "%Inlstruct.class;

   %Inlphras.class;

   %Inlpres.class;

   %Anchor.class;

   %Inlspecial.class;"

>

<!-- %Inline-noA.class; includes all non-anchor inlines,

  used as a component in mixes -->

<!ENTITY % Inline-noA.class

  "%Inlstruct.class;

   %Inlphras.class;

   %Inlpres.class;

   %Inlspecial.class;"

>

<! -- %Inline-noA.mix; includes all non-anchor inlines -->

<!ENTITY % Inline-noA.mix

  "%Inline-noA.class;

   %Misc.class;"

>

<!-- Block Elements -->

<!ENTITY % Heading.class

  "%H1.qname; | %H2.qname; | %H3.qname; | %H4.qname; | %H5.qname;

  | %H6.qname;" >


<!ENTITY % List.class "%Ul.qname; | %Ol.qname; | %Dl.qname;" >

<!ENTITY % Blkstruct.class "%P.qname; | %Div.qname;" >

<!ENTITY %Blkphras.class
```

```
  "| %Pre.qname; | %Blockquote.qname; | %Address.qname;" >

<!ENTITY %Block.extra

  "| %CCML.course.qname; " >


<!-- %Block.class; includes all block elements, used as a component

  in mixes -->

<!ENTITY % Block.class

  "%Blkstruct.class;

   %Blkphras.class;

   %Blkpres.class;

   %Block.extra;"

>

<!-- %Block.mix; includes all block elements plus %Misc.class; -->

<!ENTITY % Block.mix

  "%Heading.class; | %List.class; | %Block.class; %Misc.class;"

>

<!-- All Content Elements -->

<!-- %Flow.mix; includes all text content, block, and inline -->

<!ENTITY % Flow.mix

  "%Heading.class;

  | %List.class;

  | %Block.class;

  | %Inline.class;

  %Misc.class;"

>

<!-- end of catalog-model-1.mod -->
```

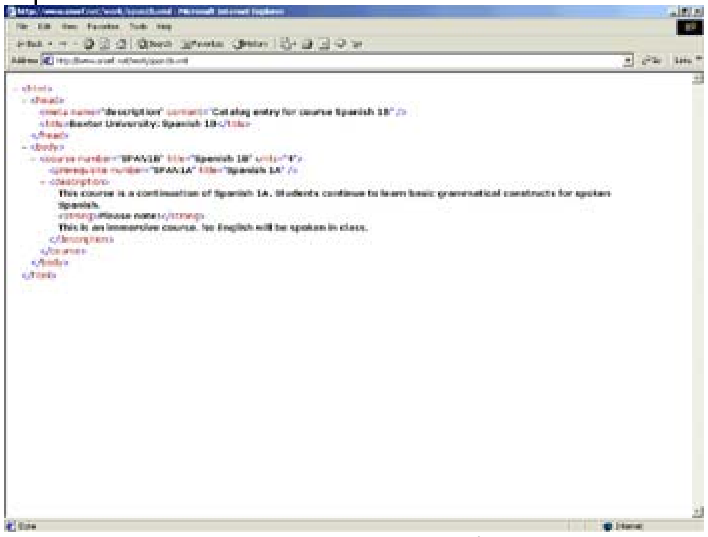
Now all that's left is to create a document using the new markup language! Listing 22.5, here and on the CD-ROM, shows a simple course display.

Listing 22.5: *span1b.xml*, a document based on the CCML Language

```
<!DOCTYPE html SYSTEM "catalog-1_0.dtd" >

<html>

<head>
```

```
<meta name="description" content="Catalog entry for course

   Spanish 1B" />

<title>Baxter University: Spanish 1B</title>

</head>

<body>

<course number="SPAN1B" title="Spanish 1B" units="4">

<prerequisite number="SPAN1A" title="Spanish 1A">

<description>

This course is a continuation of Spanish 1A. Students

   continue to learn basic grammatical constructs for

   spoken Spanish. <strong>Please note:</strong> This is an

   immersive course. No English will be spoken in class.

</description>

</course>

</body>

</html>
```

Figure 22.2 shows this file when saved as `span1b.xml` and rendered by the Internet Explorer 5.5 XML processor.



```
- <html>
  - <head>
      <meta name="description" content="Catalog entry for course Spanish 1B" />
      <title>Baxter University: Spanish 1B</title>
    </head>
  - <body>
    - <course number="SPAN1B" title="Spanish 1B" units="4">
        <prerequisite number="SPAN1A" title="Spanish 1A" />
      - <description>
          This course is a continuation of Spanish 1A. Students continue to learn basic grammatical constructs for spoken
          Spanish.
          <strong>Please note:</strong>
          This is an immersive course. No English will be spoken in class.
        </description>
      </course>
    </body>
  </html>
```

Figure 22.2: IE 5.5 shows a tree view of the new XHTML document when saved with an .XML filename.

The file can be viewed in a more traditional XHTML-like appearance by using a basic XSL style sheet. XSL, if you'll remember, is the Extensible Stylesheet Language, designed for use with XML; it functions in XML syntax, making the learning curve a little less steep than it would be for learning a completely new styling-language syntax. Listing 22.6 (also on the CD-ROM) has an extremely simple XSL style sheet. More advanced applications will use the power of XSL's matching capabilities to pull data from attribute values and then style that information, rather than having the author simply rewrite the XML file in HTML as this style-sheet template does.

Listing 22.6: A simple XSL style sheet for the XHTML file

```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

  <xsl:output method='html' indent='yes'
    doctype-public='-//W3C//DTD XHTML 1.0 Transitional//EN'/>

  <xsl:template match='/'>
  <html>
    <head>
      <title>Baxter University: Spanish 1B</title>
    </head>
  <body>
  <h2>Course SPAN1B: Spanish 1B</h2>
  <p><strong>Prerequisite:</strong>Course SPAN1A:
    Spanish 1A</p>
  <p>This course is a continuation of Spanish 1A. Students
  continue to learn basic grammatical constructs for
  spoken Spanish. <strong>Please note:</strong> This is an
  immersive course. No English will be spoken in class.</p>
  </body>
  </html>
  </xsl:template>
  </xsl:stylesheet>
```

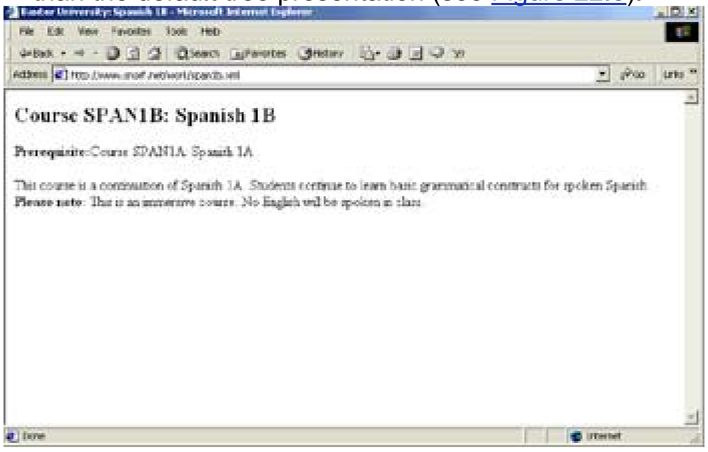
Tip Additional reading on XSL and its capabilities can be found in *Mastering XML* or *Mastering XHTML*, both from Sybex.

To bind this style sheet to the `span1b.xml` file, I only need to add a single line at the top of the document:

```
<?xml-stylesheet type="text/xsl" href="catalog.xsl"?>
<!DOCTYPE html SYSTEM "catalog-1_0.dtd" >
<html>
<head>
<meta name="description" content="Catalog entry for course Spanish
  1B" />
```

```
<title>Baxter University: Spanish 1B</title>
</head>
<body>
<course number="SPAN1B" title="Spanish 1B" units="4">
<prerequisite number="SPAN1A" title="Spanish 1A">
<description>
This course is a continuation of Spanish 1A. Students
  continue to learn basic grammatical constructs for
  spoken Spanish. <strong>Please note:</strong> This is an
  immersive course. No English will be spoken in class.
</description>
</course>
</body>
</html>
```

The line I added is formally known as a *processing instruction*, or PI. In this case, it notifies the XML parser in IE 5.5 that a style sheet should be used to render the XML output, giving it a more natural look than the default tree presentation (see Figure 22.3).


Baxter University: Spanish 1B - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address
Go
Links
Done
Internet

Course SPAN1B: Spanish 1B

Prerequisite: Course SPAN1A: Spanish 1A

This course is a continuation of Spanish 1A. Students continue to learn basic grammatical constructs for spoken Spanish. **Please note:** This is an immersive course. No English will be spoken in class.

Figure 22.3: The `span1b.xml` file rendered with an XSL style sheet attached

Summary

In this chapter, you've been introduced to the syntax used when writing DTDs—a necessary component of creating your own XHTML modules. After a quick tour of the construction of a new module, you saw how it was integrated into the XHTML Modularization framework to create a new XHTML Host Language—conformant DTD. Once the DTD was written, you saw how a simple XHTML file was created and viewed it in IE 5.5. By adding a simple style sheet, the tree structure presentation was transformed into something more familiar: headings and paragraphs.

As you've probably surmised, I've only scratched the surface of what can be done with XHTML Modularization and how to create and manage the modules and DTDs necessary to work with XHTML Host Languages. As a designer, you'll most likely be working with DTDs that have been made available to the public through the W3C, other open-source groups, or by your company's IT department. Should you wish to dive deeper into the world of XHTML module creation, additional resources are available for you to advance your knowledge.

Appendix A: XHTML Abstract Module Definitions

Overview

As discussed in Chapter 21, the Modularization of XHTML is done first by creating a set of abstract module definitions. The W3C provides abstract definitions for 28 modules, providing for most of the elements and attributes found in XHTML 1.

In order to make the abstract modules manageable for human readers, the commonly reoccurring attributes have been collected into five reusable attribute collections, which are then referred to by the collection name in the module definition. (You might think of it as a simpler form of using parameter entities.)
Each attribute is listed by its name and attribute type. XHTML attribute types include the eight basic types defined in the XML 1 Recommendation and an additional 19 types used within XHTML. Table A.1 defines these types.

Table A.1: Attribute Type Definitions

ATTRIBUTE TYPE	DEFINITION
CDATA	Character data.
ID	An identifier unique to the document.
IDREF	A reference to a document-unique identifier.
IDREFS	A space-separated list of references to document-unique identifiers.
NAME	A name with the same character constraints as ID above.
NMTOKEN	A name composed of only name tokens as defined in XML 1.
NMTOKENS	One or more white-space separated NMTOKEN values.
PCDATA	Processed character data.
Character	A single character from ISO10646.
Charset	A character encoding, as per RFC2045.
Charsets	A space-separated list of character encodings, as per RFC2045.
Color	The attribute value type *Color* refers to color definitions as specified in SRGB. A color value may be either a hexadecimal number (prefixed by a hash mark) or one of the following 16 color names. (The color values "#800080" and "Purple" both refer to the color purple.) The color names are case insensitive. Color names and SRGB values:
	Black="#000000"
	Green="#008000"
	Silver="#C0C0C0"
	Lime="#00FF00"
	Gray="#808080"
	Olive="#808000"
	White="#FFFFFF"
	Yellow="#FFFF00"
	Maroon="#800000"
	Navy="#000080"
	Red="#FF0000"
	Blue="#0000FF"
	Purple="#800080"
	Teal="#008080"
	Fuschia="#FF00FF"
	Aqua="#00FFFF"

Table A.1: Attribute Type Definitions

ATTRIBUTE TYPE	DEFINITION
ContentType	A media type, as per RFC2045.
ContentTypes	A comma-separated list of media types, as per RFC2045.
Datetime	Date and time information.
FrameTarget	Frame name used as destination for results of certain actions.
LanguageCode	A language code, as per RFC1766.
Length	The value may be either in pixels or a percentage of the available horizontal or vertical space. Thus, the value `"50%"` means half of the available space.
LinkTypes	Authors may use the following recognized link types, listed here with their conventional interpretations. A LinkTypes value refers to a space-separated list of link types. White-space characters are not permitted within link types. These link types are case insensitive, i.e., *Alternate* has the same meaning as *alternate*. User agents, search engines, etc. may interpret these link types in a variety of ways. For example, user agents may provide access to linked documents through a navigation bar.
	Alternate Designates substitute versions for the document in which the link occurs. When used together with the `xml:lang` attribute, it implies a translated version of the document. When used together with the media attribute, it implies a version designed for a different medium.
	Stylesheet Refers to an external style sheet. See the Style Module for details. This is used together with the link type Alternate for a user-selectable alternate style sheet.
	Start Refers to the first document in a collection of documents. This link type tells search engines which document is considered by the author to be the starting point of the collection.
	Next Refers to the next document in a linear sequence of documents. User agents may choose to preload the "next" document to reduce the perceived load time.
	Prev Refers to the previous document in an ordered series of documents. Some user agents also support the synonym "Previous."
	Contents Refers to a document serving as a table of contents. Some user agents also support the synonym ToC (from *Table of Contents*).
	Index Refers to a document providing an index for the current document.
	Glossary Refers to a document providing a glossary of terms that pertain to the current document.
	Copyright Refers to a copyright statement for the current document.
	Chapter Refers to a document serving as a chapter in a collection

Table A.1: Attribute Type Definitions

ATTRIBUTE TYPE	DEFINITION
	of documents.
	Section Refers to a document serving as a section in a collection of documents.
	Subsection Refers to a document serving as a subsection in a collection of documents.
	Appendix Refers to a document serving as an appendix in a collection of documents.
	Help Refers to a document offering help (such as more information or links to other sources information).
	Bookmark Refers to a bookmark. A bookmark is a link to a key entry point within an extended document. The `title` attribute may be used, for example, to label the bookmark. Note that several bookmarks may be defined in each document.
MediaDesc	The MediaDesc attribute is a comma-separated list of media descriptors. The following is a list of recognized media descriptors:
	screen Intended for nonpaged computer screens.
	tty Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities.
	tv Intended for television-type devices (low resolution, color, limited scrollability.).
	projector Intended for projectors.
	handheld Intended for handheld devices (small screen, monochrome, bitmapped graphics, limited bandwidth).
	print Intended for paged, opaque material and for documents viewed on-screen in print-preview mode.
	braille Intended for Braille tactile feedback devices.
	aural Intended for speech synthesizers.
	all Suitable for all devices.
MultiLength	The value may be a length or a relative length. A relative length has the form `i*`, where `i` is an integer. When allotting space among elements competing for that space, user agents allot pixel and percentage lengths first and then divide up remaining available space among relative lengths. Each relative length receives a portion of the available space proportional to the integer preceding the *. The value * is equivalent to 1*. Thus, if 60 pixels of space are available after the user agent allots pixel and percentage space, and the competing relative lengths are 1*, 2*, and 3*; the 1* will be allotted 10 pixels, the 2* 20 pixels, and the 3* 30 pixels.
Number	One or more digits.

Table A.1: Attribute Type Definitions

ATTRIBUTE TYPE	DEFINITION
Pixels	The value is an integer that represents the number of pixels of the canvas (screen, paper). Thus, the value *50* means 50 pixels. For normative information about the definition of a pixel, please consult CSS2.
Script	Script data can be the content of the `script` element and the value of intrinsic event attributes. User agents must not evaluate script data as XHTML markup but instead must pass it on as data to a script engine.
	The case sensitivity of script data depends on the scripting language.
	Please note that script data that is element content may not contain character references, but script data that is the value of an attribute may contain them.
Text	Arbitrary textual data, likely meant to be human-readable.
URI	A Uniform Resource Identifier.
URIs	A space-separated list of Uniform Resource Identifiers.

Future versions of XHTML may introduce new values and may allow parameterized values. To facilitate the introduction of these extensions, conforming user agents must be able to parse the media attribute value as follows:

1. The value is a comma-separated list of entries. For example, consider the following code:
2. media="screen, 3d-glasses, print and resolution > 90dpi"

It is mapped to:
"screen"
"3d-glasses"
"print and resolution > 90dpi"

3. Each entry is truncated just before the first character that isn't a U.S. ASCII letter (a–z, A–Z or ISO 10646 hex 41-5a, 61-7a), digit (0–9, hex 30–39), or hyphen (hex 2D). In the example, the truncation produces the following:
4. "screen"
5. "3d-glasses"
6. "print"
7. A case-sensitive match is then made with the set of media types defined. User agents may ignore entries that don't match. In the example, you are left with screen and print.

Tip Style sheets may include media-dependent variations within them (e.g., the CSS @media construct). In such cases it may be appropriate to use "media=all".

Now that the Attribute types have been defined, I can define the attribute collections. In Table A.2, each member attribute is followed by its associated type.

Table A.2: Attribute Collection Definitions

COLLECTION NAME	ATTRIBUTES IN COLLECTION
Core	`class` (NMTOKENS), `id` (ID), `title` (CDATA)
I18N (Internationalization)	`xml:lang` (NMTOKEN)
Events	`onclick` (Script), `ondblclick` (Script), `onmousedown` (Script), `onmouseup` (Script), `onmouseover` (Script), `onmousemove` (Script), `onmouseout` (Script), `onkeypress` (Script), `onkeydown` (Script), `onkeyup` (Script)
Style	`style` (CDATA)

Table A.2: Attribute Collection Definitions

COLLECTION NAME	ATTRIBUTES IN COLLECTION
Common	Core, Events, I18N, Style

Note that the Events collection is only defined when the Intrinsic Events Module is selected. Otherwise, the Events collection is empty. Also note that the Style collection is only defined when the Style Attribute Module is selected. Otherwise, the Style collection is empty.

Core Modules

The core modules are modules required to be present in any XHTML Family Conforming Document Type.

Structure Module

The Structure Module defines the major structural elements for XHTML. These elements effectively are the basis for the content model of many XHTML family document types. The elements and attributes included in this module are listed in Table A.3.

Table A.3: Elements and Attributes in the Structure Module

ELEMENTS	ATTRIBUTES	MINIMAL CONTENT MODEL
`body`	Common	(Heading | Block | List)*
`head`	I18N, profile (URI)	title
`html`	I18N, version (CDATA), `xmlns (URI = http://www.w3.org/1999/xhtml")`	`head, body`
`title`	I18N	PCDATA

This module is the basic structural definition for XHTML content. The `html` element acts as the root element for all XHTML Family Document Types.
Note that the value of the `xmlns` attribute is defined to be "`http://www.w3.org/1999/xhtml`". Also note that because the `xmlns` attribute is treated specially by XML Namespace-aware parsers, it is legal to have it present as an attribute of each element. However, any time the `xmlns` attribute is used in the context of an XHTML module, whether with a prefix or not, the value of the attribute shall be the XHTML namespace defined here.

Text Module

The text module defines the elements and attributes that define the different text structures, such as headings and paragraphs. The elements and attributes included in this module are listed in Table A.4.

Table A.4: Text Module

ELEMENTS	ATTRIBUTES	MINIMAL CONTENT MODEL
`abbr`	Common	(PCDATA | Inline)*
`acronym`	Common	(PCDATA | Inline)*
`address`	Common	(PCDATA | Inline)*

Table A.4: Text Module

ELEMENTS	ATTRIBUTES	MINIMAL CONTENT MODEL
`blockquote`	Common, `cite` (URI)	(PCDATA | Heading | Block)*
`br`	Core	EMPTY
`cite`	Common	(PCDATA | Inline)*
`code`	Common	(PCDATA | Inline)*
`dfn`	Common	(PCDATA | Inline)*
`div`	Common	(Heading | Block | List)*
`em`	Common	(PCDATA | Inline)*
`h1`	Common	(PCDATA | Inline)*
`h2`	Common	(PCDATA | Inline)*
`h3`	Common	(PCDATA | Inline)*
`h4`	Common	(PCDATA | Inline)*
`h5`	Common	(PCDATA | Inline)*
`h6`	Common	(PCDATA | Inline)*
`kbd`	Common	(PCDATA | Inline)*
`p`	Common	(PCDATA | Inline)*
`pre`	Common, `xmlspace:"preserve"`	(PCDATA | Inline)*
`q`	Common, `cite` (URI)	(PCDATA | Inline)*
`samp`	Common	(PCDATA | Inline)*
`span`	Common	(PCDATA | Inline)*
`strong`	Common	(PCDATA | Inline)*
`var`	Common	(PCDATA

Table A.4: Text Module

ELEMENTS	ATTRIBUTES	MINIMAL CONTENT MODEL
		| Inline)*

The minimal content model for this module defines some content sets:

Content Set	Included Elements
Heading	`h1	h2	h3	h4	h5	h6`
Block	`address	blockquote	div	p	pre`
Inline	`abbr	acronym	br	cite	code	dfn	em	kbd	q	samp	span	strong	var`
Flow	`Heading	Block	Inline`

Hypertext Module

The Hypertext Module provides the element that is used to define hypertext links to other resources. The elements and attributes included in this module are in Table A.5:

Table A.5: Hypertext Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
a	Common, `accesskey` (Character), `charset` (Charset), `href` (URI), `hreflang` (LanguageCode), `rel` (LinkTypes), `rev` (LinkTypes), `tabindex` (Number), `type` (ContentType)	(PCDATA | Inline - a)*

This module adds the a element to the Inline content set of the Text Module.

List Module

As its name suggests, the List Module provides list-oriented elements. The List Module supports the elements and attributes as listed in Table A.6.

Table A.6: List Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`dl`	Common	(`dt` | `dd`) +
`dt`	Common	(PCDATA | Inline)*
`dl`	Common	(PCDATA | Inline)*
`Iol`	Common	`li+`
`ul`	Common	`li+`
`li`	Common	(PCDATA | Inline)*

This module also defines the content set List with the minimal content model `(dl | ol | ul)+` and adds this set to the Flow content set of the Text Module.

Applet Module

The Applet Module provides elements for referencing external applications. Specifically, the Applet Module supports the elements and attributes in Table A.7:

Table A.7: Applet Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`applet`	Core, `alt` (Text), `archive` (CDATA), `code` (CDATA), `codebase` (URI), `height` (Length), `name` (CDATA), `object` (CDATA), `width` (Length)	`param?`
`param`	`id` (ID), `name` (CDATA), `type` (ContetType), `value` (CDATA), `valuetype` `("data" | "ref" | "object")`	EMPTY

Warning The Applet Module is deprecated. Similar functionality can be found in the Object Module.

When the Applet Module is used, it adds the `applet` element to the `Inline` content set of the Text Module.

Text Extension Modules

This section defines a variety of additional textual markup modules.

Presentation Module

This module defines elements, attributes, and a minimal content model for simple presentation-related markup in Table A.8.

Table A.8: Presentation Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`b`	Common	(PCDATA | Inline)*
`big`	Common	(PCDATA | Inline)*
`hr`	Common	EMPTY
`i`	Common	(PCDATA

Table A.8: Presentation Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
		| Inline)*
`small`	Common	(PCDATA | Inline)*
`sub`	Common	(PCDATA | Inline)*
`sup`	Common	(PCDATA | Inline)*
`tt`	Common	(PCDATA | Inline)*

When this module is used, the `hr` element is added to the Block content set of the Text Module. In addition, the `b`, `big`, `i`, `small`, `sub`, `sup`, and `tt` elements are added to the Inline content set of the Text Module.

Edit Module

The Edit module, as seen in Table A.9, defines elements and attributes for use in editing-related markup.

Table A.9: Edit Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`del`	Common, `cite` (URI), `datetime` (Datetime)	(PCDATA | Inline)*
`insert`	Common, `cite` (URI), `datetime` (Datetime)	(PCDATA | Inline)*

When this module is used, the `del` and `ins` elements are added to the Inline content set of the Text Module.

Bi-directional Text Module

The Bi-directional Text module, Table A.10, defines an element that can be used to declare the bi-directional rules for the element's content.

Table A.10: Bi-directional Text Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`bdo`	Core, `dir* ("ltr" | "rtl")`	(PCDATA | Inline)*

When this module is used, the `bdo` element is added to the Inline content set of the Text Module. Selecting this module also adds the attribute `dir* ("ltr" | "rtl")` to the `I18N` attribute collection.

Forms Modules

Two forms modules are defined here, one conforming to a minimal set of form elements, the second including the expanded forms introduced in HTML 4.

Basic Forms Module

The Basic Forms Module, provides the form-related elements, but only in a limited form. The Basic Forms Module supports the elements, attributes, and minimal content model as listed in Table A.11.

Table A.11: Basic Forms Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`form`	Common, `action*` (URI), `method ("get"* | "post")`, `enctype` (ContentType)	Heading | Block - form
input	Common, `accesskey` (Character), `checked ("checked")`, `maxlength` (Number), `name` (CDATA), `size` (Number), `src` (URI), `type ("text"* | "password" | "checkbox" | "radio" | "submit" | "reset" | "hidden"),value` (CDATA)	EMPTY
`label`	Common, `accesskey` (Character), `for` (IDREF)	(PCDATA | Inline - `label`)*
`select`	Common, `multiple` ("multiple"), `name` (CDATA), `size` (Number)	`option+`
`option`	Common, `selected ("selected`	Inline*

Table A.11: Basic Forms Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
	`"), value` (CDATA)	
`textarea`	Common, `accesskey` (Character), `cols*` (Number), `name` (CDATA), `rows*` (Number)	PCDATA*

This module defines two content sets:

Content Set	Included Elements
Form	`form`
Formctrl	`input | label | select | textarea`

When this module is used, it adds the `Form` content set to the `Block` content set and it adds the `Formctrl` content set to the `Inline` content set as these are defined in the Text Module.

Forms Module

The Forms Module provides all of the forms features found in HTML 4. The Forms Module supports the elements and attributes as listed in Table A.12.

Table A.12: Forms Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`form`	Common, `accept` (ContentTypes), `acceptcharset` (Charsets), `action*` (URI), `method ("get"* | "post"), enctype` (ContentType)	(Heading | Block - `form | fieldset)+`
`input`	Common, `accept` (ContentTypes), `accesskey` (Character), `alt` (CDATA), `checked ("checked"), disabled ("disabled"), maxlength` (Number), `name` (CDATA), `readonly ("readonly"), size` (Number), `src` (URI), `tabindex` (Number), `type ("text"* | "password" | "checkbox" | "button" | "radio" | "submit" | "reset" | "file" | "hidden" | "image"), value` (CDATA)	EMPTY
`select`	Common, `disabled ("disabled")`, multiple `("multiple"), name` (CDATA), `size`	`(optgroup |`

Table A.12: Forms Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
	(Number), `tabindex` (Number)	`option)+`
`option`	Common, `disabled ("disabled"), label` (Text), `selected ("selected"), value` (CDATA)	PCDATA
`textarea`	Common, `accesskey` (Character), `cols*` (Number), `disabled ("disabled"), name` (CDATA), `readonly ("readonly"), rows*` (Number), `tabindex` (Number)	PCDATA
`button`	Common, `accesskey` (Character), `disabled ("disabled"), name` (CDATA), `tabindex` (Number) `type ("button" | "submit"* | "reset"), value` (CDATA)	(PCDATA | Heading | List | Block - Form | Inline - Formctrl)*
`fieldset`	Common	(PCDATA | legend | Flow)*
`label`	Common, `accesskey` (Character), `for` (IDREF)	(PCDATA | Inline - `label`)*
`legend`	Common, `accesskey` (Character)	(PCDATA | Inline)+
`optgroup`	Common, `disabled ("disabled"), label*` (Text)	`option+`

This module defines two content sets:

Content Set	Included Elements
Form	`form | fieldset`
Formctrl	`input | select | textarea | label | button`

When this module is used, it adds the Form content set to the Block content set and it adds the Formctrl content set to the Inline content set as these are defined in the Text Module.

Warning The Forms Module is a superset of the Basic Forms Module. These modules may not be used together in a single document type.

Table Modules

As with forms, there are two table modules. One provides basic table construction, the other adds in the newer features found primarily in HTML 4.

Basic Tables Module

The Basic Tables Module provides table-related elements, but only in a limited form. See Table A.13.

Table A.13: Forms Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`caption`	Common	(PCDATA | Inline)*
`table`	Common , `summary` (Text), `width` (Length)	`caption?, tr+`
`td`	Common, `abbr` (Text), `align ("left" | "center" | "right")`, `axis` (CDATA), `colspan` (Number), `headers` (IDREFS), `rowspan` (Number), `scope ("row" | "col")`, `valign ("top" | "middle" | "bottom")`	(PCDATA | Flow - `table`)*
`th`	Common, `abbr` (Text), `align ("left" | "center" | "right")`, `axis` (CDATA), `colspan` (Number), `headers` (IDREFS), `rowspan` (Number), `scope ("row" | "col" | "rowgroup" | "colgroup")`, `valign ("top" | "middle" | "bottom")`	(PCDATA | Flow - table)*
`tr`	Common, `align ("left" | "center" | "right")`, `valign ("top" | "middle" | "bottom")`	`(td)+`

When this module is used, it adds the `table` element to the Block content set as defined in the Text Module.

Tables Module

As its name suggests, the Tables Module provides table-related elements that are better able to be accessed by non-visual user agents. Specifically, the Tables Module supports the elements, attributes, and content model in Table A.14.

Table A.14: Tables Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`caption`	Common	(PCDATA | Inline)*
`table`	Common, `border` (Pixels), `cell padding` (Length), `cell spacing` (Length) `datapagesize` (CDATA), `frame ("void" | "above" | "below" | "hsides" | "lhs" | "rhs" | "vsides" | "box" | "border")`, `rules ("none" | "groups" | "rows" | "cols" | "all")`, `summary` (Text), `width` (Length)	caption?, (`col*` | `colgroup*`), ((`thead?, tfoot?, tbody+`) | (`tr+`))
`td`	Common, `abbr` (Text), `align ("left" | "center" | "right" | "justify" | "char")`, `axis` (CDATA), `char` (Character), `charoff` (Length), `colspan` (Number), `headers` (IDREFS), `rowspan` (Number), `scope ("row", "col", "rowgroup", "colgroup")`, `valign ("top" | "middle" | "bottom" | "baseline")`	(PCDATA | Flow)*

Table A.14: Tables Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
th	Common, abbr (Text), align ("left" | "center" | "right" | "justify" | "char"), axis (CDATA), char (Character), charoff (Length), colspan (Number), headers (IDREFS), rowspan (Number), scope ("row", "col", "rowgroup", "colgroup"), valign ("top" | "middle" | "bottom" | "baseline")	(PCDATA | Flow)*
tr	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), valign ("top" | "middle" | "bottom" | "baseline")	(td | th)+
col	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), span (Number), valign ("top" | "middle" | "bottom" | "baseline"), width (MultiLength)	EMPTY
colgroup	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), span (Number), valign ("top" | "middle" | "bottom" | "baseline"), width (MultiLength)	col*
tbody	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), valign ("top" | "middle" | "bottom" | "baseline")	tr+
thead	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), valign ("top" | "middle" | "bottom" | "baseline")	tr+
tfoot	Common, align ("left" | "center" | "right" | "justify", "char"), char (Character), charoff (Length), valign ("top" | "middle" | "bottom" | "baseline")	tr+

When this module is used, it adds the table element to the Block content set of the Text Module.

Image Module

The Image Module provides basic image embedding, and may be used in some implementations independently of client-side image maps. See Table A.15

Table A.15: Image Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
img	Common, alt* (Text), height (Length), longdesc (URI), src* (URI), width (Length)	EMPTY

When this module is used, it adds the img element to the Inline content set of the Text Module.

Client-Side Image Map Module
The Client-Side Image Map Module provides elements for client-side image maps. It requires that the Image Module (or another module that supports the `img` element) be included. The Client-Side Image Map Module supports the elements listed in Table A.16.

Table A.16: Client-Side Image Map Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`a&`	`coords` (CDATA), `shape ("rect" | "circle" | "poly" |"default")`	N/A
`area`	Common `accesskey` (Character), `alt*` (Text), `coords` (CDATA), `href` (URI), `noherf ("noherf"), shape ("rect"* | "circle" | "poly" | "default"), tabindex` (Number)	EMPTY
`img&`	`usemap` (IDREF)	N/A
`map`	I18N, Events, `class` (NMTOKEN), `id*` (ID), `title` (CDATA)	((Heading | Block) | `area`)+
`object&`	`usemap` (IDREF)	Note: only when the Object Module is included

When this module is used, the map element is added to the Inline content set of the Text Module.

Server-Side Image Map Module
The Server-Side Image Map Module provides support for image-selection and transmission of selection coordinates. It requires that the Image Module (or another module that supports the `img` element) be included. See Table A.17.

Table A.17: Server-Side Image Map Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`img&`	`ismap "ismap"`	N/A

Object Module
The Object Module provides elements for general-purpose object inclusion. The elements and attributes included in this module are listed in Table A.18.

Table A.18: Object Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`object`	Common, `archive` (URIs), `classid` (URI), `codebase` (URI), `codetype` (ContentType), `data` (URI), `declare ("declare"), height` (Length), `standby` (Text), `tabindex` (Number), `type` (ContentType), `width` (Length)	(PCDATA | Flow | `param`)*

Table A.18: Object Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`param`	`id` (ID), `name*` (CDATA), `type` (ContentType), `value` (CDATA), `valuetype ("data" | "ref" | "object")`	EMPTY

When this module is used, it adds the object element to the Inline content set of the Text Module.

Frames Module
As its name suggests, the Frames Module provides frame-related elements. Specifically, the Frames Module supports the elements and attributes in Table A.19.

Table A.19: Frames Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`frameset`	Core, `cols` (MultiLength), `rows` (MultiLength)	`frame+, noframes?`
`frame`	Core, `frameborder` ("1" | "0"), `longdesc` (URI), `marginheight` (Pixels), `marginwidth` (Pixels), `noresize ("noresize")`, `scrolling ("yes" | "no" | "auto"*)`, `src` (URI)	EMPTY
`noframes`	Common	`body`

When this module is selected, the minimal content model of the `html` element of the structure module is changed to `(head, frameset)`.

Target Module
The content of a frame can specify destination targets for a selection. This module adds the `target` element to the `area` and `link` defining elements. This is defined as a separate module so it can be included in documents that will be included in frames and documents that use the target feature to open a new window. The elements and attributes included in this module are listed in Table A.20.

Table A.20: Target Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`a&`	`target` (CDATA)	N/A
`area&`	`target` (CDATA)	When the Client-Side Image Map module is selected.
`base&`	`target` (CDATA)	When the Legacy Module is selected.

Table A.20: Target Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`link&`	`target` (CDATA)	When the Link Module is selected.
`form&`	`target` (CDATA)	When the Basic Forms or Forms Module is selected.

Iframe Module
The Iframe Module defines an element for the definition of inline frames. The element and attributes included in this module are in Table A.21.

Table A.21: Iframe Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`iframe`	Core, `frameborder` ("1" | "0"), `height` (Pixels), `longdesc` (URI), `marginheight` (Pixels), `margin-width` (Pixels), `scrolling ("yes") | "no" | "auto"*),` `src` (URI), `width` (Length)	Flow

When this module is used, the `iframe` element is added to the `Inline` content set as defined by the Text Module.

Intrinsic Events Module
Intrinsic events are attributes that are used in conjunction with elements that can have specific actions occur when certain events are performed by the user. The attributes indicated in the following table are added to the attribute set for their respective elements only when the modules defining those elements are selected. Note also that selection of this module defines the attribute collection `Events` as described above. The attributes defined by this module are in Table A.22.

Table A.22: Intrinsic Events Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`a&`	`onblur` (Script), `onfocus` (Script)	N/A
`area&`	`onblur` (Script), `onfocus` (Script)	When the Client-Side Image Map Module is also used
`form&`	`onreset` (Script), `onsubmit` (Script)	When the Basic Forms or Forms Module is

Table A.22: Intrinsic Events Module

ELEMENT	ATTRIBUTES	MINIMAL CONTE NT MODEL
		used
`body&`	`onload` (Script), `onunload` (Script)	N/A
`label&`	`onblur` (Script), `onfocus` (Script)	When the Forms Module is used
`input&`	`onblur` (Script), `onchange` (Script), `onfocus` (Script), `onselect` (Script)	When the Basic Forms or Forms Module is used
`select&`	`onblur` (Script), `onchange` (Script), `onfocus` (Script)	When the Basic Forms or Forms Module is used
`textarea&`	`onblur` (Script), `onchange` (Script), `onfocus` (Script), `onselect` (Script)	When the Basic Forms or Forms Module is used
`button&`	`onblur` (Script), `onfocus` (Script)	When the Forms Module is used

Metainformation Module
The Metainformation Module defines an element that describes information within the declarative portion of a document (in XHTML within the `head` element). The element defined in this module is in Table A.23.

Table A.23: Metainformation Module

ELEMENT	ATTRIBUTES	MINIMAL CONTE NT MODEL
`meta`	I18N, `content*` (CDATA), `http-equiv` (NMTOKEN), `name` (NMTOKEN), `scheme` (CDATA)	EMPTY

When this module is selected, the `meta` element is added to the content model of the `head` element as defined in the Structure Module. Scripting Module

Scripting Module
The Scripting Module defines elements that are used to contain information pertaining to executable scripts or the lack of support for executable scripts. Elements and attributes included in this module are listed in Table A.24.

Table A.24: Scripting Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`noscript`	Common	(Heading | List | Block)+
`script`	`charset` (Charset), `defer ("defer")`, `src` (URI), `type*` (ContentType), `xml:space="preserve"`	PCDATA

When this module is used, the `script` and `noscript` elements are added to the Block and Inline content sets of the Text Module. In addition, the `script` element is added to the content model of the `head` element defined in the Structure Module.

Stylesheet Module
The Stylesheet Module defines an element to be used when declaring internal stylesheets. The element and attributes defined by this module are in Table A.25.

Table A.25: Stylesheet Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`style`	I18N, `media` (MediaDesc), `title` (Text), `type*` (ContentType), `xml:space="preserve"`	PCDATA

When this module is used, it adds the `style` element to the content model of the `head` element of the Structure Module.

Style Attribute Module
The Style Attribute Module defines the `style` attribute. When this module is selected, it activates the Style Attribute Collection.

Link Module
The Link Module defines an element that can be used to define links to external resources. These resources are often used to augment the user agent's ability to process the associated XHTML document. Table A.26 defines the element and attributes included in the Link module.

Table A.26: Link Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`link`	Common, `charset` (Charset), `href` (URI), `hreflang`	EMPTY

Table A.26: Link Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
	(LanguageCode), `media` (MediaDesc), `rel` (LinkTypes), `rev` (LinkTypes), `type` (ContentType)	

When this module is used, it adds the `link` element to the content model of the `head` element as defined in the Structure Module.

Base Module

The Base Module defines an element that can be used to define a base URI against which relative URIs in the document will be resolved. Table A.27 defines the elements and attributes included in the Base module.

Table A.27: Base Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`base`	`href*` (URI)	EMPTY

When this module is used, it adds the `base` element to the content model of the `head` element of the Structure Module.

Name Identification Module

The Name Identification module defines the attribute `name` for a collection of elements. The `name` attribute was historically used to identify certain elements within HTML documents. While the `name` attribute has been supplanted by the `id` attribute in all of these elements, there may be instances where markup languages will wish to support both. Such markup languages may do so by including this module.

Note that by including this module, both the `name` and `id` attributes are defined for the elements indicated (see Table A.28). In this situation, if the `name` attribute is defined for an element, the `id` attribute must also be defined. Further, these attributes must both have the same value. Finally, when documents that use this attribute are served as Internet Media Type `"text/xml"` or `"application/xml"`, the value of the `name` attribute on these elements shall not be used as a fragment identifier.

Table A.28: Name Identification Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`a&`	`name` (CDATA)	N/A
`applet&`	`name` (CDATA)	When the Applet Module is selected.
`form&`	`name` (CDATA)	When the Forms or

Table A.28: Name Identification Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
		Basic Forms Module is selected.
`frame&`	`name` (CDATA)	When the Frames Module is selected.
`iframe&`	`name` (CDATA)	When the Iframe Module is selected.
`img&`	`name` (CDATA)	When the Image Module is selected.
`map&`	`name` (CDATA)	When the Client-Side Image Map module is selected.

Warning The Name Identification Module is deprecated.

Legacy Module

The Legacy Module defines elements and attributes that were deprecated in previous versions of HTML and XHTML. While the use of these elements and attributes is no longer encouraged, they are provided in this module to ease their integration should markup language authors wish to support them. Table A.29 defines the elements and attributes used when the Legacy Module is selected.

Table A.29: Legacy Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`basefont`	`color` (Color), `face` (CDATA), `id` (ID), `size` (CDATA)	EMPTY
`center`	Common	(PCDATA | Flow)*
`font`	Common, `color` (Color), `face` (CDATA), `size` (CDATA)	(PCDATA | Inline)*

Table A.29: Legacy Module

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`s`	Common	(PCDATA | Inline)*
`strike`	Common	(PCDATA | Inline)*
`u`	Common	(PCDATA | Inline)*

Table A.30 shows additional attributes for elements defined elsewhere when the Legacy module is selected.

Table A.30: Legacy Module Selected

ELEMENT	ATTRIBUTES	MINIMAL CONTENT MODEL
`body&`	`alink` (Color), `background` (URI), `bgcolor` (Color, `link` (Color), `text` (Color), `vlink` (Color)	N/A
`br&`	`clear ("left" | "all" | "right" | "none"*)`	N/A
`caption&`	`align ("left" | "center" | "right" | "justify")`	N/A
`div&`	`align ("left" | "center" | "right" | "justify")`	N/A
`h1-h6&`	`align ("left" | "center" | "right" | "justify")`	N/A
`hr&`	`align ("left" | "center" | "right" | "justify") noshade ("noshade"), size` (Pixels), `width` (Length),	N/A
`img&`	`align ("left" | "center" | "right" | "justify") border` (Pixels), `hspace` (Pixels), `vspace` (Pixels)	N/A
`input&`	`align ("left" | "center" | "right" | "justify")`	When the Basic Forms or Forms module is selected
`legend&`	`align ("left" | "center" | "right" | "justify")`	When the Forms Module is selected
`li&`	`type` (CDATA), `value` (Number)	N/A
`ol&`	`compact ("compact"), start` (Number), `type` (CDATA)	N/A
`p&`	`align ("left" | "center" | "right", "justify")`	N/A
`pre&`	`width` (Number)	N/A
`script&`	`language` (CDATA)	When the

Table A.30: Legacy Module Selected

ELEMENT	ATTRIBUTES	MINIMAL CONTE NT MODEL
		Scripting Module is selected
`table&`	`align ("left" | "center" | "right" | "justify"), bgcolor` (Color)	When the Tables Module is selected
`tr&`	`bgcolor` (Color)	When the Tables Module is selected
`th&`	`bgcolor` (Color), `height` (Pixels) `nowrap ("nowrap"), width` (Pixels)	When the Tables Module is selected
`td&`	`bgcolor` (Color), `height` (Pixels) `nowrap ("nowrap"), width` (Color)	When the Tables Module is selected
`ul&`	`compact ("compact"), type` (CDATA)	

Appendix B: XHTML Element and Attribute Quick Reference

Overview

This appendix is a quick reference of valid XHTML elements and attributes. Three attribute collections are defined here, for convenience purposes only, and are found immediately following this introduction.

All XHTML elements and attributes are listed here. If an element or attribute is allowed in the Transitional document type or both the Transitional and Frameset document type, it will be followed by a (T) notation. Elements and attributes without the letter designation are also allowed in XHTML 1 Strict.

Event Attributes

- `onclick=`script
- ondblclick=`script`
- onkeydown=`script`
- onkeypress=`script`
- onkeyup=`script`
- onmousedown=`script`
- onmousemove=`script`
- onmouseout=`script`
- onmouseover=`script`
- onmouseup=`script`

Core Attributes

- `class=`name
- dir=`dir` (T)
- id=`name`
- lang=`language` (T)
- style=`style`
- title=`string`

Tip When working in XHTML 1 Strict, the Core Attribute set consists only of class, id, style, *and* title.

Internationalization (I18N) Attributes

- `lang=`*`language code`*
- `xml:lang=`*`language code`*
- `dir=`*`direction`*

Framework Elements

base

- `href=`URI
- target=`name` (T)

body

- Core Attribute set
- Event Attribute set (T)
- I18N Attribute set
- alink=`color` (T)
- background=`url` (T)
- bgcolor=`color` (T)
- link=`color` (T)
- text=`color` (T)
- vlink=`color` (T)

div

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

head

- I18N Attribute set
- profile=`URI`

html

- I18N Attribute set
- xmlns=`URI`

link

- Core Attribute set
- Event Attribute set
- I18N Attribute set
- charset=`character set code`
- href=`URI`
- hreflang=`language code`
- type=`content type`
- rel=`link type`
- rev=`link type`
- media=`media description`
- target=`name` (T)

meta

- I18N Attribute set
- http-equiv=`CDATA`
- name=`name`
- content=`CDATA`
- scheme=`CDATA`

noscript

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

script

- `Charset=`character set code
- type=`content type`
- language=`CDATA` (T)
- src=`URI`
- defer=defer
- xml:space=preserve

style

- I18N Attribute set
- type=`content type`
- media=`media description`
- title=`string`
- xml:space=preserve

Structural Elements

a

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- charset=`character set code`
- type=`content type`
- name=`NMTOKEN`
- href=`URI`
- hreflang=`language code`
- rel=`link types`
- rev=`link types`
- accesskey=`character`

- shape=`shape`
- coords=`coordinates`
- tabindex=`number`
- onfocus=`script`
- onblur=`script`
- target=`name` (T)

address

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

blockquote

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- cite=`URI`

center (T)

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

hr

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)
- noshade=noshade (T)
- size=`pixels` (T)
- width=`length` (T)

p

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

pre

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- width=`number` (T)
- xml:space=preserve

Headings

h1

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

h2

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

h3

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

h4

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

h5

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

h6

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

Lists

dd

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

dl

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- compact=compact (T)

dt

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

li

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- type=`list item style` (T)
- value=`number` (T)

ol

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- type=`list style` (T)
- compact=compact (T)
- start=`number` (T)

ul

- Core Attribute Set

- Event Attribute Set
- I18N Attribute Set
- type=`list style` (T)
- compact=compact (T)

Inline Elements

abbr

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

acronym

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

b

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

basefont (T)

- `id=`id
- size=`CDATA`
- color=`color`
- face=`CDATA`

bdo

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

big

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

br

- Core Attribute Set
- clear=`value` (T)

cite

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

code

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

del

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- cite=`URI`
- datetime=`date/time value`

dfn

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

em

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

font (T)

- Core Attribute Set
- I18N Attribute Set
- size=`CDATA`
- color=`color`
- face=`CDATA`

i

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

ins

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- cite=`URI`
- datetime=`date/time value`

kbd

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

q

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- cite=`URI`

s (T)

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

samp

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

small

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

span (T)

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

strike (T)
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

strong
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

sub (T)
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

sup (T)
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

tt
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

u (T)
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

Images, Objects, and Applets

applet (T)
- Core Attribute Set
- codebase=`URI`
- archive=`CDATA`
- code=`CDATA`
- object=`CDATA`
- alt=`text`
- name=`NMTOKEN`
- width=`length`
- height=`length`
- align=`value`
- hspace=`pixels`
- vspace=`pixels`

area
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- shape=`shape`
- coords=`coordinates`
- href=`URI`
- nohref=nohref
- alt=`text`
- tabindex=`number`
- accesskey=`character`
- onfocus=`script`
- onblur=`script`
- target=`name` (T)

img

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- src=`URI`
- alt=`text`
- name=`NMTOKEN` (T)
- longdesc=`URI`
- height=`length`
- width=`length`
- usemap=`URI`
- ismap=`ismap`
- align=`value` (T)
- border=`length` (T)
- hspace=`pixels` (T)
- vspace=`pixels` (T)

map

- Event Attribute Set
- I18N Attribute Set
- id=`ID`
- class=`CDATA`
- style=`style sheet`
- title=`text`
- name=`CDATA`

object

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- declare=`declare`
- classid=`URI`
- codebase=`URI`
- data=`URI`
- type=`content type`
- codetype=`content type`
- archive=`URI List`
- standby=`text`
- height=`length`
- width=`length`
- usemap=`URI`
- name=`NMTOKEN`
- tabindex=`number`
- align=`value` (T)
- border=`pixels` (T)
- hspace=`pixels` (T)
- vspace=`pixels` (T)

param

- `id=`ID
- name=`CDATA`
- value=`CDATA`
- valuetype=(data|ref|object)
- type=`content type`

Form Elements

form
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- action=`URI`
- method=(get|post)
- name=`NMTOKEN` (T)
- enctype=`content type`
- onsubmit=`script`
- onreset=`script`
- accept=`content types`
- accept-charset=`character set codes`
- target=`name` (T)

label
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- for=`IDREF`
- accesskey=`character`
- onfocus=`script`
- onblur=`script`

input
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- type=`text`
- name=`cdata`
- value=`CDATA`
- checked=checked
- disabled=disabled
- readonly=readonly
- size=`CDATA`
- maxlength=`number`
- src=`URI`
- alt=`CDATA`
- usemap=`URI`
- tabindex=`number`
- accesskey=`character`
- onfocus=`script`
- onblur=`script`
- onselect=`script`
- onchange=`script`
- accept=`content types`
- align=`value` (T)

select
- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- name=`CDATA`
- size=`number`
- multiple=`multiple`
- disabled=disabled
- tabindex=`number`
- onfocus=`script`

- onblur=`script`
- onchange=`script`

optgroup

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- disabled=disabled
- label=`text`

option

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- selected=`selected`
- disabled=disabled
- label=`text`
- value=`CDATA`

textarea

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- name=`CDATA`
- rows=`number`
- cols=`number`
- disabled=disabled
- readonly=`readonly`
- tabindex=`number`
- accesskey=`character`
- onfocus=`script`
- onblur=`script`
- onselect=`script`
- onchange=`script`

fieldset

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

legend

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- accesskey=`character`
- align=`value` (T)

button

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- name=`CDATA`
- value=`CDATA`
- type=(button|submit|reset)
- disabled=disabled
- tabindex=`number`
- accesskey=`character`
- onfocus=`script`
- onblur=`script`

isindex (T)

- Core Attribute Set
- I18N Attribute Set
- prompt=`text`

Table Elements

table

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- summary=`text`
- width=`length`
- border=`pixels`
- frame=`value`
- rules=`value`
- cellspacing=`length`
- cellpadding=`length`
- align=`value` (T)
- bgcolor=`color` (T)

caption

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- align=`value` (T)

colgroup

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`

col

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`

thead

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`

- valign=`value`

tfoot

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`

tbody

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`

tr

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- span=`number`
- width=`length`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`
- bgcolor=`color`

th

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set
- abbr=`text`
- axis=`CDATA`
- headers=`IDREFS`
- scope=`value`
- rowspan=`number`
- colspan=`number`
- span=`number`
- width=`pixels`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`
- bgcolor=`color` (T)
- height=`pixels` (T)

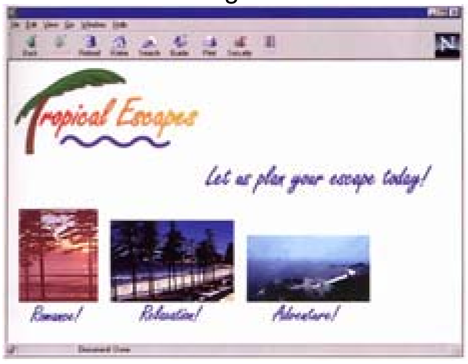
td

- Core Attribute Set
- Event Attribute Set
- I18N Attribute Set

- abbr=`text`
- axis=`CDATA`
- headers=`IDREFS`
- scope=`value`
- rowspan=`number`
- colspan=`number`
- span=`number`
- width=`pixels`
- align=`value`
- char=`character`
- charoff=`length`
- valign=`value`
- bgcolor=`color` (T)
- height=`pixels` (T)

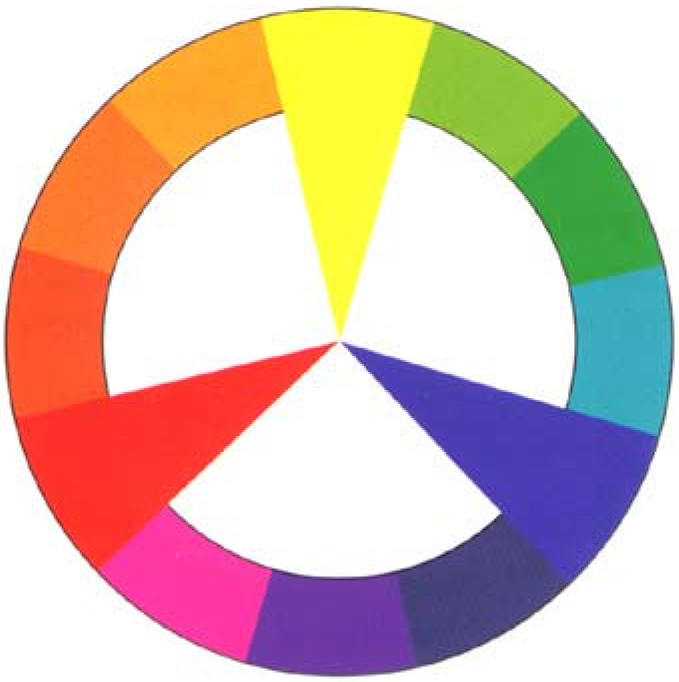
Color Section

In this section, I've selected images that bring you most benefit when you can see them in living color, versus the grayscale found elsewhere in the book. Some images you'll recognize from chapters you've already read. All the images demonstrate fundamental color pronciples and how color is rendered on the Web. By studying these pages, I hope to leave you with some fresh ideas, and a kick-start for your own colorful imagination.



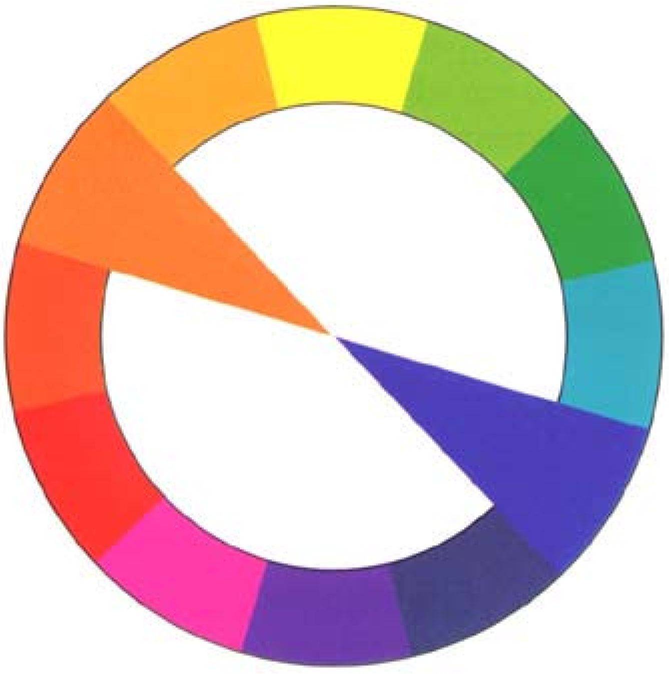
Establishing your theme with color

In this screen shot, the color used in the logo, a gradient from orange to red, bring to mind the hot colors of tropical flowers and sunsets and effectively establish the site's theme.



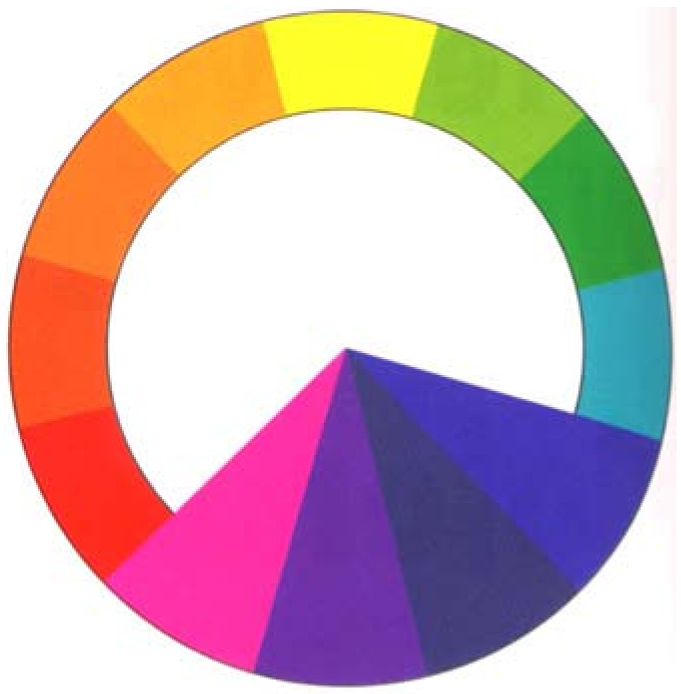
Primary Colors

The primary colors meet in the middle of the color wheel.



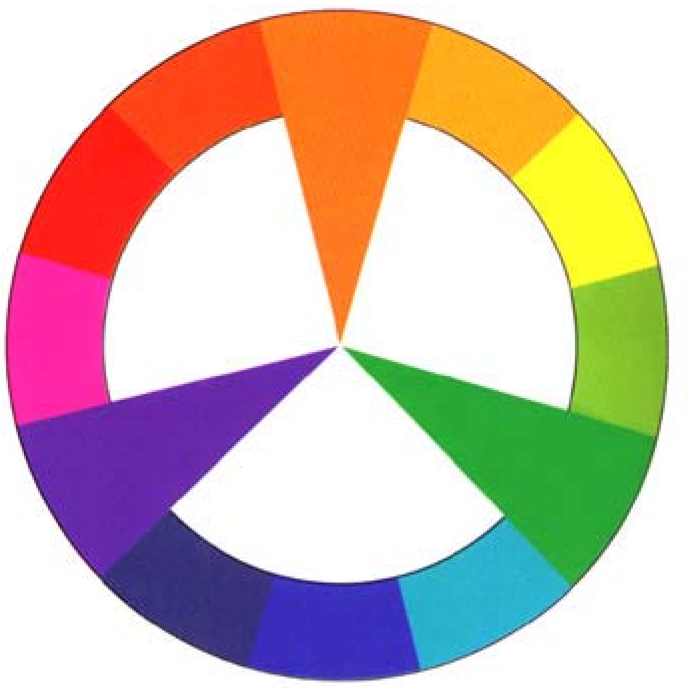
Complementary Colors

Complementary colors oppose each other on the color wheel. Notice that blue and orange are directly across the wheel from each other and meet in the middle.



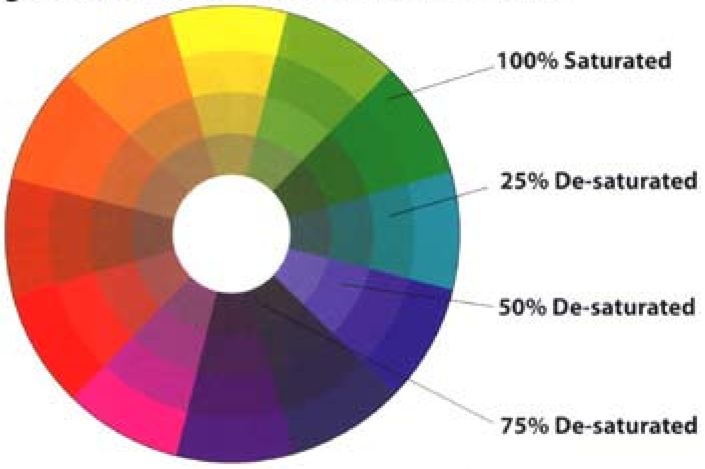
Adjacent Colors

Adjacent colors are easy for the novice designer to work with; find two or three colors beside each other, and you'll have a pleasing palette.

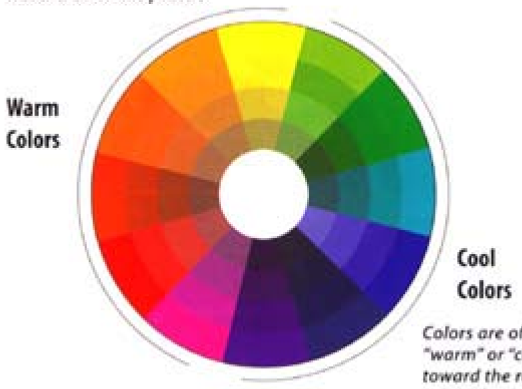


Triad Colors

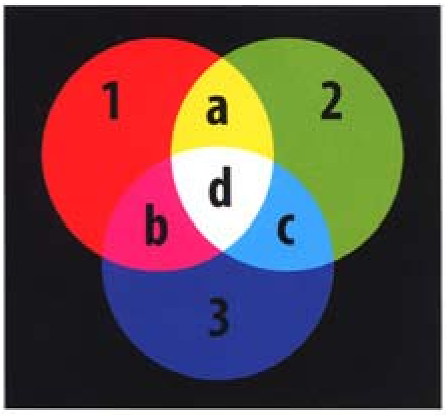
Triad color palettes are very popular, as they mimic the primary color separation. Red, yellow, and blue are a triad, as are orange, green, and violet seen in this image.



Changes in color saturation can be seen using a wheel. Fully saturated colors are bright and intense. Dusky or pale colors have very little color saturation and therefore look weathered or like pastels.



Colors are often described as being "warm" or "cool," based on a shift toward the red or blue side of the color wheel.



The RGB values are blended on your computer screen to produce the final displayed hue.

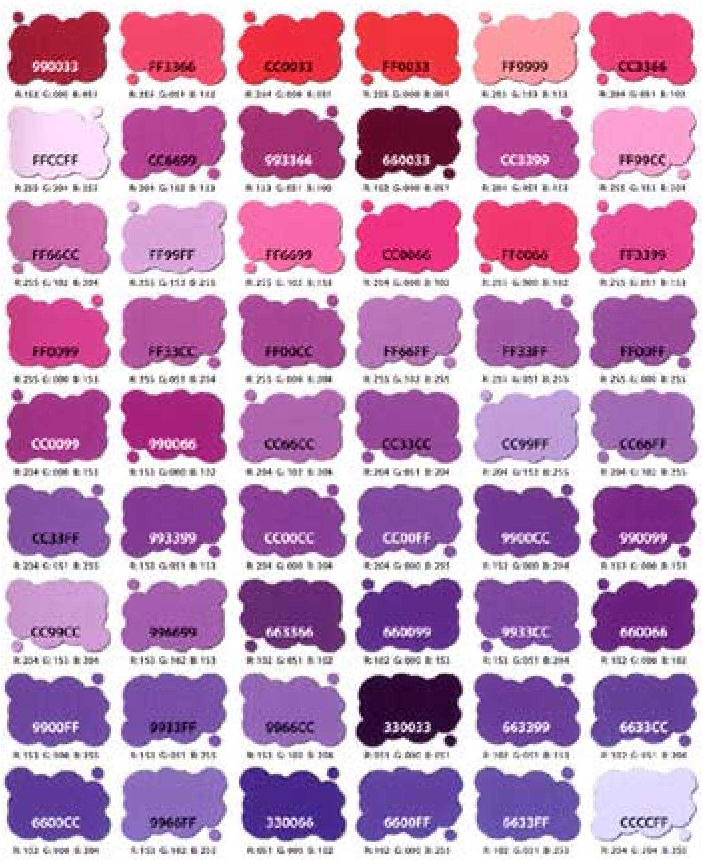
Blending RGB values

Three Different Scales that Measure Intensity

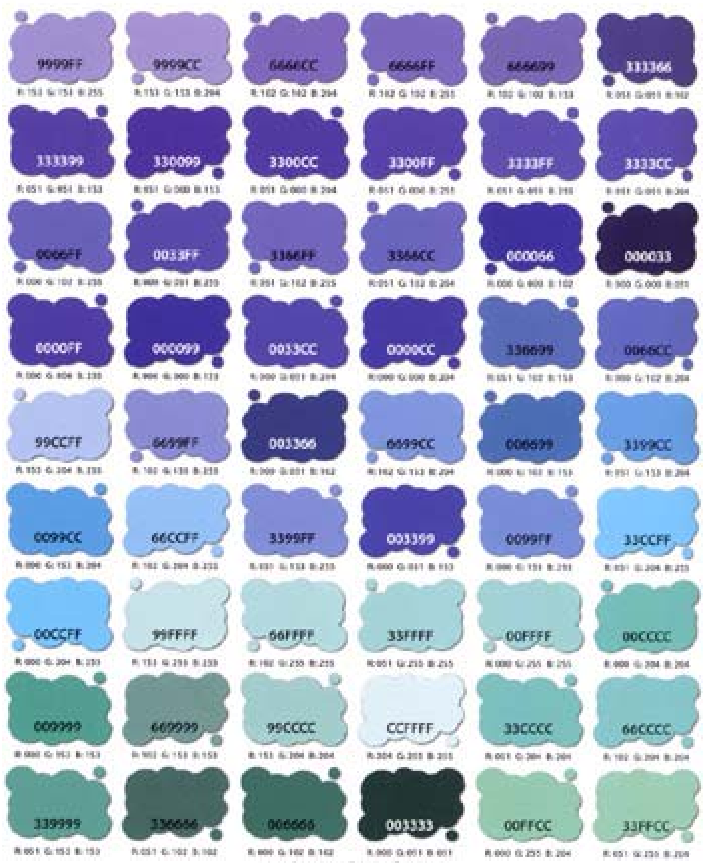
% Scale		256 Scale	Hex Scale
100%	🟦	255	FF
80%	🟦	204	CC
60%	🟦	153	99
40%	🟦	102	66
20%	⬛	51	33
0%	⬛	0	00

Color intensity can be measured in percentages of a fully saturated hue, a 0–255 scale [used in RGB values], or in the hex equivalent of 0-255.

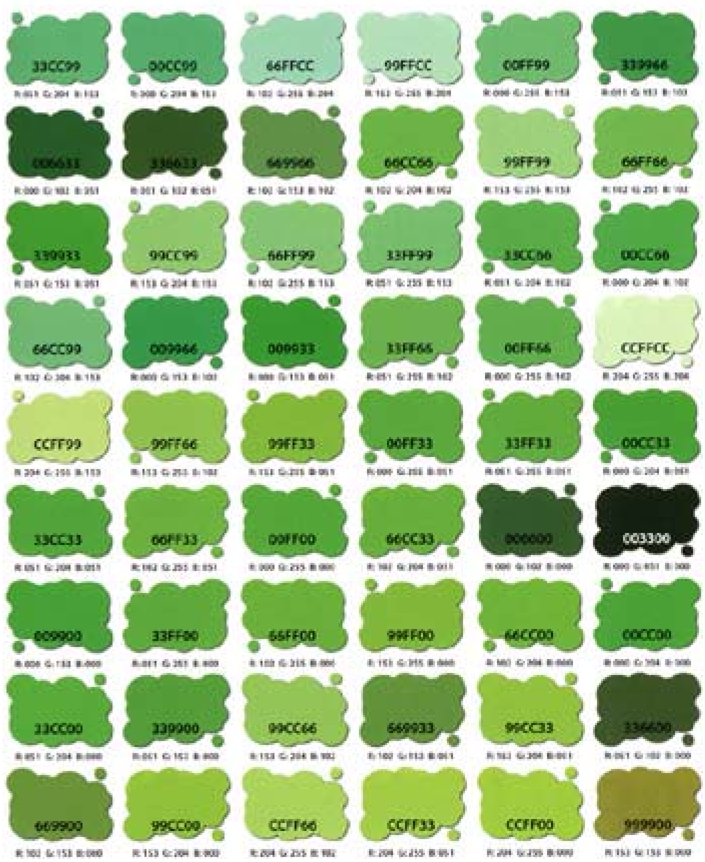
Color Safe Palette



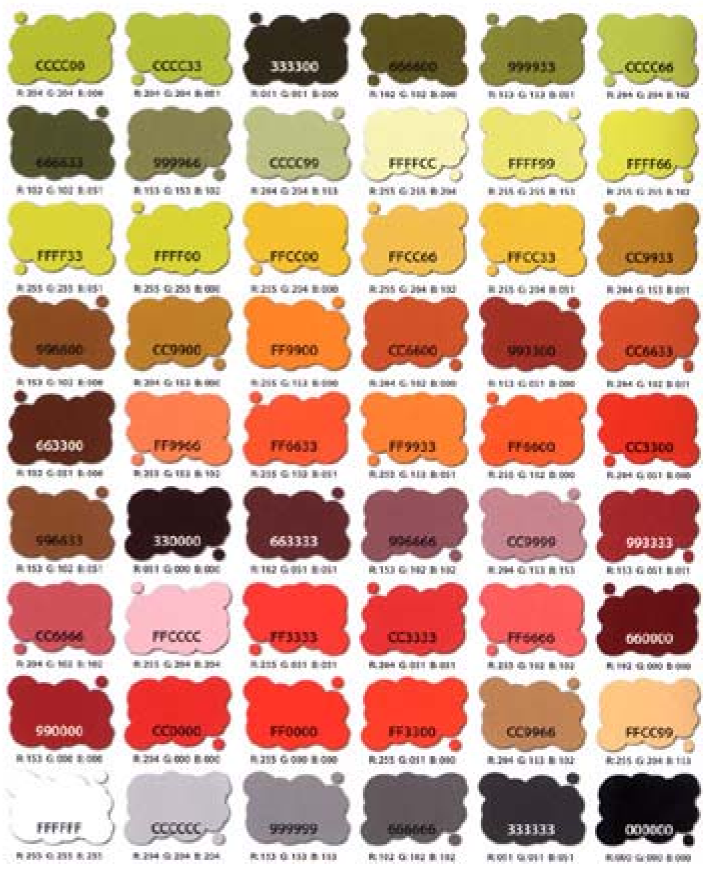
Color Safe Palette



Color Safe Palette



Color Safe Palette



EFFECTIVE WEB DESIGN

MASTER THE ESSENTIALS, BOTH NEW AND OLD

The trends and technologies behind Web design are shifting constantly. This fully revised new edition of *Effective Web Design* builds on the highly successful first edition to bring you up to speed on the latest tools, techniques, and thinking on Web site design. You'll make a strong start with XHTML, the latest and most powerful Web markup language. Then you'll learn to create pages that work flawlessly in all commonly used browsers, including the newest releases. You'll also discover how you can take advantage of progress in other technologies, from style sheets to Flash to MP3. The full-color insert helps you learn about the effective use of color on your site and provides a valuable reference guide you'll use again and again.

Ann Navarro's no-nonsense teaching approach has helped thousands of designers and developers build visually striking, functionally rich sites.

LEARN TO:

- Develop a sound architecture
- Find the right approach to navigation
- Design forms
- Use validation services and applications
- Present information effectively
- Get the right effect with colors
- Incorporate multimedia, including streaming content
- Implement e-commerce features
- Work with basic elements: tables, images, and frames
- Use cookies effectively

FEATURED ON THE CD:

The enclosed CD contains code referred to in the book, along with the latest versions of Netscape Navigator and Microsoft Internet Explorer. You also get shareware or evaluation copies of more than a dozen Web-development utilities, including an FTP program, code validator, graphics and text editors, and powerful XML tools.

Make the Leap from HTML to XHTML—and Be Ready for XM

Programmers: Gain a Designer's Eye

Designers: Gain a Programmer's Knowledge of Coding

Implement the Look You're After—and the Functionality Your Visitors Want

Ensure Your Site's Compatibility with All Browsers and Platforms

ABOUT THE AUTHOR:
ANN NAVARRO is founder and CEO of WebGeek, Inc., an Internet consulting and solutions provider established in 1996. She serves on the W3C HTML Working Group as an invited expert and at various industry conferences and events around the country. WebGeek, Inc. currently operates in the beautiful Florida Gulf Coast town of Port Charlotte.

User Level
All Levels

Book Type
How-to/Reference

Category
Internet Publishing

SYBEX®

US $39.99 CAN $59.95 UK £29.99

ISBN 0-7821-2849-1