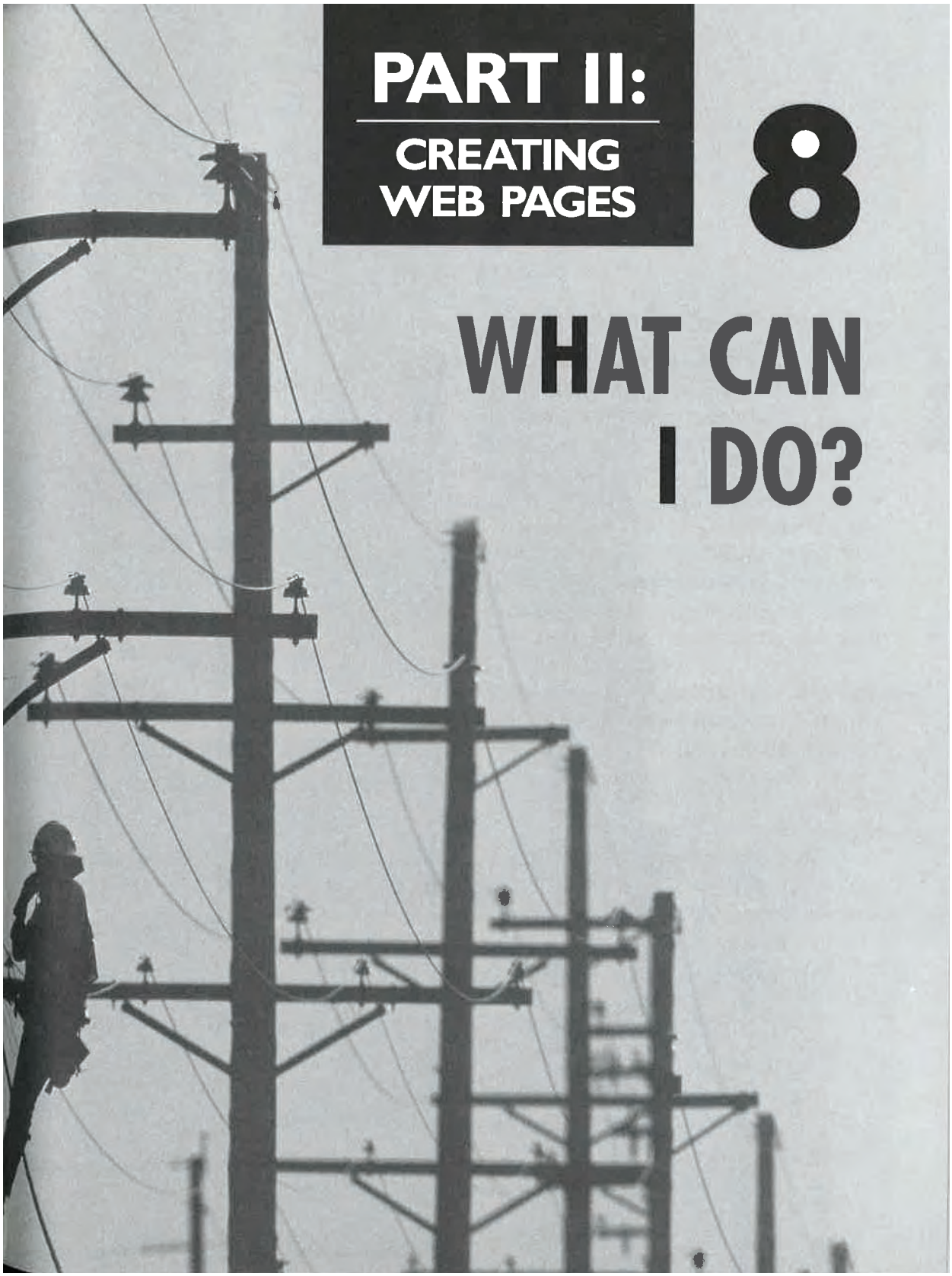
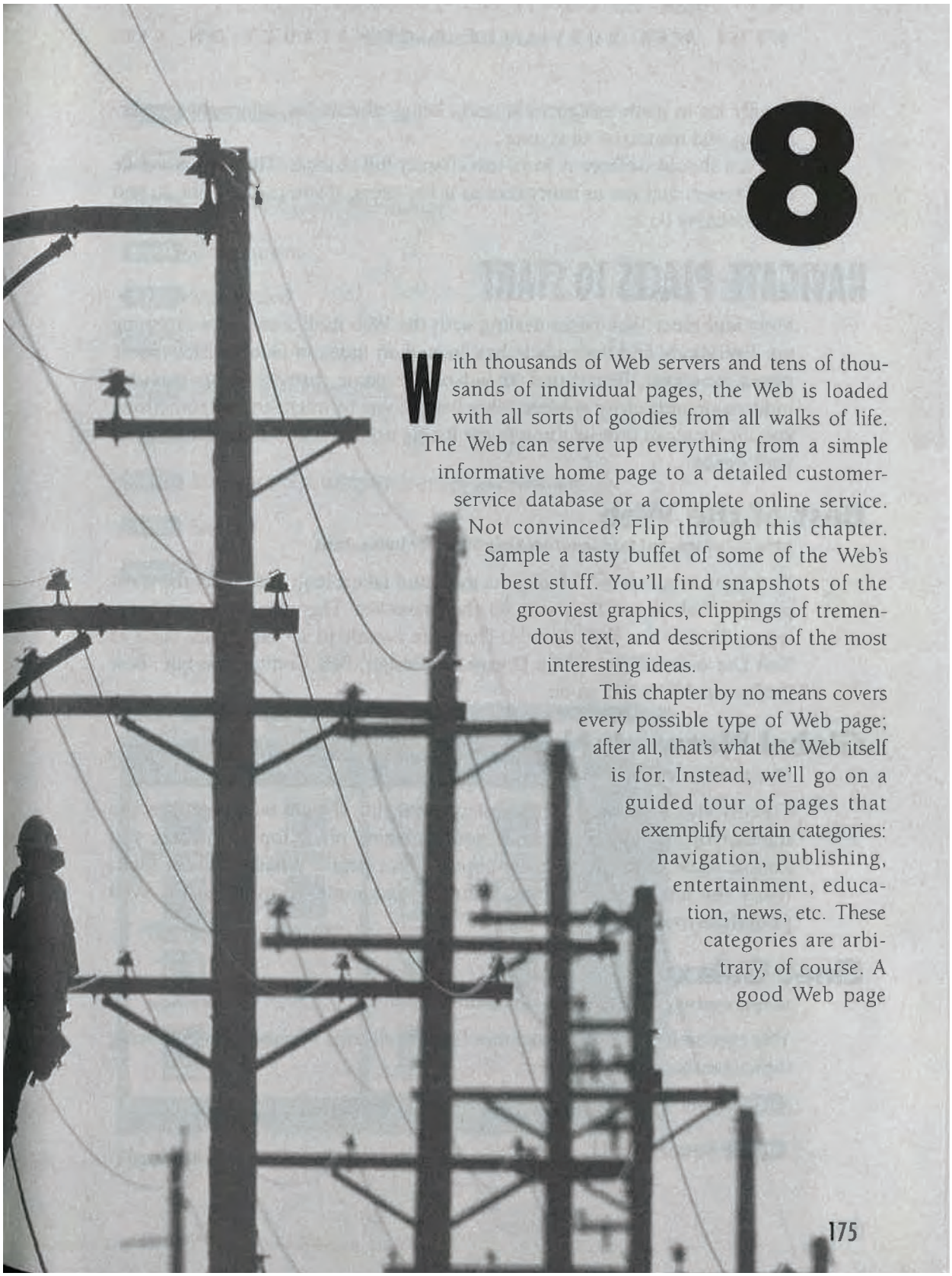


PART II: CREATING WEB PAGES

8

WHAT CAN I DO?



8

With thousands of Web servers and tens of thousands of individual pages, the Web is loaded with all sorts of goodies from all walks of life. The Web can serve up everything from a simple informative home page to a detailed customer-service database or a complete online service. Not convinced? Flip through this chapter. Sample a tasty buffet of some of the Web's best stuff. You'll find snapshots of the grooviest graphics, clippings of tremendous text, and descriptions of the most interesting ideas.

This chapter by no means covers every possible type of Web page; after all, that's what the Web itself is for. Instead, we'll go on a guided tour of pages that exemplify certain categories: navigation, publishing, entertainment, education, news, etc. These categories are arbitrary, of course. A good Web page

Google/YouTube Exhibit 1005/1105
Page 00201

usually fits in many categories at once, being educational, informative, entertaining, and interactive all at once.

You should walk away from this chapter full of ideas. The Web is a wide open venue, and has as many uses as it has users. If you can imagine it, you can probably do it.

NAVIGATE: PLACES TO START

More and more Web pages dealing with the Web itself seem to be cropping up. Everybody and their uncle has created an index of favorite sites. Some pages, however, have burst ahead of the game, providing spectacular indexes organized by subject. This way, if you're interested in something specific, you can browse through the listing until you find a page that meets your needs.

Best of the Web

`http://wings.buffalo.edu/contest/awards/index.html`

Each year, a bunch of Web experts meet and take a look at some of the most popular Web pages. They vote for their favorites. The winners receive prestigious Best of the Web awards. There are awards in 13 categories, such as Best Use of Interaction, Best Document Design, Best Commercial Site, Best Professional Site, and so on.

Global Network Navigator

`http://nearnet.gnn.com/gnn.html`

O'Reilly and Associates' Whole Internet Catalog (Figure 8-1) organizes the Internet by any subject or topic you can think of. A Top 25 list lets you sample some of the Web's most popular sites, and a What's New list highlights the latest Web ventures. There's also some information for Web publishers available at this site.

Einet Galaxy

`http://galaxy.einet.net/galaxy.html`

This comprehensive index includes links to all sorts of pages. The following topics are thoroughly covered:

- HTML Arts and Humanities
- HTML Business and Commerce

Google/YouTube Exhibit 1005/1105
Page 00202

- HTML Community
- HTML Computer
- HTML Engineering and Technology
- HTML Government
- HTML Intellectual
- HTML Law
- HTML Leisure and Recreation
- HTML Medicine
- HTML Reference and Interdisciplinary Information
- HTML Safety
- HTML Science
- HTML Social Sciences

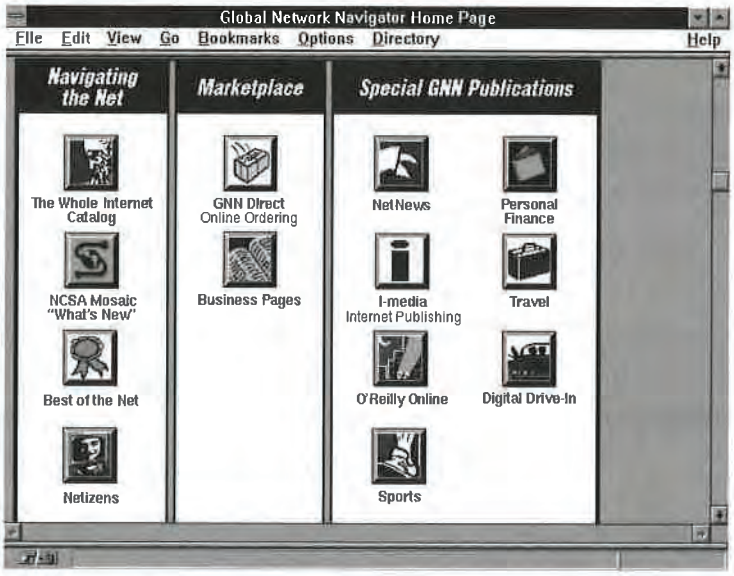


Figure 8-1 The Global Network Navigator

Google/YouTube Exhibit 1005/1105
Page 00203

Yahoo

http://akebono.stanford.edu/yahoo/

Twirl your lasso and shout a giant "Yahoo" for this mother of all Web indexes. This site, shown in Figure 8-2, contains an eclectic list of all sorts of pages. New sites are added daily, and a special New icon lets you check out the hottest sites as soon as they're developed.

Yahoo has hundreds of links to pages in all the following categories:

- HTML Art
- HTML Business
- HTML Computers
- HTML Economy
- HTML Education
- HTML Entertainment
- HTML Environment and Nature
- HTML Events
- HTML Government
- HTML Health
- HTML Humanities
- HTML Law
- HTML News
- HTML Politics
- HTML Reference
- HTML Regional Information
- HTML Science
- HTML Social Science
- HTML Society and Culture

Google/YouTube Exhibit 1005/1105
Page 00204

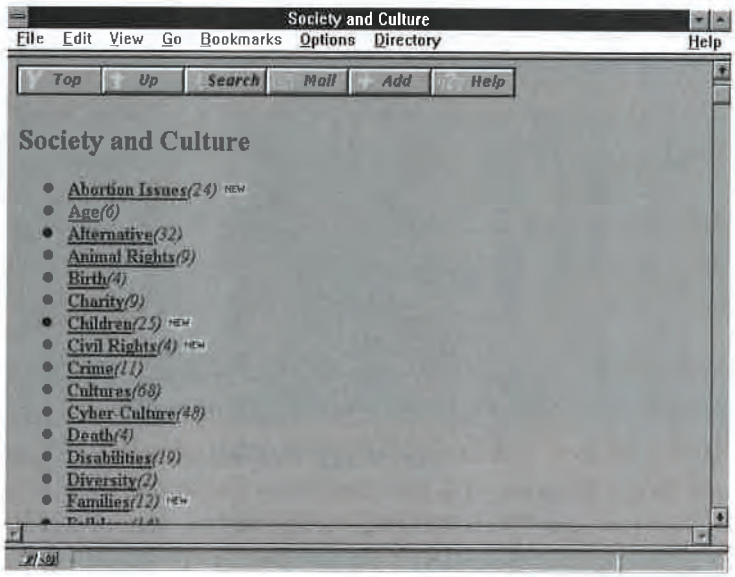


Figure 8-2 Yahoo

INFORM: TERRIFIC TEXT

Hypertext allows people to write in ways never seen before. One topic can lead to another, completely different one. An article can have deeper and deeper layers of detail. Stories can have multiple endings.

The following pages take text and use the Web to present that text in ways no book or other print medium could do. Such text is easy to read and easy to navigate using a hypertext Table of Contents, glossary, index, and such.

Manuals: NCSA Mosaic

`http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/Docs.html`

The Web is great for online reference books or manuals. A hyperlinked Table of Contents can lead to each topic, and cross references pop up at the click of a mouse.

For example, NCSA offers full online documentation for Mosaic. This neat idea lets you learn Mosaic while you use it. From any given page you can flip to the next page, review the previous page, turn to the beginning of the section, or view the entire Table of Contents.

Google/YouTube Exhibit 1005/1105
Page 00205

Helpful Information: OncoLink

`http://cancer.med.upenn.edu/`

Some of the Web's information can be downright essential. Created by E. Loren Buhle, the University of Pennsylvania Medical Center's OncoLink (Figure 8-3) ties together cancer information from around the world and puts it at your fingertips. Soak up the latest research. Or scroll to one of the many sections that help parents, teachers, families, patients, and health care workers deal with cancer.

A New Type of Text

`http://www-swiss.ai.mit.edu/samantha/travels-with-samantha.html`

Travels with Samantha is a full-length travel journal. Author Philip Greenspun toured North America with his companion Samantha and came away with lots of stories, quirky characters, strange sights, and over 250 snapshots. These photographs are scattered throughout, as in Figure 8-4.

These pages allow you to see a side of a country and a culture that you probably never knew existed.

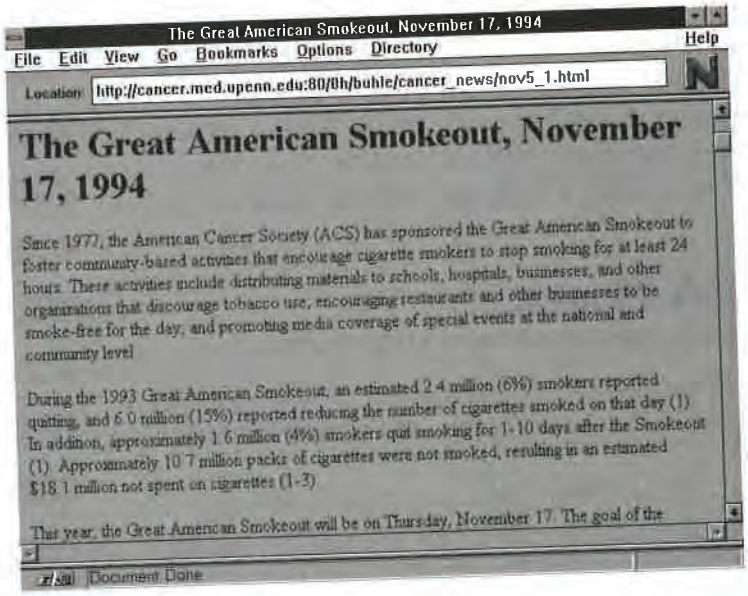


Figure 8-3 OncoLink

Google/YouTube Exhibit 1005/1105
Page 00206

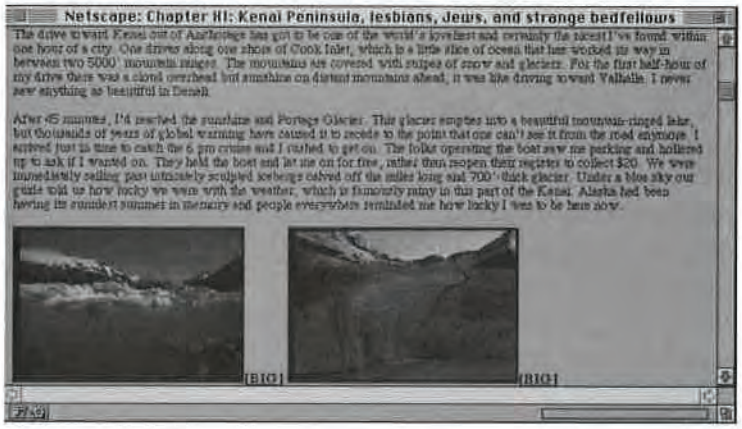


Figure 8-4 Travels with Samantha

HOME-MAKING: NO PLACE LIKE HOME PAGES

Home pages are the Web's front door to a company, organization, or person. They can serve as a home base, a starting point that leads to other related pages, a portfolio, or a sort of interactive business card.

Corporate Pages

It pays for businesses to make their presence known on the Web. Interested customers, associates, stockholders, and sellers can easily contact or read about your corporation—whether it's an international conglomerate or a company of one.

A typical corporate home page contains, or may lead to:

- HTML Corporate philosophy
- HTML Overview of products and services
- HTML Customer service or troubleshooting
- HTML Latest company news
- HTML Stock/bond reports
- HTML Research and development
- HTML Directory of employees

Google/YouTube Exhibit 1005/1105
Page 00207

IBM

http://www.ibm.com

Big Blue has scads of information about the latest software, hardware, and services. There are photos, sound clips, and plenty of text, as shown in Figure 8-5. Information about quarterly dividends, stock programs, and fourth quarter results is also included. There's even an area that tells you about IBM's latest technology and research. The index is searchable, so interested parties can find what they want, quickly.

Many of IBM's pages are laid out like a company newsletter or magazine, making them easy and fun to read.

Sun Microsystems

http://www.sun.com/

The Sun page (Figure 8-6) acts as a showroom for Sun's full line of computer systems. There's a section for Product Overview, Service and Support, Company Overview, Doing Business with Sun, Technology & Research, and Solutions.

Rather than just provide a list of systems and features, Sun's Web pages seek to give in-depth solutions to specific problems.

Figure 8-5 Hello from Lou Gerstner, chairman of IBM

Google/YouTube Exhibit 1005/1105
Page 00208

Figure 8-6 Sun Microsystems

University Home Pages

One of the biggest slices of the Internet pie belongs to educational institutions. Academics from around the world are among the Web's biggest customers. Many schools–indeed, many *departments*–have created their own home pages to guide, instruct, or bring together students, faculty, staff, prospective students, and researchers.

The typical university home page includes:

- HTML List of faculty, students, and staff
- HTML Course catalogs and schedules
- HTML Admissions info
- HTML Financial aid info
- HTML A map of the campus
- HTML Administration and services
- HTML Calendar of student life

Google/YouTube Exhibit 1005/1105
Page 00209

HTML Campus publications

HTML University library catalogs

HTML Other research resources

University of Illinois

`http://www.uiuc.edu/`

The University of Illinois home page has all the stuff you'd expect: People, class catalogs and schedules, University honors and awards, descriptions of current research, an online catalog of the university library, admissions info, financial aid, and campus policy manuals. There are also links to registered student organizations, job hunting areas, recreation, and info about the surrounding Champaign-Urbana community.

One neat section designed by a webspinner known as Ducky lets you walk around the university campus (see Figure 8-7). To get there, use the URL

`http://www.uiuc.edu/navigation/locator.html`

You can also get detailed maps of any University of Illinois campus, as well as of a specific building or even a room within a building.

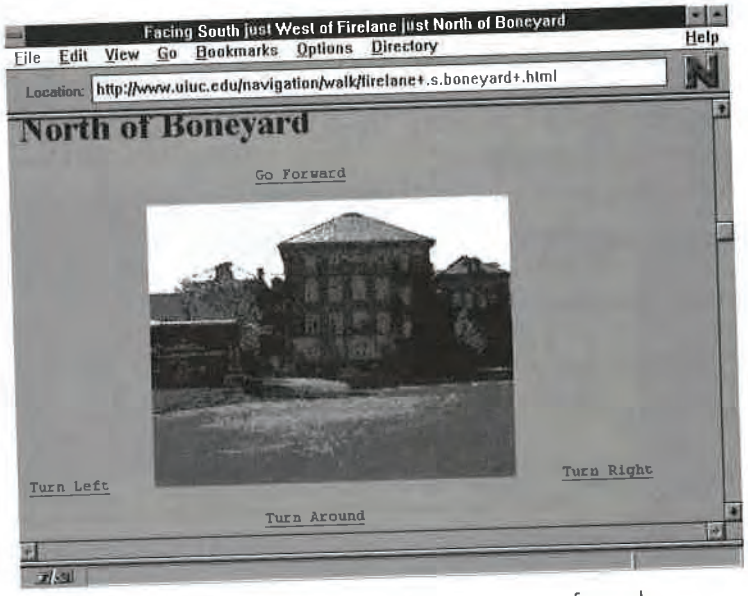


Figure 8-7 Walking around the UIUC campus from the comfort of your own home

Google/YouTube Exhibit 1005/1105
Page 00210

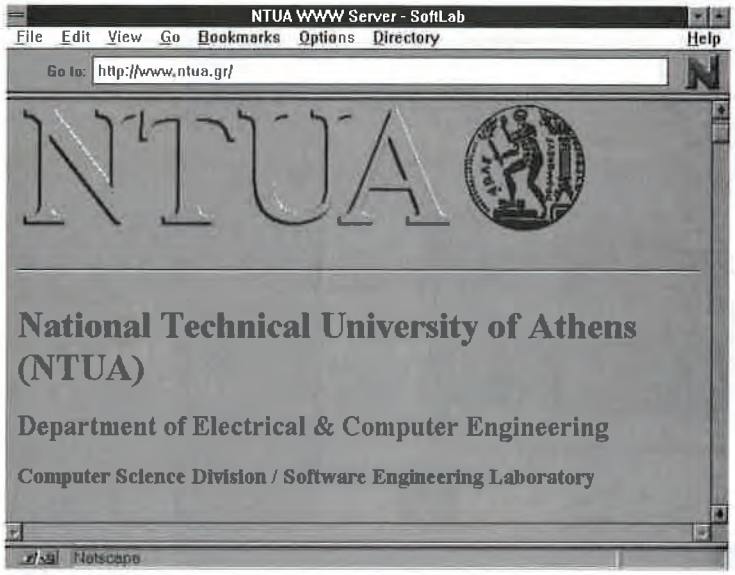


Figure 8-8 The Computer Science department at NTUA

National Technical University of Athens (NTUA)

`http://www.ntua.gr/`

Interested in studying abroad? Colleges from around the world have made their presence known on the Web. At the National Technical University of Athens (NTUA) home page (Figure 8-8), you can read all about the NTUA people and their research activities. You can pick up an application for the grad school, find out about local weather, interactively explore a Mandelbrot fractal, see a gallery of local artwork, and read some funny documents. A treasure of information about the Web, the Internet, and Unix is also available.

If you have the right fonts on your system, you can read some of these pages in their original Greek.

Government Home Pages

`http://www.whitehouse.gov/`

Most every government bureau, division, department, and office is beginning to realize the usefulness of Web pages. Huge stores of official information are available online. Perhaps the White House Home Page in Figure 8-9 typifies

Google/YouTube Exhibit 1005/1105
Page 00211

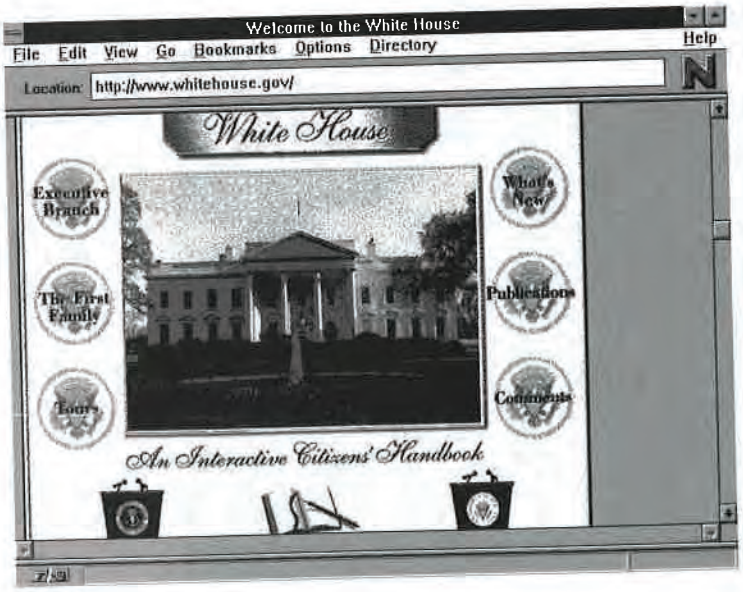


Figure 8-9 WWW White House

what a government home page should look like. It includes official addresses, tour information, an introduction and history of the White House, and a biography of the first family itself. This way, you can pay ol' Bill and Hillary (or whoever the current first couple happens to be) a visit without being frisked.

Personal Home Pages

You don't need to be famous, rich, or crazy to tell the world about yourself. Nearly anybody involved with the Web has created a personal home page. This page can be part résumé and part biography, part business and part pleasure. You can design a personal page as unique as yourself.

Personal home pages typically include a photograph or two, addresses and phone numbers, descriptions of current projects, educational or work background, some talk about your favorite hobbies or causes, links to your best friend's Web page, or any other quirky side of yourself you want to unleash upon the world.

Consultants, salespeople, or other freelancers often rely on Web pages to sell themselves. Many senators and other public servants are creating their own pages. These pages can be a great help around election time, when voters need all the help they can get to understand which issues a politician stands for and which he or she stomps on.

Google/YouTube Exhibit 1005/1105
Page 00212

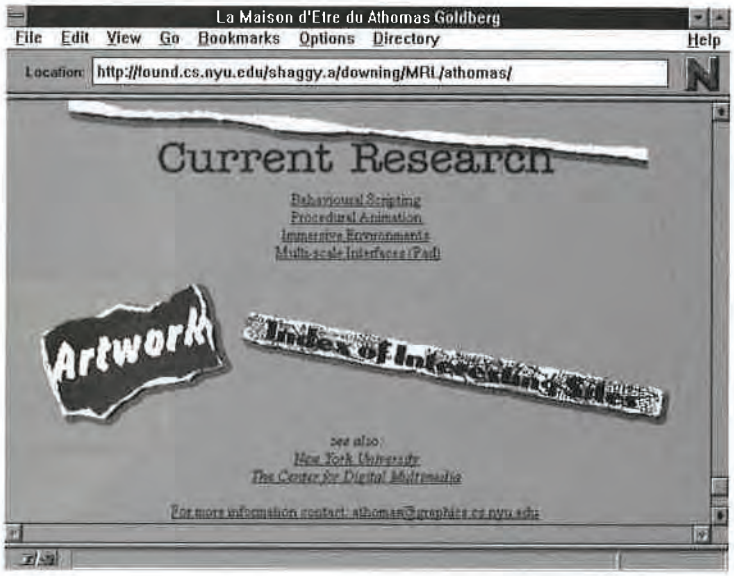


Figure 8-10 Athomas' Home Page

Athomas Goldberg

`http://found.cs.nyu.edu/athomas`

The Web page in Figure 8-10 is a sterling example of what an ultra-snazzy personal home page can look like. Design expert Athomas Goldberg's page includes links to papers he's written, projects he's been involved with, and organizations he works with. An online art gallery lets you scroll through Athomas' portfolio of oils, pastels, and even T-shirts he's designed.

David Fox

`http://found.cs.nyu.edu/dFox`

Why not? At the risk of being eponymous, Figure 8-11 shows one of your authors' own home pages. The design is simple but effective.

A Literary section talks about books and articles in press or in progress. It even includes a few sample short stories and poems to read through. An Art section lets you preview several drawings and paintings. A Computer section discusses latest research projects. And a Links section takes you to other interesting pages.

The full source code and images for this set of pages is included in the enclosed CD-ROM in the \HTML\DAVID directory. Chapters 11 and 12 use sections of this home page as examples.

Google/YouTube Exhibit 1005/1105
Page 00213

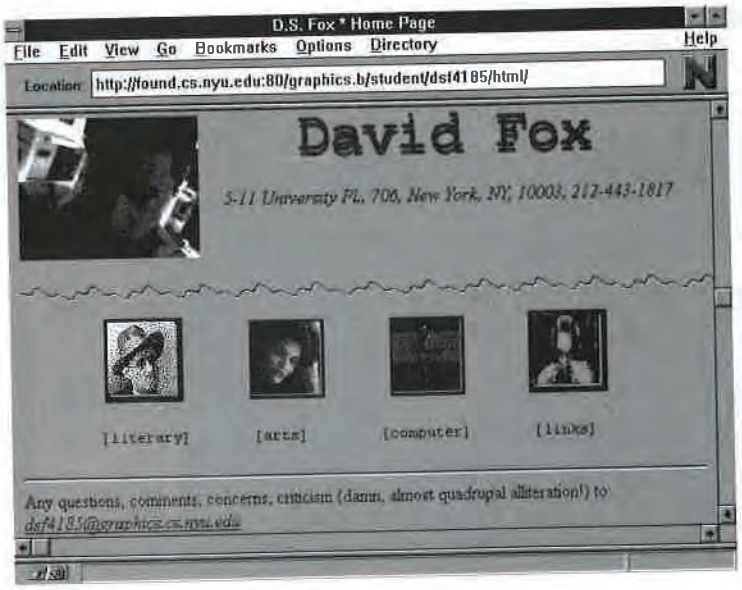


Figure 8-11 David Fox's Home Page

TEACH: EDUCATION

One of hypertext's greatest advantages is the vast amount of information that can be processed in a short period of time. This turns the Web into a fantastic learning tool. Kids (not to mention grown-ups) absolutely *love* clicking on pictures and text. By giving people some control over the information presented to them, studying becomes almost as much fun as playing video games. Almost.

Geography Skills

`http://ww.usgs.gov/education/teacher/what-do-maps-show/index.html`

The United States Geological Survey has a vast depository of cool information. One of the neatest pages in the Web is the What Do Maps Show section. This page consists of a huge "poster" consisting of several frames (see Figure 8-12). Each frame contains a different type of map—relief map, road map, aerial map, political map, etc. Clicking on each map tells you more about it. You can learn how maps are made, and which maps are good for which purposes. Parts of the poster can also be downloaded, printed, or viewed on a full screen.

Google/YouTube Exhibit 1005/1105
Page 00214

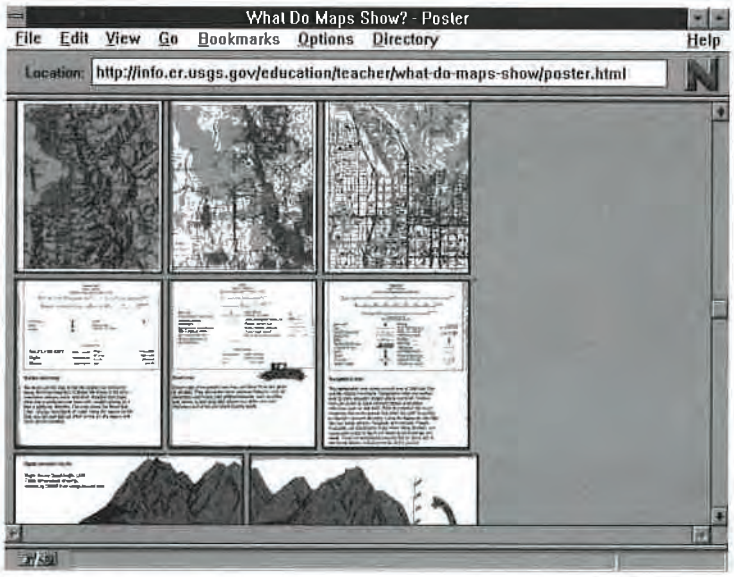


Figure 8-12 What do maps show?

ERIC

`http://ericir.syr.edu/Main.html`

The Educational Resources Information Center (ERIC) is a huge database of ideas, stories, lesson plans, educational technology, textbooks, literature, and other educational media.

The AskERIC section allows K-12 teachers, librarians, and administrators to ask any education-related question at all. ERIC experts will respond within two days with advice, a reference, or an exact answer.

Britannica Online

`http://www.eb.com/`

The world-famous *Encyclopædia Britannica* is now online–all 44 million words of it. The Britannica Online service also includes Merriam-Webster's Collegiate Dictionary and the Britannica Book of the Year. As of now, only schools or other institutions can subscribe to this service.

An online encyclopedia not only lets you look up words, but you can search the entire text for a particular term, hyperlink to related topics, and browse a vast index. Inlined images spice up the text with thousands of maps, flags, diagrams, and photos.

Google/YouTube Exhibit 1005/1105
Page 00215

ENTERTAIN: GROOVY GRAPHICS

The Web may be great for business and great for learning, but the reason the Web is booming so fast and so soon is because it's downright fun. The Web helps tie together entertainment, culture, and other important pastimes, bringing the whole world into your room.

And it's much more than text–text can be terrific, but images are what really suck the eyes toward Web pages. As a result, a great variety of artistic styles spice up the Net. Some pages are abstract, with strange images or symbols that lead to even stranger places. Others go for the minimalist look. Still others ... You've got to see them for yourself!

Here are a few places that make excellent use of visual elements.

TV: Interactive TV Index

`http://www.tvnet.com/ITVG/itvg.html`

Can't get enough of the tube? The TV Internet Resource Guide has everything you could need. There's an interactive TV index, reviews of the hottest new shows, and a TVPoll with surveys about your favorite shows. Links to other television sites let you find out the plots to your favorite soaps, join the Seinfeld fan club, or gossip about one of the *Star Trek* shows.

There's even an online TV guide.

Movies: Buena Vista

`http://bvp.wdp.com/BVPM/MooVPlex.html`

The Buena Vista Movies page (Figure 8-13) gives you access to reviews, stills, and clips from some of the most popular Disney and Touchstone movies. Sample a breathtaking scene from *The Lion King* or hear some dialog from the next boy-dog movie.

Music: Underground Music Archive

`http://www.iuma.com/`

This is one of the most in-depth music sites known to humankind (see Figure 8-14). If you want to take a break from Top 40, Pop, and MTV, check these pages out. There are dozens of labels and hundreds of bands. You can read about most bands, see their album cover art, and even hear bites from their best songs.

The *Addicted To Noise* online magazine gets down and dirty about the underground music scene and lets you know about the latest success

Google/YouTube Exhibit 1005/1105
Page 00216

Figure 8-13 Movies

Figure 8-14 Underground Music Archive

stories, concerts, and album releases. For more information about this page, contact info@iuma.com or phone 408-426-4862.

Google/YouTube Exhibit 1005/1105
Page 00217

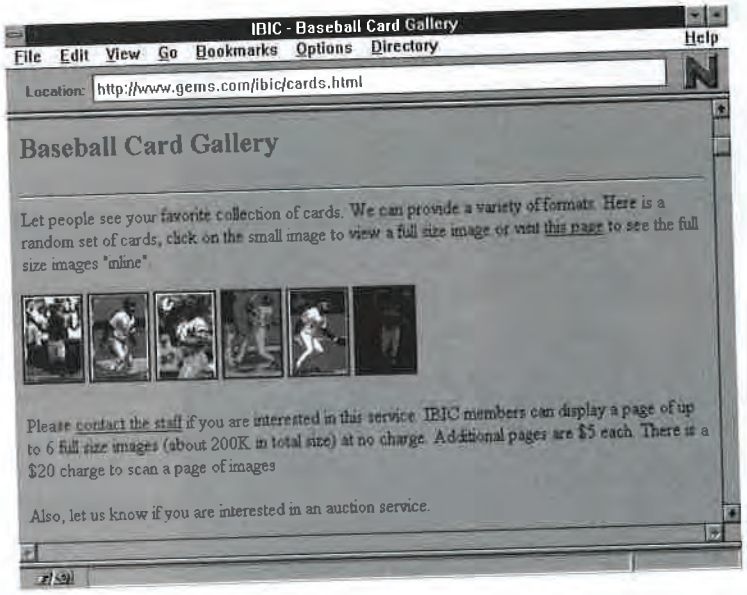


Figure 8-15 Baseball cards at the IBIC

Art: ArtSource

`http://www.uky.edu/Artsource/artsourcehome.html`

ArtSource is a gateway to all sorts of Web pages about art and architecture. You can find everything from pottery to modern art to Victorian houses. This site is growing constantly. ArtSource is selective about what it includes, and it links to some of the Web's best art-related sites.

Sports: Internet Baseball Information Center

`http://www.gems.com/ibic/`

If you click it, they will come. The Internet Baseball Information Center is a baseballer's dream. You can join the IBIC, access the full database of player statistics, view league reports, create your own fantasy league to compete against others, visit the Hall of Fame, or just shoot the baseball breeze.

An online baseball card gallery (Figure 8-15) lets you add to your virtual collection.

PUBLISH: PAPERS, BOOKS, MAGAZINES, AND MORE

Playboy, Penthouse, VIBE, Mother Jones, Saint Petersburg Press,—these are just a few of the more traditional print magazines and newspapers who've

Google/YouTube Exhibit 1005/1105
Page 00218

branched out to the Web. A bunch of literary journals and specialized 'zines have also been developed primarily for the Web.

From detailed technical manuals to experimental hyperpoetry, the Web can handle it.

Academic Papers

http://www.nta.no/telektronikk/

The Web is a great medium to publish your latest findings. Using different fonts, you can create clear papers with abstracts, summaries, and headings. Most articles have a hypertext Table of Contents leading to various sections, hypertext footnotes and references, and links to other notable sites. Some articles even offer inlined illustrations, photos, charts, and diagrams.

Norway's Telenor Research, for example, publishes the *Telektronikk* journal (Figure 8-16). This journal has a very professional, slick look to it. Articles deal with multimedia, networks, and telecommunications and can range from "SCREAM: Screen-based navigation in voice messages" to "Altruism and benefit in Cyberspace." Each article includes photos, hyperlinked Tables of Contents, and other supporting graphics.

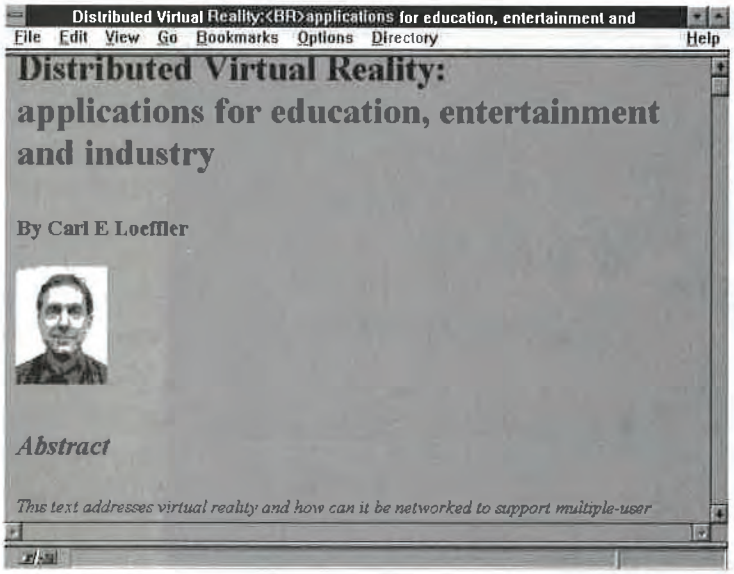


Figure 8-16 Telektronikk

Google/YouTube Exhibit 1005/1105
Page 00219

Multimedia Magazines

An online magazine can have departments, timely articles, editorials, advertisements, letters to the editor, and everything else magazines are known for. However, these magazines can also offer sounds, movies, software, and lots of links to related Web pages. An online magazine, then, nearly becomes its own online service.

VIBE

http://www.timeinc.com/vibe/

VIBE Online, shown in Figure 8-17, has the hip look and feel of its paper counterpart. There are profiles of recording artists, actual recordings of the latest music releases, and even an interactive Dotcom Bar, where people can chat, listen to music, and pick each other up. You'll also find lots of music-related information.

WIRED

http://www.hotwired.com/

WIRED Magazine's HotWIRED service, designed by Max Kisman, is one of the best-dressed places in cyberspace. (See Figure 8-18.) Much like *WIRED Magazine* itself, the fonts are bold and varied, the design is irreverent and

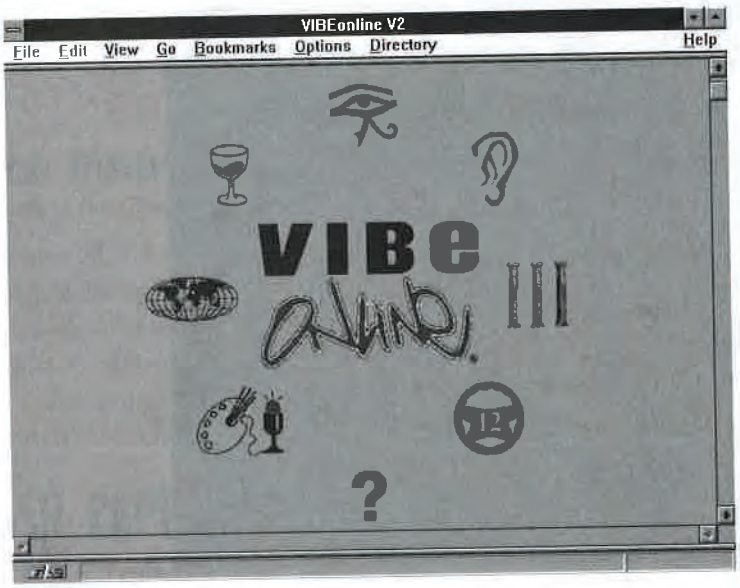


Figure 8-17 Some good vibes

Google/YouTube Exhibit 1005/1105
Page 00220

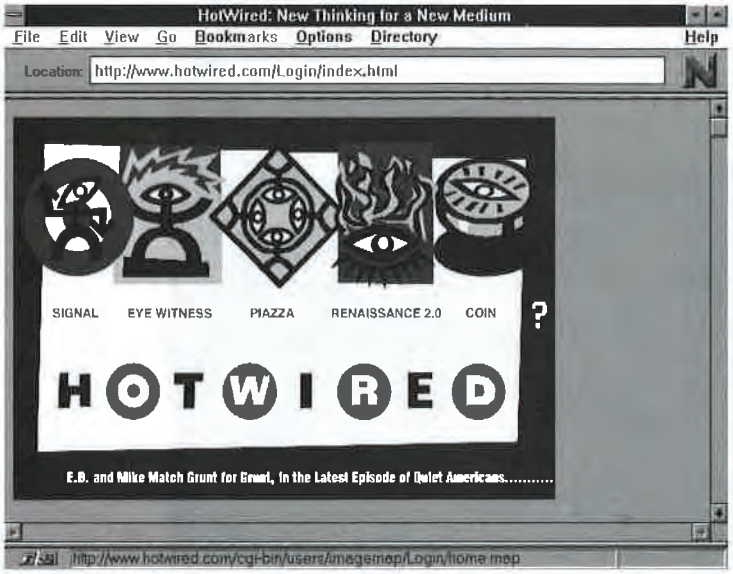


Figure 8-18 The HotWIRED service

flip, and the articles are fast-paced and hot. To "flip" through this magazine, you'll need to become a member (which, for now, is free). HotWIRED requires a username and password.

Once inside, you can read about the latest technology trends, gossip, etc. An ever-rotating selection of articles, stories, and other miscellany also awaits. A link to an MOO allows you to have real-time conversations in a text-based virtual reality. The Coin section has classified ads with some great personals, help wanted, and position wanted selections.

HotWIRED materials are copyright 1994-1995 by Wired Ventures, Ltd. All rights reserved. HotWIRED is a trademark of Wired Ventures, Ltd.

'Zines

`http://sunsite.unc.edu/shannon/ckind/title.html`

If you have a political bent, literary leer, or lascivious outlook, why not publish your own 'zine? In paper form, these 'zines are usually ratty black-and-white pamphlets. Online, however, you can use your full potential to include all sorts of graphics, fonts, lettering, and of course content.

Cyberkind, for example, features some of the Web's most interesting art, poetry, fiction, articles, and essays. The *twentyNothing* 'zine takes a hard Generation X look at life, death, sex, and other sundries.

Google/YouTube Exhibit 1005/1105
Page 00221

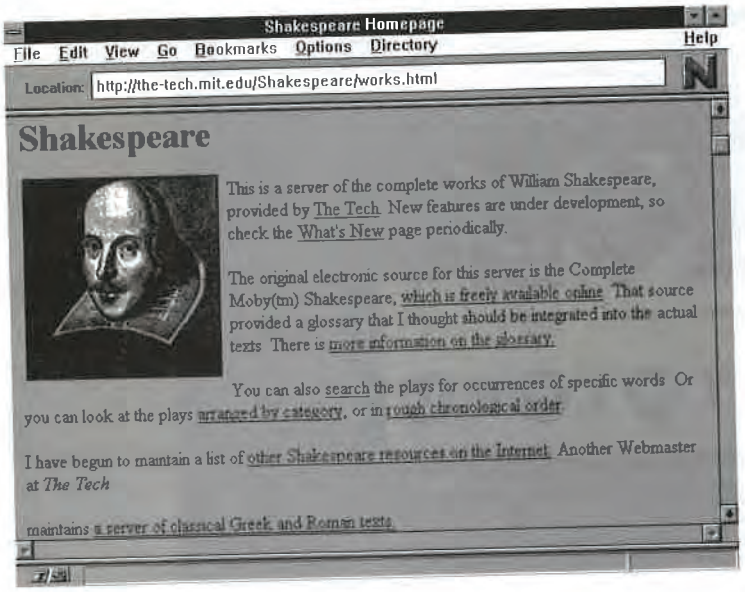


Figure 8-19 The Bard, caught asunder in thy Web

Books

`http://the-tech.mit.edu/Shakespeare.html`

More and more full-length books are appearing online. Whether your goal is to infiltrate, educate, or enlighten, online books can help spread the word. For example, the full works of Shakespeare are available on the Web (see Figure 8-19). You can search any play or sonnet for a particular word or scene or just begin reading.

SHOWCASE: PUTTING YOUR TALENTS ONLINE

`http://sunsite.unc.edu/Dave`

Whether you're a beginner or a pro, the Web is always a fitting place to strut your stuff. All sorts of artists, designers, and other folk have put the following online:

- HTML Art, sketches, paintings, cartoons
- HTML Fiction, poetry, screenplays, plays

Google/YouTube Exhibit 1005/1105
Page 00222

HTML Videos, film trailers, dramatic scenes

HTML Music, singing

HTML Resumes

HTML Demo programs

U2, Tori Amos, and Kate Bush are just a few of the singers who have official (or unofficial) home pages. Lewis Carroll, Isaac Asimov, Terry Pratchett, J.R.R. Tolkien, and Bram Stoker are just a few of the authors. And many up-and-coming artists have features. Even if you're not-so-famous, the Web can be a great place to showcase your talents. For example, Figure 8-20 shows Dave Farley's Doctor Fun comic, which appears every weekday. This single-panel comic has become a cult favorite.

Agents, casting directors, employers, producers, and art dealers are beginning to realize the potential and talent available over the Web. So should you.

Figure 8-20 Dave Farley's Doctor Fun

Google/YouTube Exhibit 1005/1105
Page 00223

SELL: TAKING CARE OF BUSINESS

Money makes the world go round, and the same can be said for the World Wide Web. A huge chunk of the Web's growth is due to large corporations and enterprising entrepreneurs hoping to turn the Web into tomorrow's most lucrative medium.

Some pages are no more than online display ads. These may or may not be interesting or fun to look at. Other financial pages, however, get much more in-depth.

Stock Quotes

`http://www.quote.com/`

QuoteCom's Web page, shown in Figure 8-21, provides the latest market data to Webbers. It offers lots of general information about the market, the stocks game, and company names. If you subscribe to the service, you can get even more detailed information—actual quotes, charts, a personal portfolio service, a PR newswire, a financial newsletter, a stock guide, and company stock histories and annual reports.

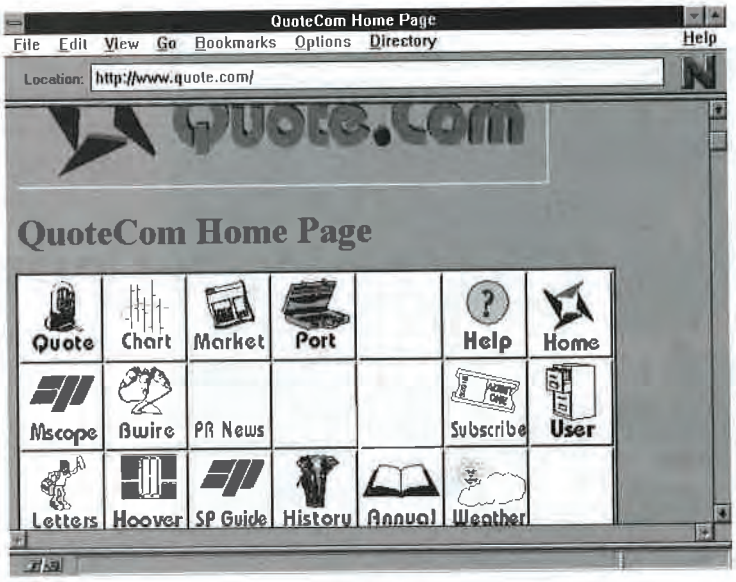


Figure 8-21 Locked in the stocks

Google/YouTube Exhibit 1005/1105
Page 00224

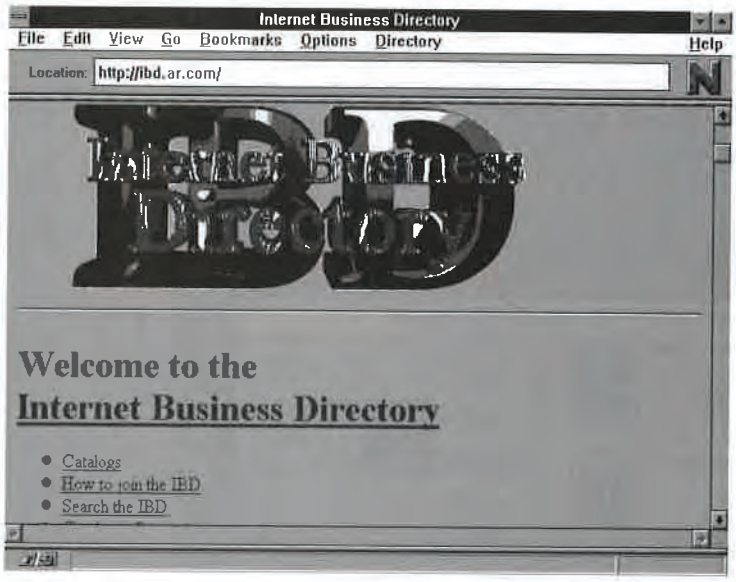


Figure 8-22 Internet Business Directory

Networking Over the Net

`http://ibd.ar.com/`

Many businesses have made a whole business of bringing other businesses together. The Internet Business Directory (Figure 8-22) contains online résumés and a full catalog of commercial Web sites.

Job Searching

`http://www.monster.com/home.html`

The Monster Board, shown in Figure 8-23, has all sorts of job-related data. There are company profiles, recruitment information for recent graduates, and a full set of employment classifieds.

BUY: SHOPPING

Ah, Web shopping. Unlike shopping malls, you can browse from the comfort of your home, any time of day, and with no hassles. Unlike traditional online shopping, you can see actual photos of whatever it is you want to buy, and read piles of detailed information. This atmosphere makes the Web a dream for shoppers and sellers alike.

Google/YouTube Exhibit 1005/1105
Page 00225

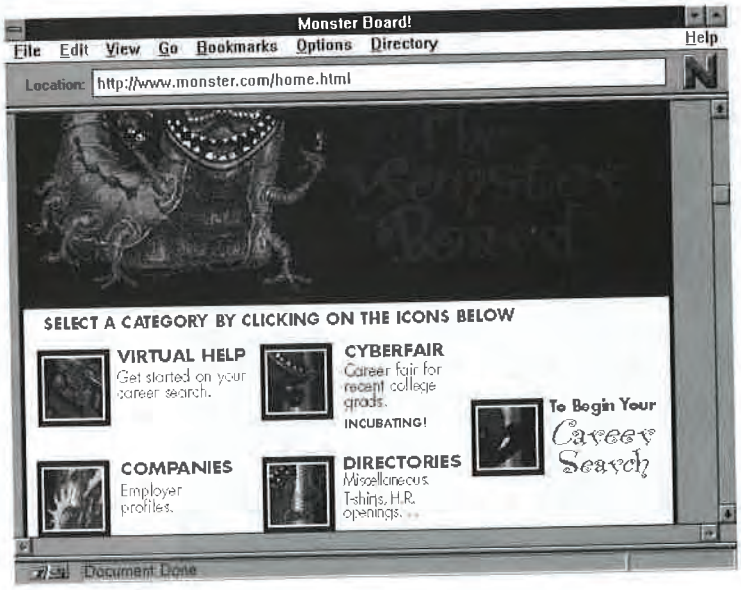


Figure 8-23 Facing the mighty Monster Board

The Internet Shopping Network

`http://shop.internet.net/`

Brought to you by the same people who run the Home Shopping Network, the Internet Shopping Network (ISN) is one of the world's biggest shopping malls. See Figure 8-24. You can browse through over 600 companies, order products, and have them delivered to you the next day.

Each company has its own look to it. And you can find everything from office supplies to flowers to cars waiting for you on the Web.

Direct Ordering

`http://www.pizzahut.com/`

If you're a small company, why not use the Web to hook in a few new customers? You can even offer direct marketing or ordering. The Pizza Hut Web page, for example (Figure 8-25) lets you order a pizza without picking up a phone. (As of now, this only works in parts of Southern California—the drivers won't go to Kansas for you....) This service is great if you get a sudden pizza attack while cruising the Web.

Google/YouTube Exhibit 1005/1105
Page 00226

Figure 8-24 A huge cybermall for you to spend your money in

Figure 8-25 Pizza Hut Online: May I take your order?

Google/YouTube Exhibit 1005/1105
Page 00227

SPEAK FREE: POLITICS

The Web can be used to spread more than information. Like all media, it attracts propagandists and activists. Lots of political viewpoints find homes on the Web.

PeaceNet

http://www.igc.apc.org/igc/pn.html

PeaceNet brings together social justice, human rights, and peace organizations from around the world. The latest news and information about justice, disarmament, health care, and other issues is supported.

FedWorld

http://www.fedworld.gov/

FedWorld, shown in Figure 8-26, is a vast depository of government stuff. There are links to Web servers, FTP archives, gopher menus, and telnet sites run by all sorts of U.S. government departments.

You can view information about topics from administration to ocean technology to the military to health care to industrial engineering to justice to urban and regional technology.

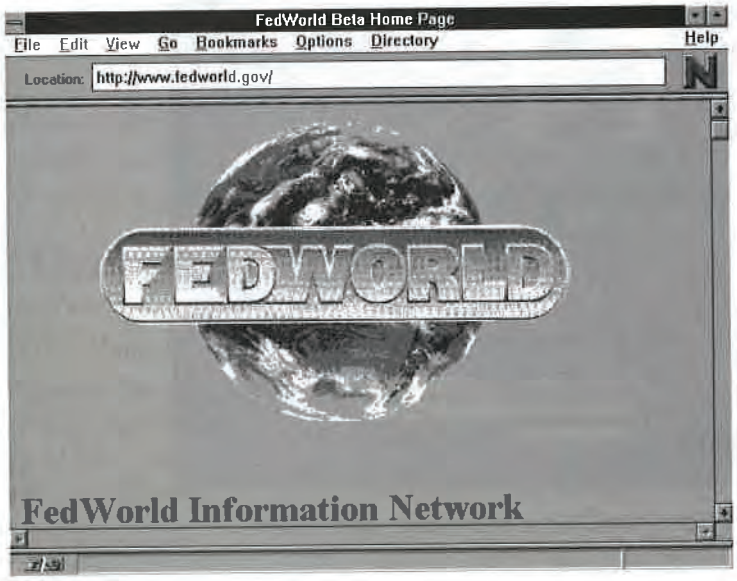


Figure 8-26 FedWorld Home Page

Google/YouTube Exhibit 1005/1105
Page 00228

An Interactive Citizens' Handbook lets you browse government Web servers by agency. The National Technical Information Service (NTIS) has tons of literature, reports, and databases about governmental science and technology. A Recent U.S. Government Reports section lets you read the latest studies and findings from every government agency you could think of.

OBSERVE: THE NEWS

Since Web publishing is so fast and easy, Web pages can be replaced daily or even hourly. This makes the Web a great place to announce the latest news. If your life depends on the distribution or acquisition of accurate (or at least hot) news, the Web will surely interest you.

Headline News

`http://www.cfn.cs.dal.ca/Media/TodaysNews/TodaysNews.html`

Much of *The Daily News*, a newspaper from Halifax, Nova Scotia, is available here (Figure 8-27). Summaries of the top local and national news stories, entertainment news, sports news, weather, and business news are included. The page even comes complete with a daily comic, "Mou's Cartoon."

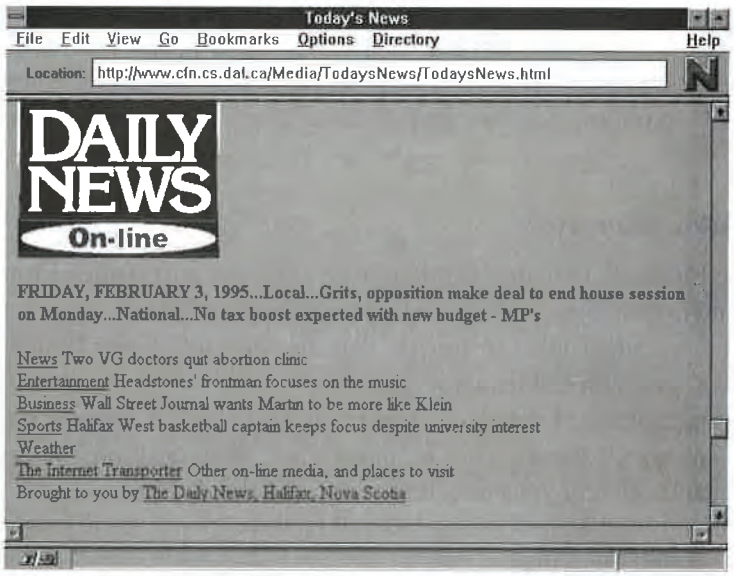


Figure 8-27 Today's News

Google/YouTube Exhibit 1005/1105
Page 00229

Weather

http://www.ucar.edu/metapage.html

Nice day today, isn't it? If you're not so sure, there are tons of instant weather stats and maps. One of the most versatile sites is the National Center for Atmospheric Research (NCAR) in Boulder, Colorado. There are also links to all sorts of meteorological resources and articles.

INTERACT: FILL-IN FORMS

Interactivity allows you to talk back to the Web. You can communicate with the Web by clicking a link, clicking an imagemap, or—more likely—filling in a form. There are tons of uses for forms, ranging from shipping orders to membership information to detailed surveys and polls. Some Web pages make full use of these forms, turning the Web into an intelligent responder.

Personal Ads

http://www.mall2000.com/date/date.html

One of the most exciting aspects of the online world is the ability to meet people, whether it be business associates, friends, or even lovers. The Waite Group's *Love Bytes: The Online Dating Handbook* talks all about the potential for cyberlove. The Web offers tons of romantic resources.

The Face to Face personals page, for example, allows you to write a detailed description of yourself. You can then search other people's ads and, if interested, write someone—or several someones—a little love letter. Some ads even have inline photos.

Databases

http://www.msstate.edu/Movies/

Cardiff's movie database contains details, facts, reviews, and names from tens of thousands of movies. It covers almost every popular movie in the world, ranging from silent films to movies that are currently being filmed. Figure 8-28 illustrates a typical database page.

The full filmgraphies of actors, directors, and other film moguls are included. You can search the database by name, date, title, position, credit, and more. You can even add your own information to the database.

To top it all off, there are a few snapshots of famous scenes, some poster art, and sound bites of great lines.

Google/YouTube Exhibit 1005/1105
Page 00230

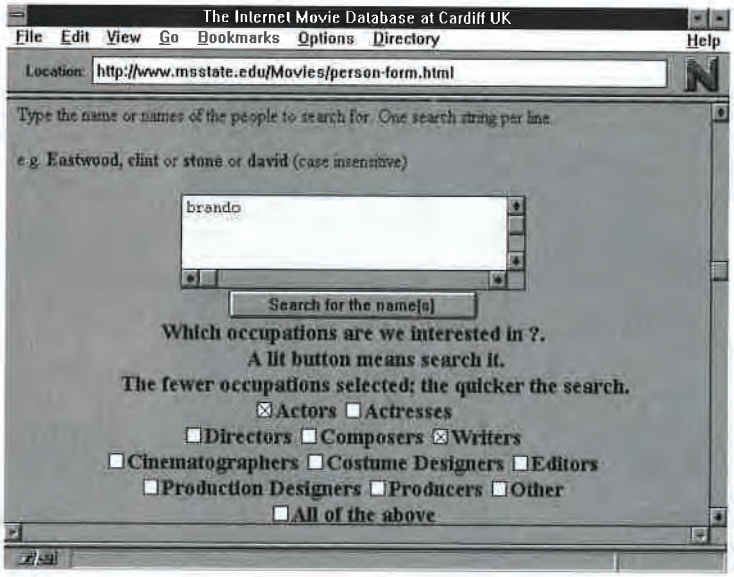


Figure 8-28 Cardiff's Movie Database

Security

The Netscape Commerce server, along with Netscape Navigator, offers transmission of secure fill-in forms. As more advanced Web browsers and servers come out, these secure forms will become much more usable and widespread.

FIND: MAPS

Since the Web can display any sort of graphic or text, it's a great resource for finding specific places. The Web sports many maps, ranging from building floor plans to the map of a campus to the map of the Milky Way.

Interactive Weather Map

`http://rs560.cl.msu.edu/weather/interactive.html`

The Interactive Weather Map server, shown in Figure 8-29, allows you to click on any state or city to view detailed satellite weather information.

Google/YouTube Exhibit 1005/1105
Page 00231

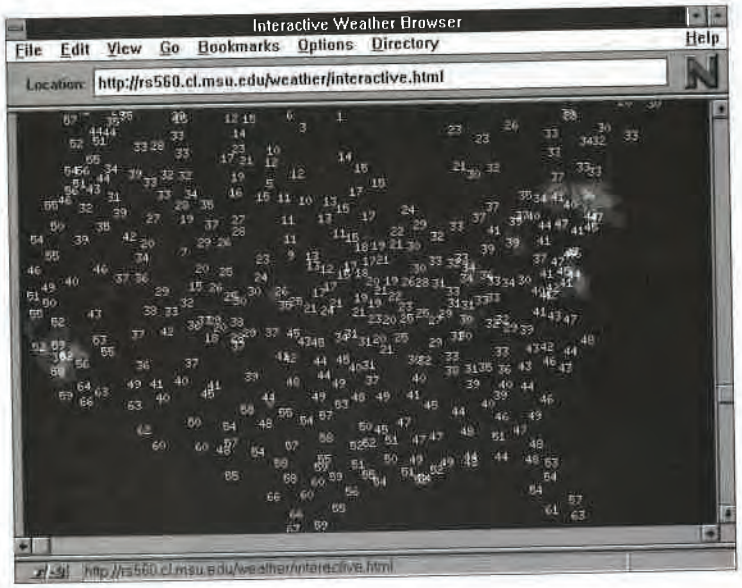


Figure 8-29 The Interactive Weather Map server

Xerox Map Server

`http://pubweb.parc.xerox.com/map`

The Map Viewer, courtesy of the Xerox Palo Alto Research Center, is one of the craftiest sites on the Web. It presents an image of the world, and you can zoom in on any point by entering a longitude and latitude. Or simply by clicking on the map, as shown in Figure 8-30. You can continue zooming until you're looking at the exact region of the globe you're interested in.

By clicking on various hyperlinks, you can adjust the look of the map. You can view borders, rivers, physical features, colors, and more. All in all, the Map Viewer allows you to design your own custom map image.

Map of Mars

`http://www.nasa.gov/nasa_online_education.html`

NASA's Mars Browser (Figure 8-31) is part of its vast educational resources. The browser is a full atlas of Mars' surface. You can rove the Mars surface by using arrow controls or picking a particular feature. There are also special color shots of famous Mars landmarks.

The NASA Web site also includes lots of multimedia presentations about topics such as space, the space shuttle, the history of the space

Google/YouTube Exhibit 1005/1105
Page 00232

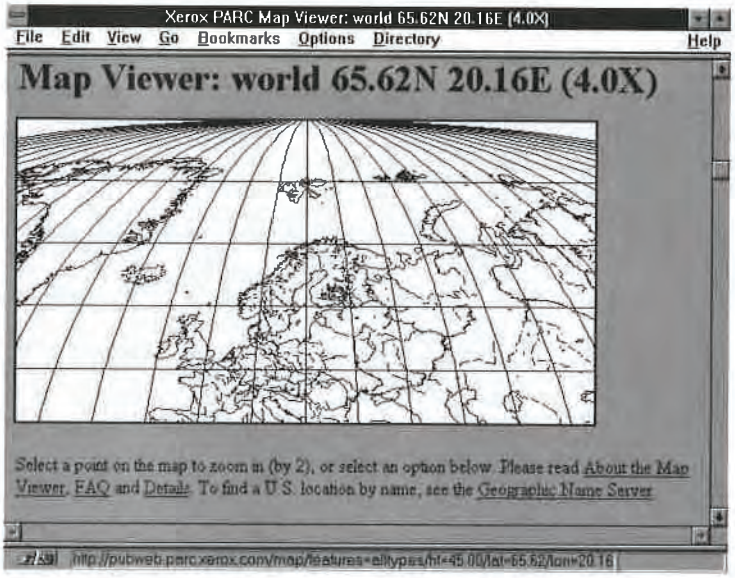


Figure 8-30 PARC Web Map Viewer

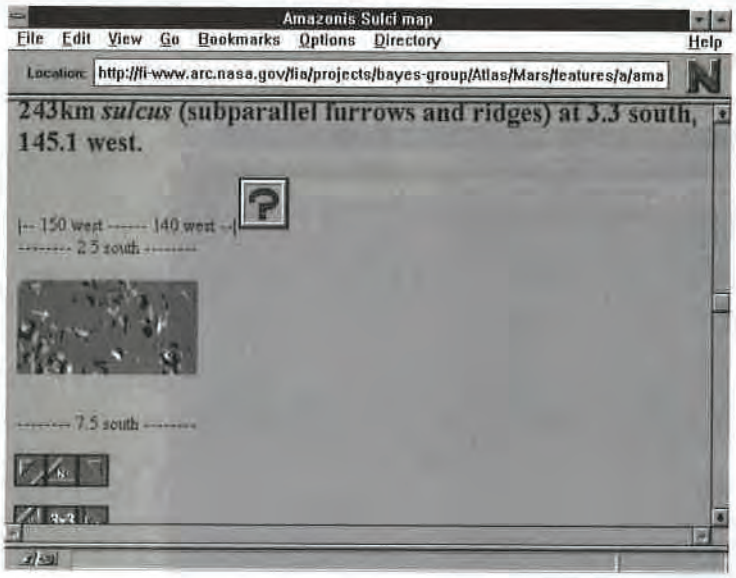


Figure 8-31 Mars Browser

program, space colonies, remote sensing, radio waves, the Hubble telescope, ocean color from space, and astronomy. A vast photo gallery lets you browse through tons of space-related images and video clips.

Google/YouTube Exhibit 1005/1105
Page 00233

TOUR: MUSEUMS

Museums are places that present interesting materials, artifacts, exhibits, or artwork to the world. The Web can be thought of, then, as one huge museum. Some sites, however, have really gotten the museum look and feel down pat.

Le WebLouvre

http://mistral.enst.fr/

Designer Nicolas Pioch created this sleek series of Web pages. Starting at the grand Le Pyramide ticket booth, tour the Louvre museum. Along the way, you can learn about artistic movements such as cubism, surrealism, impressionism, dadaism, and modernism. Read biographies of famous artists and sample one or two of their paintings, as shown in Figure 8-32.

You can also take a leisurely stroll around the Louvre palace's beautiful grounds. If you get bored with that, Le WebLouvre even lets you take a quick trolley around Paris. Stroll the Seine, scale the Eiffel Tower, and gaze at the Arc de Triomphe.

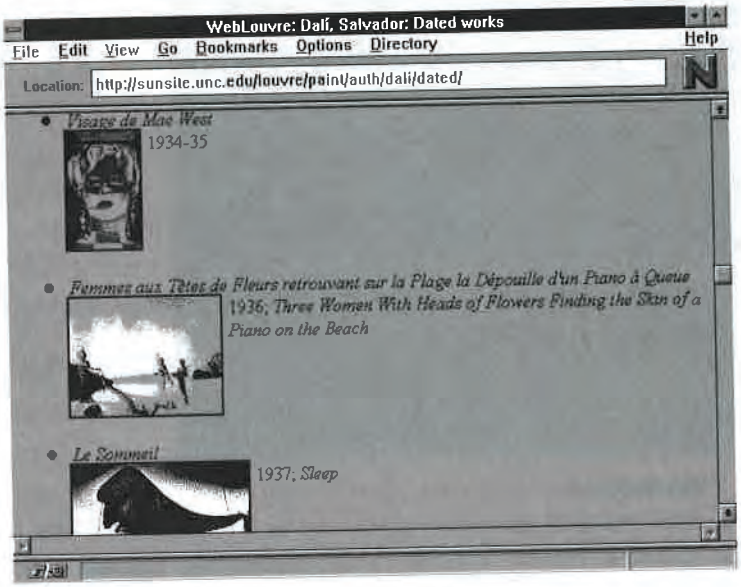


Figure 8-32 The *pièce de résistance:* Le WebLouvre

Google/YouTube Exhibit 1005/1105
Page 00234

Figure 8-33 Exploratorium Home Page

Exploratorium

`http://www.exploratorium.edu/`

The Exploratorium is a real museum in California dedicated to interactivity, multimedia, and technology. Its home page (shown in Figure 8-33) opens the door to tons of online exhibits showing you the best interactive art, science, education, virtual realities, and commerce. You can also flip through the library of scientific images and sounds.

If you wish, you can find out about the Exploratorium itself or read about exhibits and tours. If you really want to make your museum trip authentic, you can even shop at the Exploratorium Store.

Museum of Paleontology

`http://ucmp1.berkeley.edu/`

The University of California at Berkeley's Museum of Paleontology has tons of photographs and text about fossils, bones, and other real old stuff (Figure 8-34). This includes plants, protists, invertebrates, and of course our good friends the vertebrates. You can also learn all about paleontology itself and read about the latest research.

Google/YouTube Exhibit 1005/1105
Page 00235

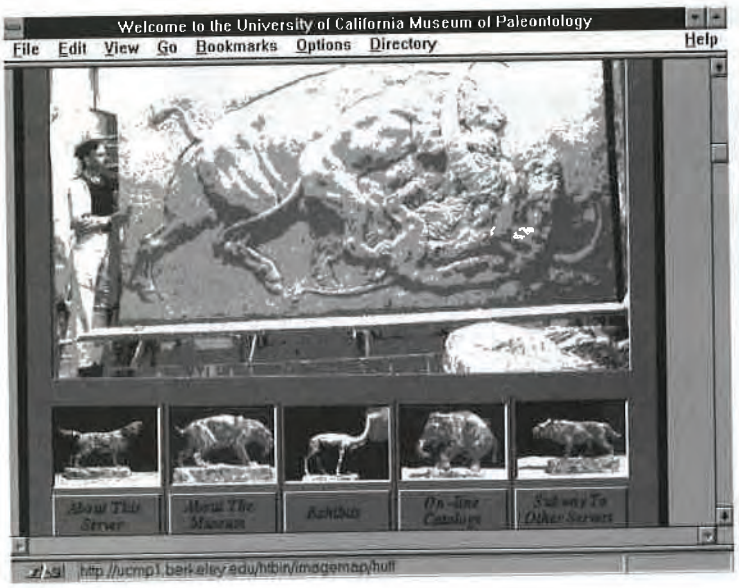


Figure 8-34 The Main Menu at the U.C. Berkeley Museum of Paleontology

You can view each exhibit by phylogeny (what it looks like) or geology. Learn about evolutionary thought or follow a species family tree. There's even a geological time machine that lets you bring the Jurassic and other epochs to your Web browser—much safer than *Jurassic Park!*

VISIT: VIRTUAL PLACES

Being worldwide, the Web can bring remote parts of this Earth to your own home. As a Web publisher, you can create entire worlds—real or fictional—for others to romp around in. If you have a tourist bureau or travel agency, or live or work someplace interesting to visit, why not duplicate the look and feel of your surroundings?

San Diego

`http://white.nosc.mil/sandiego.html`

This immense Web server tells you anything and everything you could possibly want to know about San Diego (see Figure 8-35). There are tons of maps, photos, and lists. News, finance, government, lifestyles, real estate, jobs, law, weather—it's all in here. This site is a godsend for tourists and residents alike.

Google/YouTube Exhibit 1005/1105
Page 00236

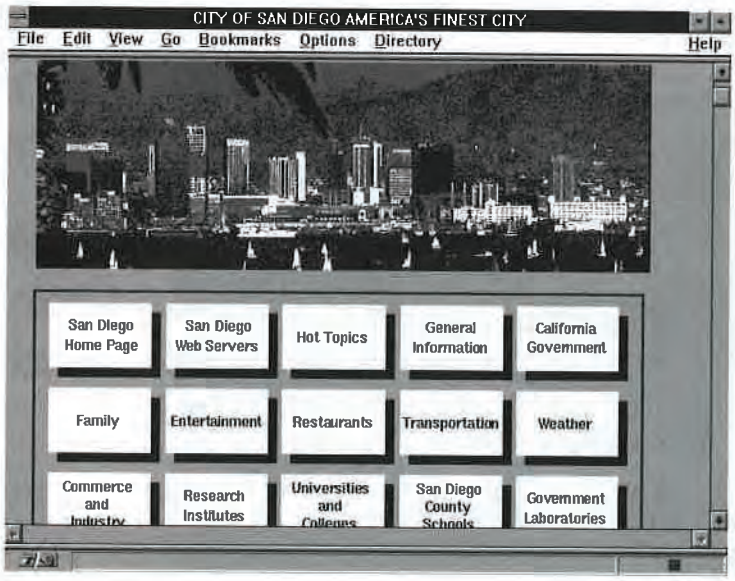


Figure 8-35 San Diego Home Page

You can easily find a cool place to tour, a hotel, an apartment, a gallery, a company, a restaurant, a TV or radio station, a park, a library, or a place to shop. The public transportation area shows you how to get where you want to go. Find out about sporting events (those poor Chargers), conventions, concerts, or special festivals.

This site also includes links to many other San-Diego-related pages.

Paris

`http://meteora.ucsd.edu/~norman/paris/`

Snap on your beret and visit the city of lights, shown in Figure 8-36. This site includes a Paris Visitor's Center, where you can get your bearings. You can then take the Metro, RER, or Bus to one of the monuments, museums, expositions, stores, or cafés. An interactive map lets you click on a location to automatically travel there.

If you want a more subdued tour, flip through the beautiful gallery of Paris scenes. You can also get tons of historical, anecdotal, and trivial information about Paris.

Google/YouTube Exhibit 1005/1105
Page 00237

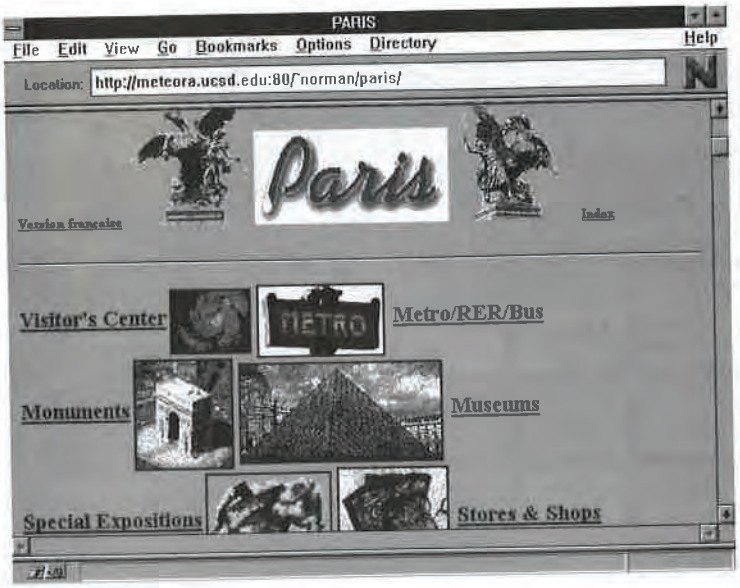


Figure 8-36 Vive Paris

EXPO Ticket Office

`http://sunsite.unc.edu/expo/expo/busstation.html`

The EXPO Ticket Office is an exhibition of some of the best sites on the Web. Click on the map of famous sites to decide where you want to visit. Grab your free ticket and take a shuttle bus to one of the following Web-wide exhibits:

- HTML The Vatican Exhibit: Many of the Vatican Library's most valuable books, documents, and maps.
- HTML Soviet Archive Exhibit: Read about the espionage, thrills, and hidden side of the Soviet's now-defunct Communist government.
- HTML 1492 Exhibit: Read about conquerors and explorers who braved the New World from 1492 to 1600.
- HTML Dead Sea Scrolls Exhibit: View the actual 2,000-year-old Dead Sea Scroll manuscripts and learn about who may have written them.
- HTML Fossil Life Exhibit: The Museum of Paleontology's exhibit about evolution and geology.

Google/YouTube Exhibit 1005/1105
Page 00238

HTML Spalato Exhibit: Learn about the ancient Roman city of Spalato.

HTML The Krannert Art Museum Exhibit: Sample the museum's works.

HTML Dinosaur Exhibit: A look at the history and fossils of some Jurassic giants.

HTML History of Prints: Learn all about printmaking.

HTML Ansel Adams Exhibit: Photographs by the famous nature artist.

HTML Mount Wilson Observatory Exhibit: Take a virtual reality walking tour of this observatory.

Downtown Anywhere

http://www.awa.com/

Downtown Anywhere (Figure 8-37) is one of the most realistic (and capitalistic) virtual places in the world. You can buy real estate space on the Internet. You can do what you want with your space—create your own Web pages (that would be my suggestion) or just keep it as an investment.

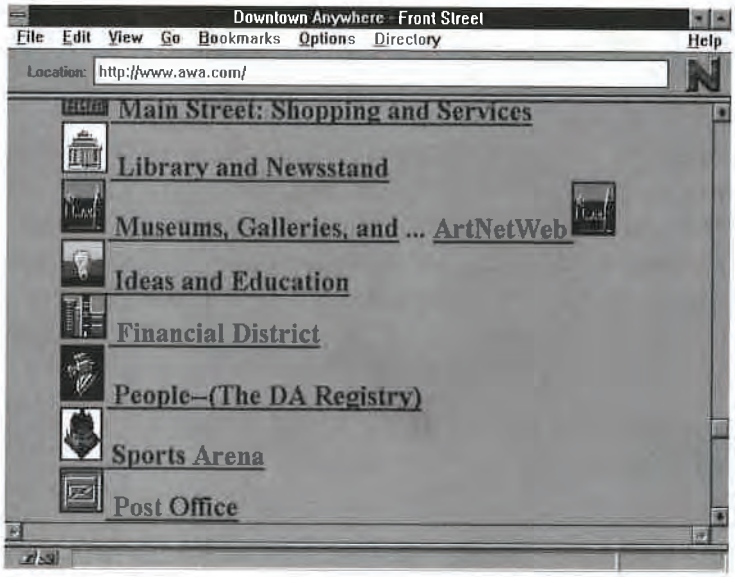


Figure 8-37 Downtown Anywhere

Google/YouTube Exhibit 1005/1105
Page 00239

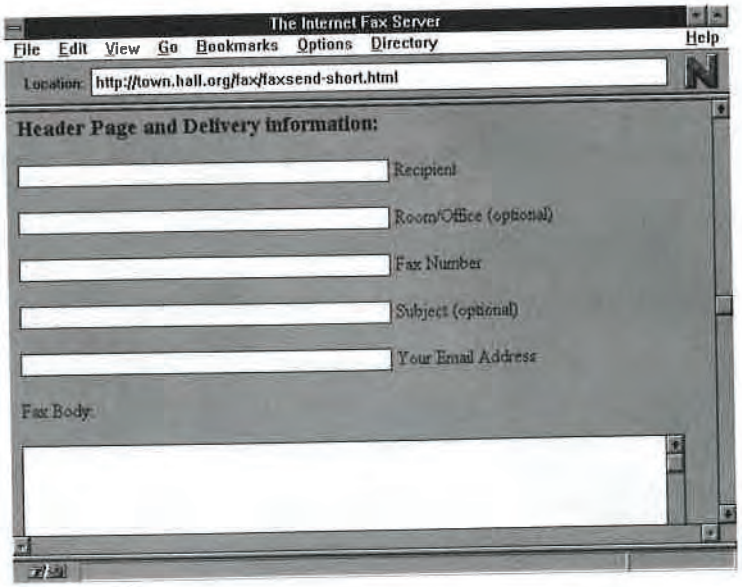


Figure 8-38 Internet Town Hall's fax utility

You can visit some of the businesses and people who have made Downtown Anywhere a home away from home. Downtown Anywhere is basically a commercial Internet provider geared toward expanding businesses.

Internet Town Hall

`http://town.hall.org/`

This friendly corner of the Web contains an eclectic collection of ideas, places, and services. You can tour the Digital Deli to tickle your taste buds, browse bookstores, or even visit the North Pole and meet a virtual Santa.

Much of this is geared toward Internet Talk Radio—a service that transmits live radio programs over the Internet. Additionally, lots of miscellaneous government databases can help you find any sort of official information you're interested in. The Internet Town Hall's Phone Company allows you to send an actual fax out to a friend just by filling in a form, shown in Figure 8-38.

BROWSE: LIBRARIES

`http://lcweb.loc.gov/`

Lots and lots of libraries and bookstores are online. Some libraries even have the full text of journals or books available for you to read. The Library of Congress has much more:

Google/YouTube Exhibit 1005/1105
Page 00240

- HTML Exhibits: Documents you could see in person at the Library of Congress, such as the Dead Sea Scrolls or the original Gettysburg Address (see Figure 8-39)
- HTML American Memory: Special collections of documents and other American memorabilia
- HTML Thomas: All the legislative information you could need
- HTML Country Studies: Detailed information about any country, area, or city
- HTML POW/MIA Database: The names, ranks, and serial numbers of prisoners of war or those soldiers missing in action
- HTML LOCIS: The entire Library of Congress catalog, listing nearly every English-language book in the world

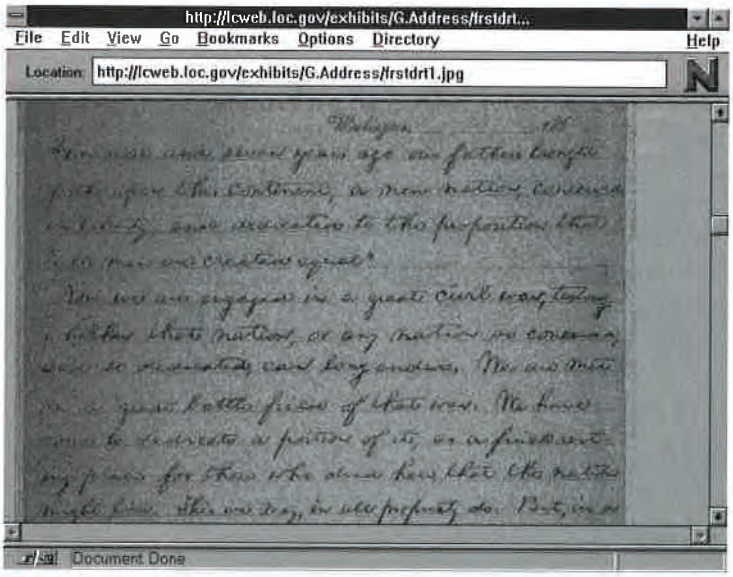


Figure 8-39 The Library of Congress has tons of treats to offer

Google/YouTube Exhibit 1005/1105
Page 00241

SURPRISE: OTHER NEAT GIMMICKS

As the Web matures, various creative people are finding totally new uses for it. By hooking up Web servers to video cameras, robots, or clever software programs, the Web can serve as a global front-end for almost any local routine. Some of these gimmicks are silly, others are downright amazing. As technology gets faster and more detailed, the Web will be used to control bigger and better things. Eventually, you may be able to duck into the Web before you leave work and switch on your home porch light, turn up your thermostat, start your coffee maker, and warm up the barbecue—almost as good as having a Spousal Unit minding the home fires!

LabCam

`http://found.cs.nyu.edu/cgi-bin/rsw/labcam1`

NYU's original LabCam, shown in Figure 8-40, lets you view a frame from a live interactive movie. A robot-mounted camera points to a spot in the NYU Robotics Lab. Click on the image. The camera will now focus wherever you clicked. The Web page is reloaded and the picture is updated. You can use LabCam to gaze all around the lab. Sometimes, you and another Web user will both try to control LabCam at once. You can have some neat fights with people, trying to beat them to the draw (or the click).

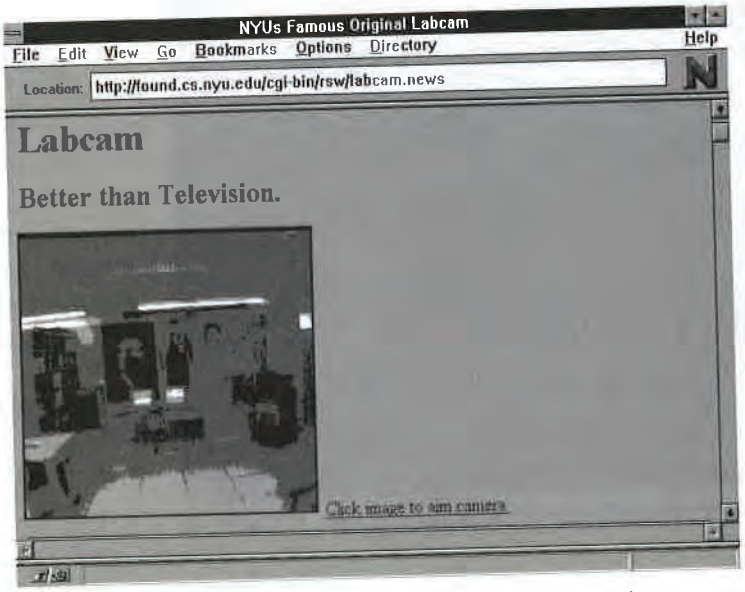


Figure 8-40 Peeking at a New York city lab with LabCam

Google/YouTube Exhibit 1005/1105
Page 00242

The Cyrano Server

http://www.nando.net/toys/cyrano.html

Make any day Valentine's Day. Use this form to select a type of love letter (choose from Steamy, Indecisive, Surreal, Desperate, Intellectual, or Poetic). Fill in a few adjectives, verbs, or other descriptions. Tell Cyrano who the letter should be sent to. A custom love letter whizzes off to your soulmate. Results may vary. For example:

```
Dearest Lover,
My love, I can imagine myself kissing your curious body and slathering
you with various oils and ice cream. Your hairs are my anchor in the
stormy sea of life; I wonder how I ever made it through a day without
you. Please meet me dressed in your flannel shirt on Valentine's Day
and we will celebrate our tentative love together.
```

Lite-Brite

http://www.galcit.caltech.edu/~ta/cgi-bin/asylhome-ta

The Inmate's Asylum home page leads to all sorts of wacky areas. There's a revolving door hotlist that will toss you into a random Web site. There's a fiction therapy group. There's an online cuckoo clock.

One of the neatest Asylum areas, however, is the Web Lite-Brite machine, shown in Figure 8-41. Using small colored lights, fill in a grid

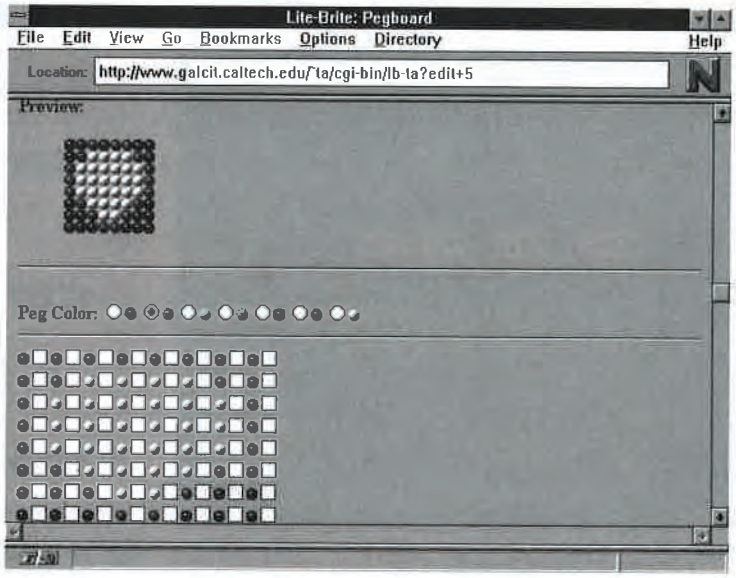


Figure 8-41 The Asylum's Lite-Brite machine

Google/YouTube Exhibit 1005/1105
Page 00243

with the lights you want. Once you get used to the interface, the Lite-Brite machine gets addictive. You can create some truly beautiful, albeit primitive, scenes.

The Virtual Frog

`http://george.lbl.gov/ITG.hm.pg.docs/dissect/info.html`

Tadpoles of the world, rejoice! Squeamish biology students, eat your lunches! The virtual frog Web pages shown in Figure 8-42 allow you to "slice" through an illustrated frog, learning about all its organs and other anatomical features. You can peel away the frog's skin and remove each of its organs, one at a time. You can rotate the frog or zoom in on it. You can even create a froggy movie. Select the axis of rotation and the frog will spin on your screen, allowing you to see its innards from all angles.

The Virtual Frog Dissection Kit comes to you courtesy of Lawrence Berkeley Laboratory, University of California.

Say...

`http://www_tios.cs.utwente.nl/say/formpost/`

Want your Web to talk back? This simple but extremely neat Web page allows you to type any word or phrase, as shown in Figure 8-43. Say... will

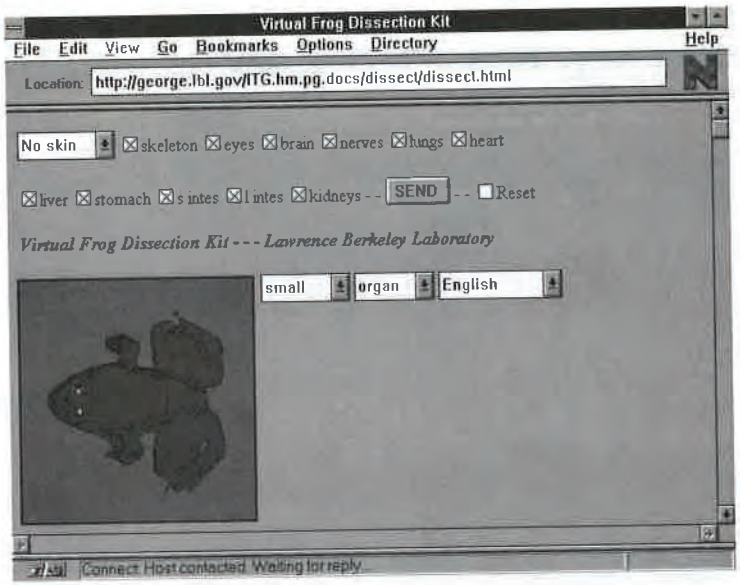


Figure 8-42 The Virtual Frog Dissection Kit

Google/YouTube Exhibit 1005/1105
Page 00244

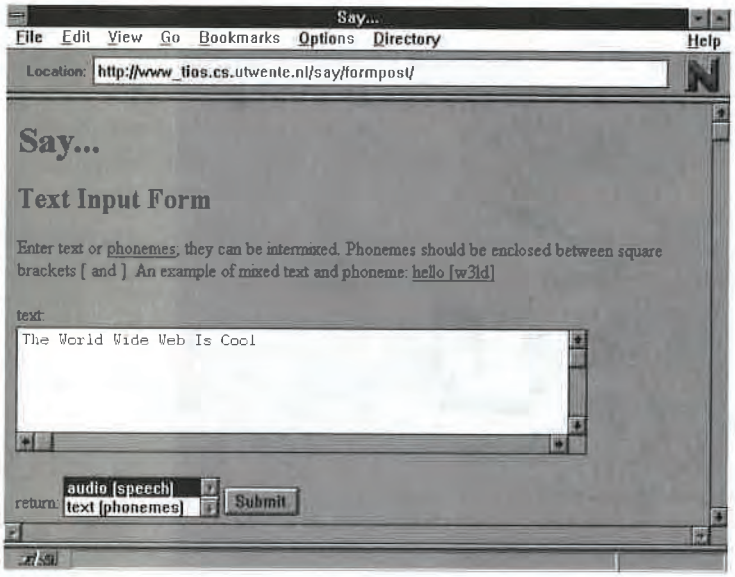


Figure 8-43 Say...

then create a special sound file containing the digitized speech and send that file to you.

Bianca's Smut Shack

`http://bianca.com/shack/index.html`

This colorful off-color Web page is a fun place to wallow in. Even better, though, are some of the neat rooms you can visit:

HTML The Bathroom: Read or write graffiti on the walls, ceiling, or floor. You can use HTML commands within your message to make bold, italicized, or otherwise emphasized statements.

HTML Bianca's Bedroom (and Love Chamber): Talk with a nasty-mouthed Troll. These trolls respond to whatever you say, and you may find yourself wondering whether the Troll is computer generated, human, or something else entirely.

HTML The Foyer: Read about Bianca's latest exploits, and sign the guest book.

HTML Hall of Fame: Gaze at Bianca's favorite photographs.

Google/YouTube Exhibit 1005/1105
Page 00245

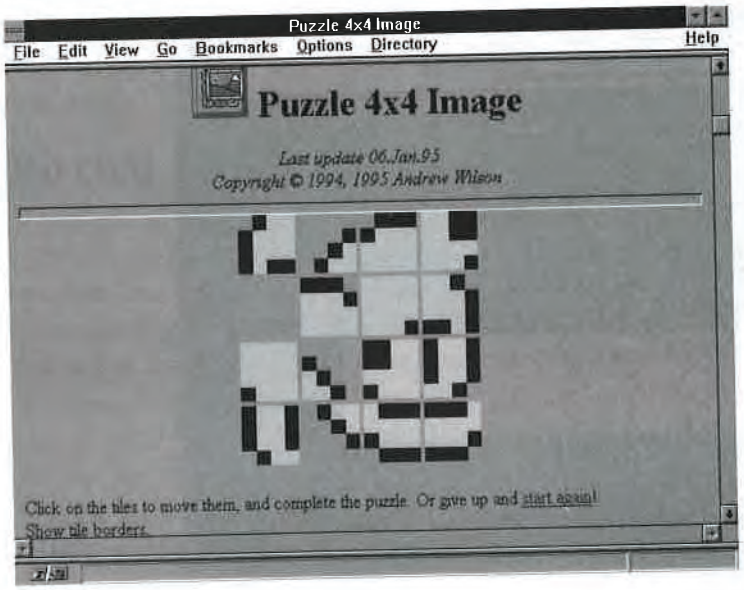


Figure 8-44 A Webbable finger puzzle

Most sections have a sort of forum area, where you can read or write comments. These comments are immediately "posted" to the Smut Shack for others to enjoy. Folks have been to known to have near-real-time conversations using these forums.

A Puzzler

`http://www.cm.cf.ac.uk/htbin/AndrewW/Puzzle/puzzle4x4image/`

Remember those cheap plastic puzzles that sometimes came as the prize in Cracker Jack boxes? There are fifteen colorful squares, arranged in a 4x4 matrix with one square missing, as shown in Figure 8-44. You can slide any square into the blank space, moving around the cubes until you form some sort of picture. It's like something between a jigsaw puzzle and Rubik's cube.

Well, now you can play the same game online. Andrew Wilson's creative Web page allows you to simply click on the tile you want to move.

WHAT NOW?

Inspired? A photo or brief description of a Web page can never do it justice. Try out some of these sites for yourself. Start thinking about your own

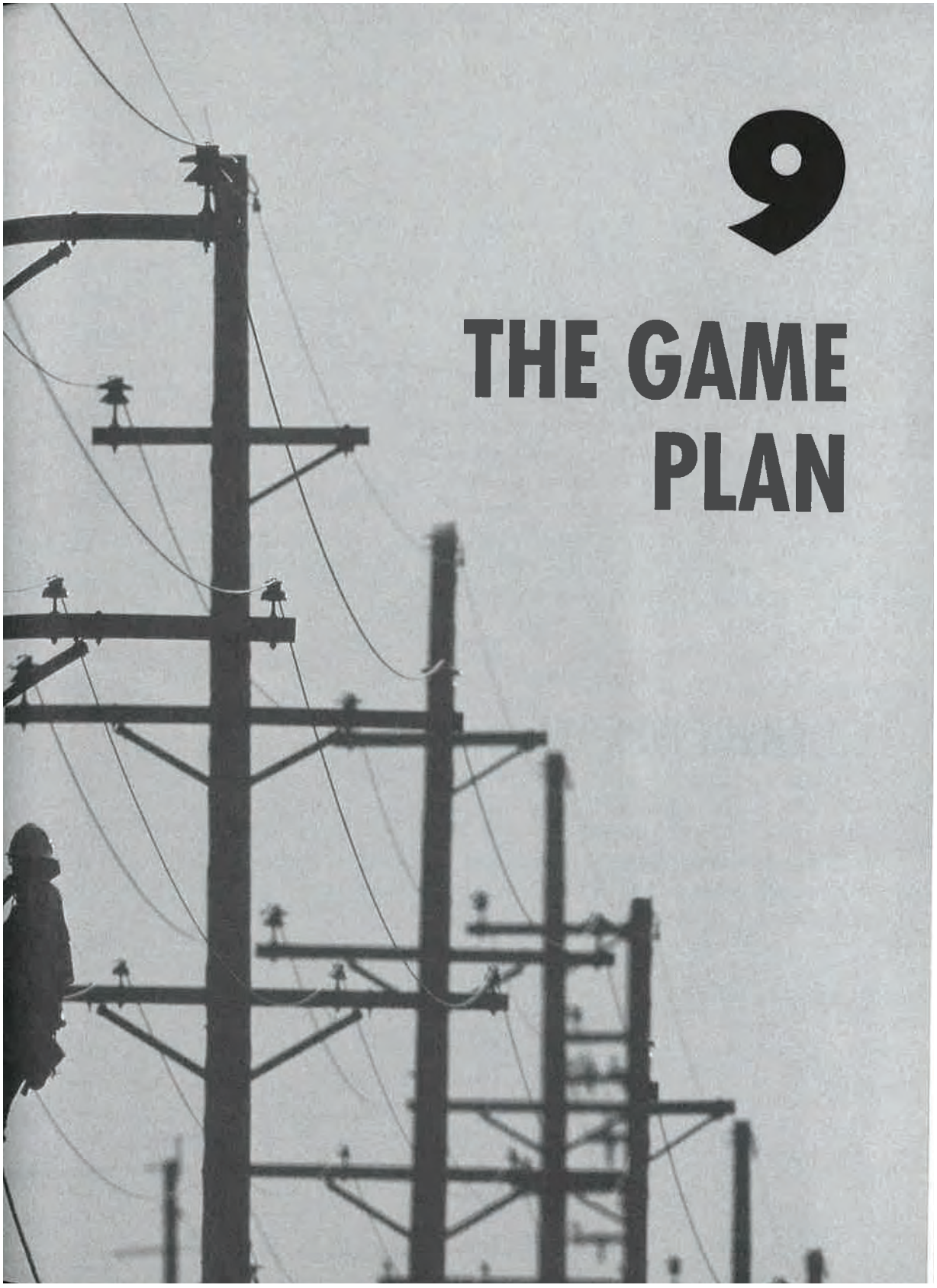
Google/YouTube Exhibit 1005/1105
Page 00246

pages—how they'll look, what they'll offer, and how you'll design all those features. The next chapter gives you some hints on what to concentrate on when you create a Web page, how to refine your view of your audience and your objective, and what pitfalls to watch out for.

Chapters 10 through 16 get down and dirty, teaching you how to pull off everything from basic hypertext to great-looking graphics to sound to full interactivity. Chapters 17 through 19 cover various editors, translators, and other utilities that can make writing a Web page a lot easier and a lot more fun.

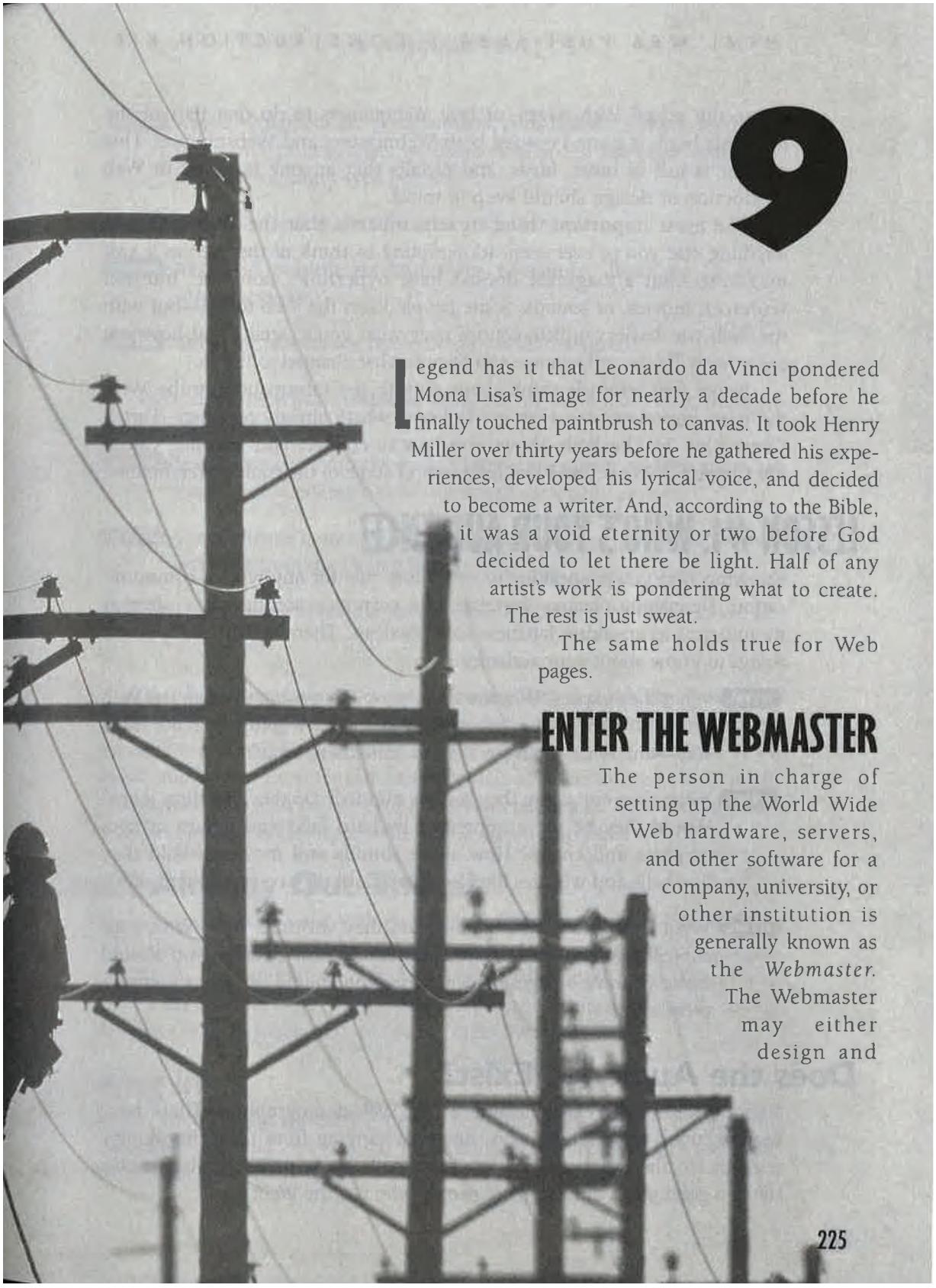
So get started! The world is looking forward to seeing you.

Google/YouTube Exhibit 1005/1105
Page 00247



9

THE GAME PLAN



9

Legend has it that Leonardo da Vinci pondered Mona Lisa's image for nearly a decade before he finally touched paintbrush to canvas. It took Henry Miller over thirty years before he gathered his experiences, developed his lyrical voice, and decided to become a writer. And, according to the Bible, it was a void eternity or two before God decided to let there be light. Half of any artist's work is pondering what to create. The rest is just sweat.

The same holds true for Web pages.

ENTER THE WEBMASTER

The person in charge of setting up the World Wide Web hardware, servers, and other software for a company, university, or other institution is generally known as the *Webmaster*. The Webmaster may either design and

create the actual Web pages, or hire *Webspinners* to do that part of the job. This book is geared toward both Webmasters and Webspinners. This chapter is full of hints, ideas, and pitfalls that anyone involved in Web production or design should keep in mind.

The most important thing to remember is that the Web isn't like anything else you've ever seen. It's tempting to think of the Web as a vast magazine—but a magazine doesn't have hyperlinks, software, Internet resources, movies, or sounds. Some people liken the Web to TV—but with the Web you have complete control over what you're seeing and how you see it; with TV the only choice you have is what channel to flip to.

Before you seriously think about putting any information on the Web, the most important step is to get to know what's already out there. Part I: Connecting To The Web, shows you how to connect and navigate. Check out Chapter 8, What Can I Do? for a tour of some of the coolest Web haunts.

LESSON #1: WHO'S YOUR AUDIENCE?

Knowing who you're speaking to is the first rule for any type of communication. Describing creative impulses to a corporate accountant is often as meaningless as speaking Japanese to a Mexican. There are three important things to know about your audience:

HTML What they expect. Why are these people browsing through the Web to begin with? If they reach your site, what's going to make them stay—and what's going to bore or scare them away?

HTML What they enjoy. Are they serious minded? Do they like dirty jokes? Would they be more impressed by hard facts and figures or cool graphics and charts? How about sounds and movies; would they like bells and whistles like these or would they consider them silly?

HTML What they know. How familiar are they with the Web? With your subject matter? This lets you determine how detailed you should make your Web pages, and whether you should use fancy computerese terms.

Does the Audience Exist?

Nobody knows the exact breakdown of Web demographics. There have been a bunch of recent surveys, however, ranging from the painstakingly scientific to the quick and casual. Most polls show pretty similar results. Here's a good guess of the type of people who use the Web:

Google/YouTube Exhibit 1005/1105
Page 00250

- About 40 percent are programmers, consultants, hackers, computer buffs, MIS professionals, software developers, hardware engineers, or other computer-type people.
- About 35 percent are college or graduate students.
- About 15 percent are professors, educators, librarians, or other academic types.
- About 10 percent are hobbyists or casual net-surfers.

What does the typical Web user look, act, or think like? Well, there are some clear trends:

- The audience is mostly male: Nearly 90 percent of Web users are men. The male/female ratio is reduced each day, however.
- The audience is mostly North American: About 65 percent of Web users call from the United States or Canada.
- The audience is mostly young: Over 50 percent of Webbers are 20-something.

You should remember, however, that the Web is doubling in size and traffic every two months or so. Since the Web is so lively, intuitive, easy-to-use, and fun, it will likely become a major medium, joining the ranks of TV, radio, and books. Even if your audience doesn't seem to exist yet, it might be worthwhile to start creating for tomorrow's audiences. Your Web site may even become a reason for people to decide to join the Web.

What's Already Out There?

If you want to target your Web pages to a specific group, you should check out a list of Usenet newsgroups. If a group exists, that means there are people on the Internet who find that topic interesting. You should also search for mailing lists, WAIS databases, gopher sites, *frequently-asked-question* (*FAQ*) archives, and, of course, other Web pages.

Focusing in

What, then, is the difference between a Web page written for a home computer user and one written for a Biology researcher? Here are some basic guidelines:

Google/YouTube Exhibit 1005/1105
Page 00251

Hackers/Techies

Most computer-type people like their data fast and accurate. You don't need to bother defining terms like The Internet, TCP/IP, or RAM, though you may want to make these terms into hyperlinks that lead to sections in an associated glossary page.

Try to include software for as many operating systems and computer types as possible. Ideally, a user should be able to click on a hyperlink and automatically download a demo version of your product or idea.

Academics

If your Web pages are directed toward a professional or academic group, you can think of your Web pages as an online journal. You can publish articles, photographs, images, or even movies that help explain the latest research, trends, news, or other developments. Be sure to use lots of references. Don't be afraid of specific jargon, either.

Businesspeople

Businesspeople buy and sell ideas as well as goods. Your Web page should be fact filled and very convincing. You can think of the page as a proposal, brochure, or report. Your product or service can be clearly explained and supported with relevant images, tables, sounds, or movies.

Artists

The Web is a great place to showcase any sort of writing, visual art, music, film, or animation work. Other artists, agents, producers, or art dealers can then review your pages.

Think of your Web page as a multimedia portfolio. Showcase only your best work, and make it easy to get to. If you're designing an online gallery, make the design mimic the art. If the art is traditional, create simple pages with an image and a title. You may also want to include a hyperlink that explains why you created the piece, or your artistic theory behind the piece. If the art is abstract, your Web page may be most effective if it is scattered with a collage of whimsical images, hyperlinks, and wild fonts.

Hobbyists

If your page is geared toward people who share your hobbies or favorite activities, then feel free to toss in photos, sounds, movies, and lots and lots

Google/YouTube Exhibit 1005/1105
Page 00252

of text. When it comes to things like fan clubs, the more information available the better. You can use slang or other specialized words for people in the know, though it's a nice idea to define these words (perhaps through hyperlinks) for beginners.

Man or Woman on the Street

The man or woman on the street may be an artist, a professional, an academic, or all or none of the above. If your page is geared toward general audiences, you should simply try to be as entertaining and illuminating as possible. Chances are, if you and your friends and your family seem to enjoy a Web page, so will many others.

LESSON #2: WHAT'S YOUR OBJECTIVE?

Why are you creating a Web area to begin with? Because it's the "hot" thing to do? Because you want to contribute useful information to the world at large? Because you want to promote a product or service? Because your competitors have already created Web pages? Because you want to showcase your talents?

Be sure your objective can be handled by the Web. If you want a detailed catalog and ordering system complete with secure credit-card retrieval, live customer service representatives, thousands of video images, and complex background music, then you may be biting off more than you can chew.

The Web works best in accomplishing the following three goals:

- HTML Familiarizing the Internet community with your name, product, service, or idea
- HTML Publishing text, images, software, or any other computer file to a potential audience of millions
- HTML Creating a sort of home base for any worldwide club, company, institution, or organization

The above goals are obviously very broad. Many Web pages manage to accomplish all three goals. If you have enough creativity, time, and patience, you can probably use the Web as a launching pad to meet any objective. There are, however, a few objectives that are much more difficult or risky than others.

Google/YouTube Exhibit 1005/1105
Page 00253

Sound and Video

Placing an occasional video or sound clip in a Web page is a great idea. The only caveat is that huge amounts of data take a long, long, time to transmit, especially over the "high-speed" modems many Webbers have to use. This type of connection works fine for text and simple graphics, but the Web cannot as yet transmit live television, quality video, live sound, or other lengthy recordings. See Chapter 12, Graphics, and Chapter 13, Sound, for more details.

Talking Back

Thanks to fill-in forms, the Web is somewhat interactive. You can have a user give you all sorts of information. You can then store this information, run various programs, and deliver a Web page or other Internet service that fits the user's specific needs. However, this process is too slow to actually hold live typed conversations. However again, you can use the Web to telnet to an IRC or other talk site, where such "chats" can take place. Much more information about interactivity, forms, and other on-the-fly Web programs can be found in Chapter 14, Interactivity, and Chapter 15, CGI Scripts.

Security

Right now, most Web pages are not at all secure. Any credit card information, payroll data, or other sensitive names or numbers are easily captured by prying eyes. Some Web servers, however, such as the one developed by Netscape, have various levels of *encryption*. When used in conjunction with the Netscape Navigator, a Netscape Commerce Web page can be transmitted over the Internet in a scrambled fashion. Such security is not yet implemented in all servers or browsers, however.

Most Web browsers and servers also support *user validation,* which allows only people with a proper password to view your Web pages. More information about security issues can be found in Chapter 14, Interactivity, and Chapter 15, CGI Scripts.

LESSON #3: THE HARD SELL

You can certainly try selling things over the Web. Most Web pages manage to sell things: ideas, services, artwork, products, people, or companies. But, at the same time, most Webbers abhor commercials. Why would someone want to waste time visiting you if all you're trying to do is push a product?

Google/YouTube Exhibit 1005/1105
Page 00254

There are enough ads in magazines, on radio, on TV, and on the sides of busses. How, then, can a Web page sell without directly selling?

Never take a boring print ad or display ad and just stick it on the Web. Not only will it generally be ignored, it will also offend some people. Instead, try to abide by the following three cardinal rules:

HTML Astound!

HTML Inform!

HTML Entertain!

Astound!

Why should people visit your page if they can get the same information someplace else? No matter what your objective is, always use a unique angle in presenting yourself. If you're an automotive company, don't just show a picture of your latest car. Anyone could do that! Instead, maybe create an animated car cartoon character, or maybe show exactly how your car was built—from the engine to the shell to the computer-controlled fuel injection system.

If you create a Web page that nobody has ever seen before, it will become hot in no time. If your layout is so darn cool and in-your-face, if you offer a unique interactive service, or if your page contains images or text that aren't available anywhere else in the world, then you've made your mark on the World Wide Web.

If you're creating a home page, for example, don't just stick in your name, address, and photograph. That's too standard. Instead, why not include a heap of information about your favorite hobby, or about your job, or even about your favorite rock group? For example, if you're a huge fan of Bruce Springsteen, you can create Web pages containing his album covers, lyrics, tour dates, latest gossip, etc. If you're a ballet dancer, include a list of dancing schools, ballet terms, photographs of famous ballerinas or information about upcoming shows.

By making your home page the main source for a given topic, hobby, or scene, you are not only telling the world more about yourself and your interests, you're allowing other people with similar interests to meet you and share your wealth of knowledge.

NOTE: Use a search index or list of sites to look for Web pages that sound similar to yours. Study those sites and think of a way to provide something new. Searching for similar pages also provides a good opportunity to cross-reference. When you build your page, include links to sites that deal with similar topics, and ask the makers of those sites to include links back to you.

Google/YouTube Exhibit 1005/1105
Page 00255

Inform!

Being informative goes hand in hand with being unique. Take advantage of the limitless possibilities of the Web, and offer data or advice that no other page can offer.

For example, if you are an electronic surveillance company, why not include some real-life case studies? You could then tell all sort of James-Bondesque stories, explaining what equipment was used, how the equipment worked, and what the outcome was. Show pictures of your products in action. Teach the reader all about surveillance techniques. You may even want to include a directory of licensed private eyes, organized by city, where people can turn if they need to do some snooping.

NOTE: Information should be accurate and up to date. It's better to have no information than erroneous information.

Entertain!

The real reason most people prowl the Web is entertainment. Depending on what your page offers, Web users may feel like they've just tiptoed through an art museum—or stomped through a mall.

Keep the writing on your page lively and to the point. Include clear, interesting images. Be savvy, be sexy, be surprising. Traveling around the Web is sometimes *too* fast and easy. If you bore a Webber, even for a moment, he may already be off across the world, browsing through another Web page.

If you're the tourist bureau for Des Moines, for example, don't just describe your city. Include lots of maps. Show lots of great pictures. Play sounds. All in all, try to capture the entire Des Moines experience and mood with multimedia. A visit to your Web page should feel almost like a visit to the city itself.

If you're trying to promote a new band, include samples of that band's music. If you're talking about a new book, be sure to include a full scene for people to read.

LESSON #4: OVERALL DESIGN

There is no right or wrong way to create a set of Web pages. There are some issues, however, you should be aware of.

Google/YouTube Exhibit 1005/1105
Page 00256

Point of Entry

Most Web sites have a home page, with graphics, hyperlinks, or even a structured Table of Contents leading the user to the other pages. The thing to remember about the Web, though, is that it's open-ended. You have no control over where users will enter your Web site. For example, you may have a home page that leads to the News page, the Sports page, and the Entertainment page. A famous Web site may decide to include a hyperlink to your Entertainment page only. You may get hundreds of people visiting your Entertainment site per hour, though only a fraction of them used your home page to get there.

Since your Web pages can have an arbitrary point of entry, keep the following in mind:

- HTML Every page counts. Don't spend a lot of time on one page and leave the others plain, jumbled, or boring.
- HTML Put your most important information, such as your e-mail address or phone number, on every page.
- HTML Always make it easy to return to your home page—or to get there after arriving from someplace else. Don't leave a page stranded out in cyberspace.

The Home Page

The home page generally makes the first impression. Based on your home page, people decide whether or not to continue exploring your particular corner of the Web. Here's the essential information for a home page:

- HTML Owner's name or company name
- HTML Logo or photograph
- HTML Contact address
- HTML Statement of objective
- HTML Description of what else can be found in this Web site

Google/YouTube Exhibit 1005/1105
Page 00257

The Main Menu

If you only have a few Web pages, you should list them all on your home page. You can create a descriptive menu—usually just a list of hyperlinks. You can also create clickable icons or other hyperimages to make your menu easier to access, as shown in Figure 9–1.

Many home pages now have a clickable *imagemap*. This is one image with many different sections, each leading to a different place when a user clicks on it.

Some "menus" are hidden within a story, or within other text. For example, you could create the following paragraph:

```
Welcome to Company X. We specialize in widgets, though we also have quite
a fine line of gizmos. The President of Company X, Frank Stein, has been
in the widget business since 1982. Since then, our company has grown to be
a worldwide widget importer.
```

A user could then click on the word "widgets," to learn about X's widgets, or could click on "worldwide" to see a world map with information about each of X's offices.

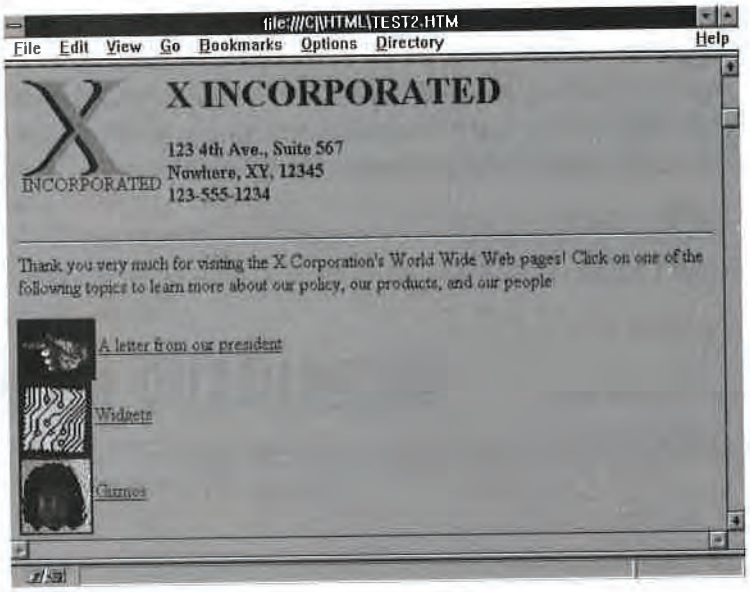


Figure 9–1 A Web home page with a menu of choices

Google/YouTube Exhibit 1005/1105
Page 00258

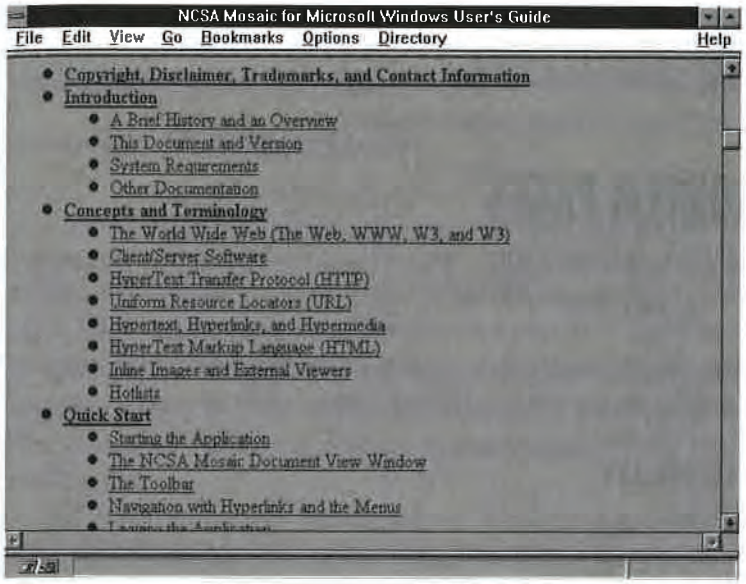


Figure 9–2 A Web home page with a Table of Contents

Table of Contents

If you have quite a few Web pages, or if you are putting a manual or book online, its a good idea to organize your pages in a Table of Contents. This is usually written in an outline-like style, as shown in Figure 9–2.

At the bottom of every page, there should be an option to return to this Table of Contents. This way, users can flip through your Web pages the same way they might use a reference book.

Search Index

If your Web site is *really* big, you may want to create a searchable index. This way, a user can type in a given word or phrase and a list of all relevant Web pages appears. Indexes can also be a great help if your Web site deals with many different types of products or issues. You'll find more information about indexes in Chapter 15, CGI Scripts.

Permanence

Does your Web page change often, like a magazine or weekly newsletter, or is it pretty much a one-time deal? If you want to create a weekly journal, for example, be sure you have the resources to design, write, and publish elab-

Google/YouTube Exhibit 1005/1105
Page 00259

orate content week after week. If not, you may want to keep your Web page design relatively simple.

If you want to create a more permanent page, then spend the extra time ensuring that everything looks as perfect as can be.

LESSON #5: PROBLEM PAGES

Many Web pages are sparse, sloppy, and designed without any interesting graphics or layout. Other pages use up *too many* resources—and that can be an even bigger problem. You may have created an absolutely beautiful Web page, a Web page that belongs on the ceiling of the Sistine Chapel, but if it takes too long to load then nobody will stick around long enough to see it.

Art Bombardment

A few Web pages have graphic lines zooming every which way. Every headline is in a different type of font. Small graffitios and sketches always pop up interrupting the text.

Using lettering, lines, and clip art as part of your layout turns a plain Web page into something special, but too many graphical elements can quickly get annoying. Don't murk up the content of your page with an abundance of cute trimmings. If you overdo any particular style, layout, or art element, then a page can quickly lose its appeal.

Graphic Gluttony

Some pages use screen-captures of fancy styled text and custom fonts instead of plain Web text. This may indeed make the page look better, but it can take what feels like forever to load. For example, if you use a fancy paint program to create a multi-colored, shadowed headline, it can take up to a minute to load and render. If you create the headline using standard HTML heading commands, then it takes a fraction of a second!

Other pages include a full-screen photograph after every paragraph, or a piece of artwork every few inches. Not only does this make the page look cluttered, it can take tons of time to load up.

If you include graphics, be sure they're small enough to load quickly. Even icons or reduced-size artwork, however, can get out of hand if you scatter them left and right. In general, your Web page should have less than six images.

NOTE: Some pages, such as art galleries, will naturally have many more graphics of paintings, photographs, or other images. If possible, warn the user

Google/YouTube Exhibit 1005/1105
Page 00260

before he or she selects such a page, with a message similar to: "WARNING: The following page contains a dozen large images and may take a while to fully load." If possible, split up such pages into many smaller pages.

Multimedia Moderation

Movies, images, and sound files are always huge. Although some Web users have lightning-fast connections, many others have to use a modem. For those with 14,400 baud modems, it takes nearly ten minutes to load a megabyte of information. In other words, downloading a movie or a large sound breaks up the Web session.

Don't rely entirely on a movie or song to carry your Web page. If possible, always include stills from the movie, or snippets from the song. This way, even users with slower connections can grab a taste of what you have to offer.

Textual Terseness

Text can load in fast, but it's still a good idea not to overdo it. Whenever most people come across a long story or article, they only read the first paragraph. This trend is well known in the world of journalism, and is the reason why most news articles are written in the *pyramid style*. The most important information always goes first. After that, you get the details; the most important details first, and the other stuff at the end. You never know how long a person will stay on your Web page. Try to use the pyramid style and hit your readers with the most essential facts, figures, or ideas first.

Use lots of numbered and bulleted lists to organize concepts. These lists leap off the screen and are always easier to read than blocks of text. Split your articles into lots of headings and subheadings, so people can easily scan through and find what they want.

You should also split up any long page into several different pages. This way, a reader will never have to wait too long for a Web page to load.

LESSON #6: GETTING THE RIGHT LOOK

If they've done their job, some of the Web pages illustrated in Chapter 8, What Can I Do? have indeed given you ideas about what you can do. Many elements affect the look of a Web page:

HTML Text layout

HTML Hypertext and hyperimage links

Google/YouTube Exhibit 1005/1105
Page 00261

HTML Lettering

HTML Artwork

HTML Color

HTML Photos

HTML Fill-in forms or other interactivity

Placing these elements together is always a fun challenge. How you use these elements depends, of course, on your audience, your objective, and your sense of aesthetics.

If you're planning a vast Web for a big company, you may want to do what big-time film directors do for each scene: create *storyboards*. These are illustrated mockups of what you want each of the Web pages at your site to look like. You can then hold a meeting or two, and people can comment on these storyboards, add to them, or suggest other designs.

Consistent Controls

Consistency is always a key. A set of Web pages can certainly have a fresh, whimsical look to it, but if it gets too confusing, people will just drift away. Always begin each page with a similar heading. Put your address and other contact information at the bottom of every page. Links to similar areas should look the same every time you use them—don't use icons on one page and a bulleted list on another page.

The best way of achieving consistency is to use a menu bar, or a set of uniform links. You can include these navigation controls at the top or bottom of every page. For instance, many Web pages use the following links at the bottom of each document:

```
[Home] [Next] [Previous]
```

If a user clicks on [Home], then the home page appears. The [Next] button loads up the next page in the series, and the [Previous] button, naturally, loads up the previous page.

Figure 9–3 shows another example of a consistent menu bar. A set of custom icons at the top of the screen leads to the City Life, Nite Life, Expo, Media, or Main Menu section. This same menu is printed at the top of every page.

Google/YouTube Exhibit 1005/1105
Page 00262

Figure 9–3 The New York Web, where each Web page contains links to other major sections

The Look Should Fit the Structure

The style of your pages depends largely on how your Web site is organized. The idea is to link all pages to each other. If you have an unlinked Web page, then it's almost impossible for anyone to reach it.

The Menu Tree Look

If you aren't dealing with too many Web pages, or if your Web pages are organized in different sections, you probably want to use menus.

A typical menu consists of a home page that leads to main sections. Each of the main sections, in turn, may have its own home page, with menus to subsections. At the top of each page are links that return the user to the previous menu, or to the main menu. The Web page shown in Figure 9–3 fits this bill.

The Book Look

If your Web pages are meant to be read in some particular order, you should probably opt for the book look. The book always begins with a

Google/YouTube Exhibit 1005/1105
Page 00263

Table of Contents that points to various chapters, sections, or subsections. To access a section, just click on it.

At the bottom of each section there should be controls to take the reader back to the Table of Contents, to the next page, or to the previous page.

The Whimsical Look

If your Web pages don't have much to do with each other, you may want to link them arbitrarily. For example, you can have a page about Horses link to a page about Racing. You can link the Racing page to the Gambling page, and the Gambling page to the Nevada page. While this makes it harder for a reader to find a specific topic, it is often fun to shuffle through these types of pages just to see what you'll get next.

Cross-Browser Dressing

Finally, you must be sure your pages look good on all Web browsers. This includes not only Mosaic, Cello, and Netscape, but also text-only browsers such as Lynx. If your page includes lots of fancy images, lettering, or art, be sure to include enough text so that people without graphics won't get lost or confused. You may even want to create an alternate page, free of images altogether, designed specially for text-based browsers.

LESSON #7: LINKS TO OTHER PAGES

The only way people are going to reach your page is if other Web pages are linked to yours. The more interesting or pertinent your subject, the more links will be created. You may even get on a worldwide "Best of the Web" or "What's Cool" list, ensuring hundreds of daily visitors.

To keep this spirit of sharing alive, you should use your pages to help promote other Web pages. You should link to pages that:

- HTML Deal with the same topic you're dealing with
- HTML Cover a deeper or alternate angle on the topic you're dealing with
- HTML You used as references in any paper you wrote
- HTML You just think are darn cool

Chapter 20 discusses dozens of ways to publicize your Web site. Good publicity can draw a lot of people to your Web pages, but only permanent links will ensure a steady flow.

Google/YouTube Exhibit 1005/1105
Page 00264

NOTE: Although external links are a nice gesture, once people leave your Web area, they may get so preoccupied they'll never return. To avoid losing readers, you probably want to list external links as the last option or at the end of your Web pages.

Stale Links

It's perfectly fine to use external links or to include images, sounds, or other files from other Web sites. These sites, however, may often go down. Files or Web pages may be deleted or moved. Periodically check all external links to be sure they haven't gone *stale*. If a user clicks on a stale link, a rude message similar to "Domain Not Found!" usually appears.

If you find any stale links, remove them immediately. Often, a linked-to Web page may have moved to a different locale. If so, be sure to change the linked-to URL.

Senseless Links

Another thing to watch out for are links that don't make any sense. For example, if you have a Web page about sharks and then all of a sudden have the sentence

```
Click here to go to the Fine Art page.
```

people will be confused. What in the world do sharks have to do with Monet? Instead, use some sort of logic to introduce a hyperlink:

```
Although I spend most of my time studying sharks, I'm also very interested
in fine art.
```

If you are creating a menu of links, be sure the links are parallel, and make sense adjacent to each other. For example, the following links seem strange next to each other:

```
1. Lions
2. Tigers
3. Bears
4. The socioeconomic influences of poverty
```

Instead, organize your links under various headings or listings:

```
The zoology department publishes many Web pages about animals:
1. Lions
2. Tigers
3. Bears
The sociology department also has several interesting pages:
1. The socioeconomic influences of poverty
```


Google/YouTube Exhibit 1005/1105
Page 00265

LESSON #8: STUDYING THE STATS

The final task of Web design is to stick your completed pages up on the Web and analyze them. Most Web servers automatically generate statistics, letting you know how many people visited each of your Web pages, how long they stayed, and what systems they came from.

In some cases, you may find that your Web site is attracting more visitors than you can handle. If so, you should get a faster Internet line or more memory on your Web server. Otherwise, your site may get so crowded that it'll grind to a halt, or people may be refused access.

Another trend you might discover is that one Web page is getting all the attention. For example, you may come up with the following statistics:

```
The Home Page          70 visitors a day
The Entertainment Page 98 visitors a day
The Sports Page         4 visitors a day
The News Page           5 visitors a day
```

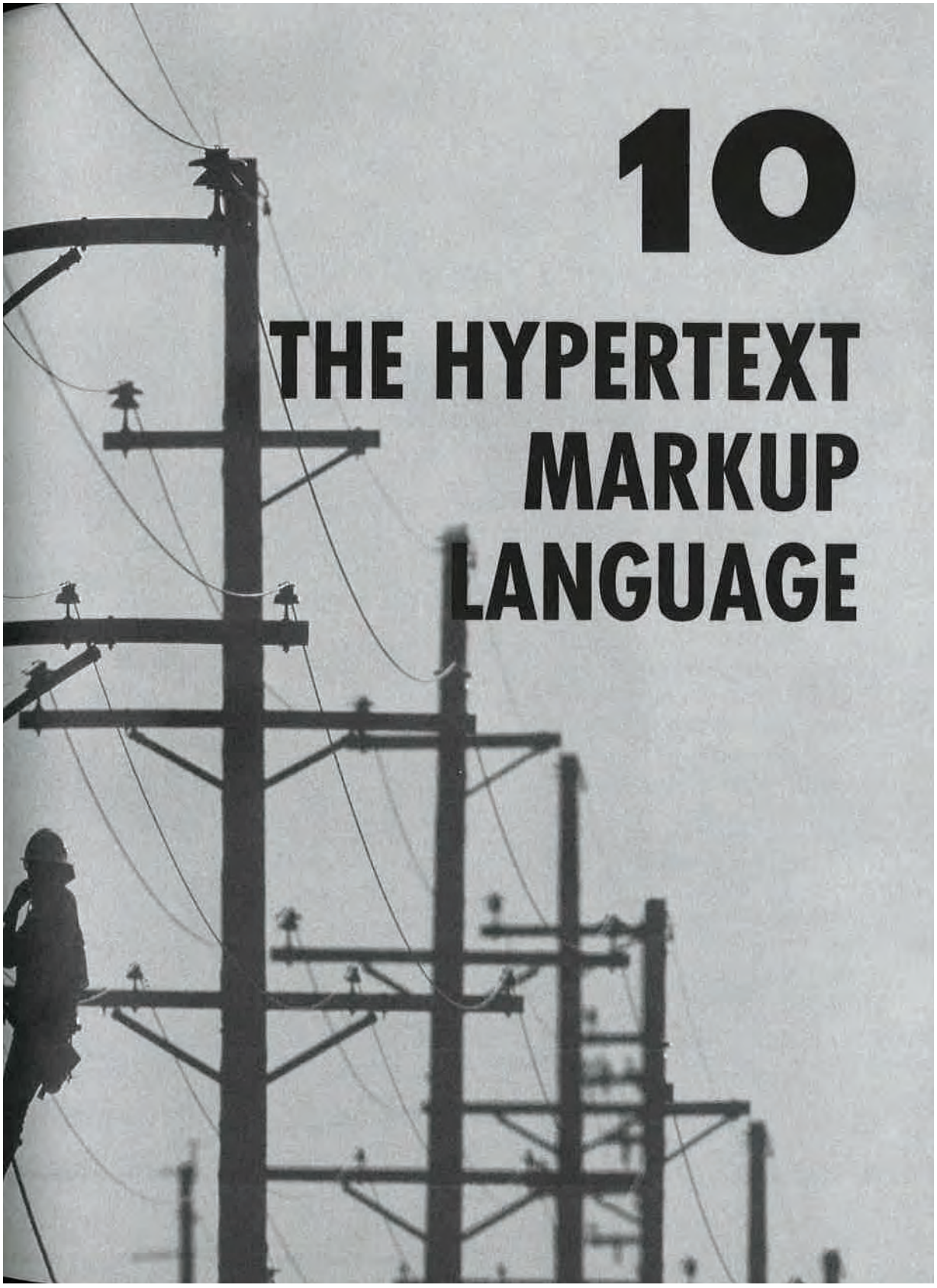
Why is the Entertainment Page so popular? Does the design of your Web make it difficult to reach the Sports Page or the News Page? In any case, you should be sure to take advantage of this fact. Be sure the Entertainment Page contains the most essential information you want to distribute.

After a while, you'll also learn which pages people avoid and which they love. This can help you know how to design and lay out any future Web pages.

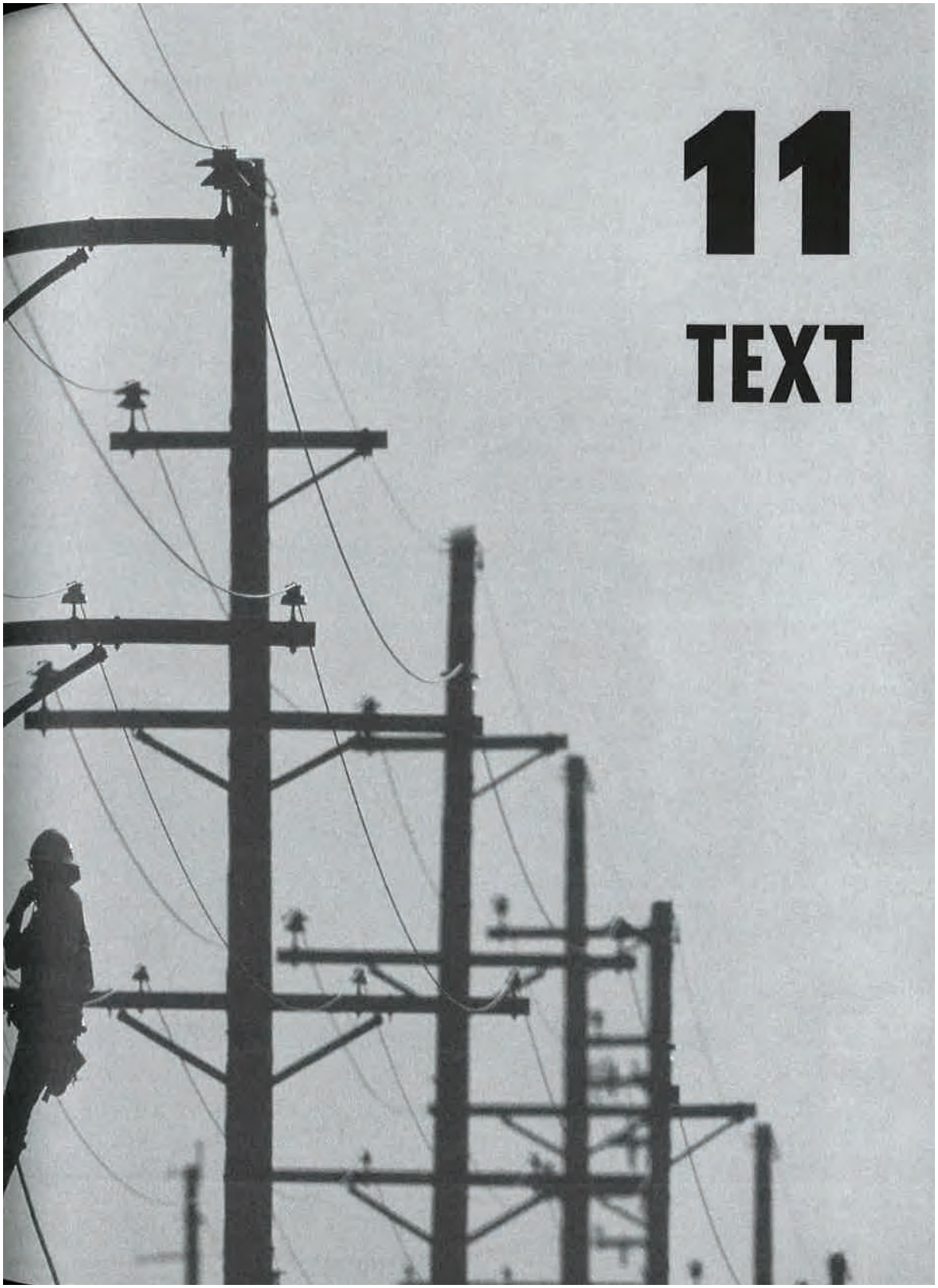
WHAT NOW?

Now it's time to start spinning those pages. The next chapter introduces the language used to create most Web pages: Hypertext Markup Language (HTML). Chapter 11 tells you everything you need to know about putting together Web text. Chapter 12 gives you a comprehensive gander at Graphics. Chapter 13 screams about sound.

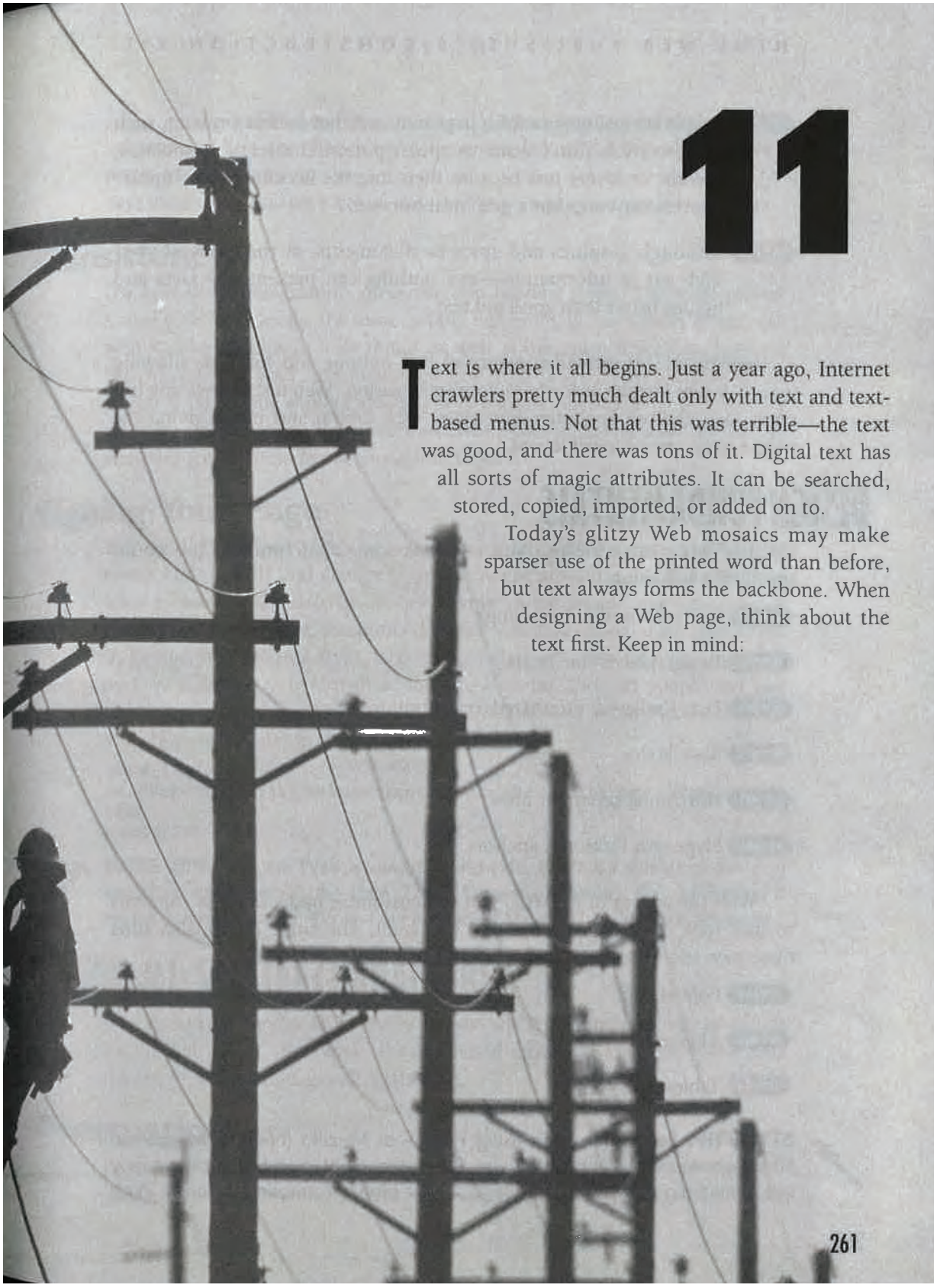
Google/YouTube Exhibit 1005/1105
Page 00266



10 THE HYPERTEXT MARKUP LANGUAGE



11
TEXT



11

Text is where it all begins. Just a year ago, Internet crawlers pretty much dealt only with text and text-based menus. Not that this was terrible—the text was good, and there was tons of it. Digital text has all sorts of magic attributes. It can be searched, stored, copied, imported, or added on to.

Today's glitzy Web mosaics may make sparser use of the printed word than before, but text always forms the backbone. When designing a Web page, think about the text first. Keep in mind:

Google/YouTube Exhibit 1005/1105
Page 00269

- People browsing your Web page may only have a text browser, such as Lynx. You don't want to spurn potential clients, associates, friends, or lovers just because their Internet accounts or computer systems can't support a graphical browser.
- Although graphics add spice to documents, at the heart of your Web site is information—and nothing can present raw facts and figures better than good old text.

Luckily, HTML provides plenty of text options and formats, allowing you to design layouts as fancy as your imagination. Web text is anything but plain. You can use headings, font sizes, lists, tables, and more, giving the page a crisp, professional layout.

DESIGN FUNDAMENTALS

Until HTML+ came along, Web text was somewhat limited. You could specify the following:

- Up to six levels of headings
- Paragraphs or line breaks
- Lists (bulleted, numbered, or definition style)
- Basic styles
- Horizontal separator lines
- Hypertext links and anchors

With the advent of HTML+, you can customize many of these elements so they look or act exactly the way you want. The latest HTML also adds these new text elements:

- Font sizing
- Alignment
- Tables

STYLE TIP: Be careful when using HTML+ or Mozilla (Netscape-supported HTML commands). People who use Lynx or another primitive browser may see something completely different. If you plan on creating a fancy Web

Google/YouTube Exhibit 1005/1105
Page 00270

page, it's a good idea to design a simple page containing the same general information. You can then have a hypertext link at the top of your fancy page, saying something like:

```
THIS PAGE LOOK STRANGE? Click here for the simple text-based version!
```

Consistency

Use consistent punctuation, grammar, and heading styles. The more professional your page looks, the more people will enjoy it, learn from it, and not skip it. This applies to little things as well as big ones: If you use bulleted lists to organize your information on one Web page, try to use lists on all pages. If the theme of your document is particle physics, try not to get off track by talking about the history of cream cheese. If you want to discuss something else, include a separate link to it.

Signing Your Page

You should sign every Web page you create so interested parties can get in touch with you. If you change your document around a lot, you may also want to include the date of the last change. Most people put signatures at the bottom of the page, surrounded by the <ADDRESS> element (described in Lesson #6, Section Style, later in this chapter). You may also want to include a link to your e-mail address as a *mailto: URL*, so people can just click on your name to send you e-mail. You'll learn about links in Lesson #13, Hypertext, later in this chapter.

```
<ADDRESS>
<A HREF="mailto:jsmith@smartpants.edu">John Smith</A>, February 14,
1995
</ADDRESS>
```

STYLE TIP: Since the Web is viewed worldwide, don't use numerical dates. In America, for example, the date 2/1/95 means February first; in Europe it means January second. Write it out instead—February 1 or Jan 2 or whatever.

LESSON #1: SPECIAL CHARACTERS

The computer keyboard is really handy when it comes to letters of the English alphabet. But what about special currency symbols, or accent marks, or other languages altogether?

Punctuation

By now you know that HTML commands are indicated by the less-than and greater-than signs (< and >). But what if you want to include one of these

Google/YouTube Exhibit 1005/1105
Page 00271

signs as part of your text? HTML uses the ampersand (&) to indicate special characters, as shown in Table 11-1. To print out a special character, write the ampersand followed by a special code and a semicolon, as follows:

```
&charactercode;
```

For example, to write <WOW!> you'd use the commands:

```
&lt;WOW!&gt;
```

Table 11-1 An HTML trick for basic punctuation

HTML Text	Appears As
&	&
>	>
<	<
	A nonbreaking space; use this to force a blank space

STYLE TIP: Since HTML ignores spaces it deems unnecessary, you can use the nonbreaking space character to force white space. You can then use this white space to separate text or even graphics.

Enhanced Characters

If your Web page includes any Spanish, French, or Latin words you'll probably want to use accented characters. Using the ampersand coding system, you can output most any type of extended or enhanced character.

STYLE TIP: Not every browser supports these enhanced characters. Be wary of overusing them.

For example, to write the word café, you'd write the command:

```
caf&eacute;
```

Table 11-2 Use these codes to print extended characters on your Web pages

HTML Text	Appears As	Description
Æ	Æ	(capital AE diphthong (ligature))
Á	Á	(capital A, acute accent)
Â	Â	(capital A, circumflex accent)
À	À	(capital A, grave accent)
Å	Å	(capital A, ring)
Ã	Ã	(capital A, tilde)
Ä	Ä	(capital A, dieresis or umlaut)
Ç	Ç	(capital C, cedilla)

Google/YouTube Exhibit 1005/1105
Page 00272

HTML Text	Appears As	Description
É	É	(capital E, acute accent)
Ê	Ê	(capital E, circumflex accent)
È	È	(capital E, grave accent)
Ë	Ë	(capital E, dieresis or umlaut)
Í	Í	(capital I, acute accent)
Î	Î	(capital I, circumflex accent)
Ì	Ì	(capital I, grave accent)
Ï	Ï	(capital I, dieresis or umlaut)
Ñ	Ñ	(capital N, tilde)
Ó	Ó	(capital O, acute accent)
Ô	Ô	(capital O, circumflex accent)
Ò	Ò	(capital O, grave accent)
Ø	Ø	(capital O, slash)
Õ	Õ	(capital O, tilde)
Ö	Ö	(capital O, dieresis or umlaut)
Ú	Ú	(capital U, acute accent)
Û	Û	(capital U, circumflex accent)
Ù	Ù	(capital U, grave accent)
Ü	Ü	(capital U, dieresis or umlaut)
Ý	Ý	(capital Y, acute accent)
á	á	(small a, acute accent)
â	â	(small a, circumflex accent)
æ	æ	(small ae diphthong (ligature))
à	à	(small a, grave accent)
å	å	(small a, ring)
ã	ã	(small a, tilde)
ä	ä	(small a, dieresis or umlaut mark)
ç	ç	(small c, cedilla)
é	é	(small e, acute accent)
ê	ê	(small e, circumflex accent)
è	è	(small e, grave accent)
ð	∂	(small eth, Icelandic)
ë	ë	(small e, dieresis or umlaut mark)
í	í	(small i, acute accent)
î	î	(small i, circumflex accent)
ì	ì	(small i, grave accent)
ï	ï	(small i, dieresis or umlaut mark)

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00273

Continued from previous page

HTML Text	Appears As	Description
ñ	ñ	(small n, tilde)
ó	ó	(small o, acute accent)
ô	ô	(small o, circumflex accent)
ò	ò	(small o, grave accent)
ø	ø	(small o, slash)
õ	õ	(small o, tilde)
ö	ö	(small o, dieresis or umlaut mark)
ß	ß	(small sharp s, German (sz ligature))
ú	ú	(small u, acute accent)
û	û	(small u, circumflex accent)
ù	ù	(small u, grave accent)
ü	ü	(small u, dieresis or umlaut mark)
ý	ý	(small y, acute accent)
ÿ	ÿ	(small y, dieresis or umlaut mark)
®	®	(registered trademark; supported by only a few browsers)
©	©	(copyright; supported by only a few browsers)

Other Characters

If you're interested in designing a Web page in a language other than English or containing hard-to-find characters, you have a few options:

HTML Some computers have automatic graphic filters to convert Roman characters into things like Arabic, Hebrew, Chinese, or Japanese. You can then use the standard ASCII set to create your Web pages. For those who don't own such filters, however, the screen will read as gibberish.

HTML The latest version of HTML has a CHARSET attribute for paragraphs, headers, and such, which can be used to specify alternate character sets, such as Hebrew, Japanese, or any other language a browser may support.

HTML Create your entire page using any foreign word processor or graphic design tool and then save the entire page as a GIF graphic. You can then make this graphic transparent (see the next chapter) and place it on your page for all to see. See Figure 11-1 for an example.

Google/YouTube Exhibit 1005/1105
Page 00274

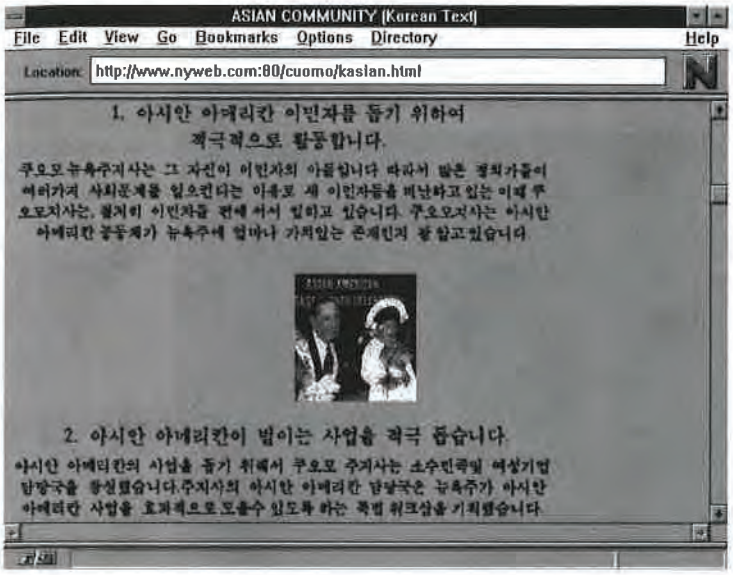


Figure 11-1 A Korean article on the New York Web

HTML You can obtain or create complete font sets with each letter stored in a separate transparent graphic file. These can be useful for foreign language headlines or small captions.

If you want to use complex mathematical equations, the latest version of HTML+ has some built-in mathematical functions. However, to be sure everybody can read your Web page, your best bet is to create the equations using a program such as LaTex. You can then use the LaTex2HTML converter (see Chapter 17) to form HTML pages.

Here are a couple of sites dedicated to multilingual servers: *http://andrew.triumf.ca/cgi-bin/country-switch/*

http://www.fer.uni-lj.si/.

LESSON #2: SECTION HEADINGS

Headings organize your document into smaller, easily-found blocks of information. Headings will generally appear as in Figure 11-2.

Google/YouTube Exhibit 1005/1105
Page 00275

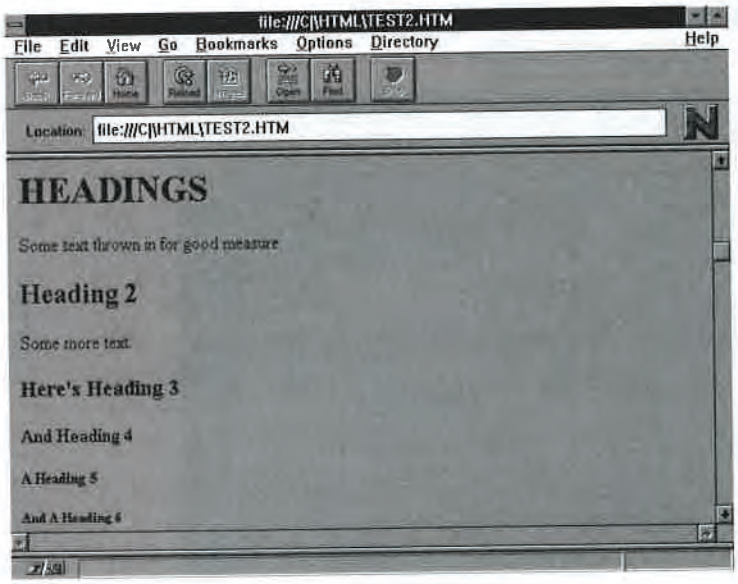


Figure 11-2 A hierarchy of headings

STYLE TIP: Although HTML supports six (and in some cases, seven) levels of headings, its a good idea to stick to four, otherwise your pages can get too complicated to wade through.

HTML headings take the form

```
<H#>heading</H#>
```

where # is the heading level—a number from one to six. For example, a document's main (largest) heading looks like:

```
<H1>Main Heading</H1>
```

STYLE TIP: The first heading is generally the same as the document title.

As in a printed document, the main headings should summarize the broadest ideas, with lower levels of headings forming a logical structure below them. Think of headings as your document's outline.

Here's the full HTML code for the headings and text in Figure 11-2:

```
<HTML>
<HEAD><TITLE>HEADINGS</TITLE></HEAD>
<BODY>
<H1>HEADINGS</H1>
Some text thrown in for good measure.
```


Google/YouTube Exhibit 1005/1105
Page 00276

```
<H2>Heading 2</H2>
Some more text.
<H3>Here's Heading 3</H3>
<H4>And Heading 4</H4>
<H5>A Heading 5</H5>
<H6>And A Heading 6</H6>
</BODY>
</HTML>
```

In general, you should only use heading tags when you're actually defining a heading. In other words, don't stick a heading in the middle of a list. Although Netscape and Mosaic can handle these misplaced headings, other browsers may become confused.

Style: Drafting Good Headings

Although there's no steadfast rule, subheads generally should not stand alone. Always have at least two subheads in each section. Here's an example of good heading layout:

Desserts

Pies

Apple

Key Lime

Pumpkin

Coconut Cream

Cakes

Cheese

Chocolate

Angel Food

Whereas a less-acceptable, splotchy layout would be:

Soups

Chicken

Desserts

Pies

Apple

It also usually looks nicer if you follow each heading by at least one sentence of text instead of immediately tacking on another heading.

Google/YouTube Exhibit 1005/1105
Page 00277

Describe each subsection so that the reader can decide whether or not to study, skim, or skip it. For example:

```
<H2>Desserts</H2>
There are too many popular and sumptuous desserts to name. However, the
traditional American favorites have been pies, cakes, mousses, and
various fruit dishes.<P>
```

The final caveat with headings is to be sure not to skip a level. Try not to follow an <H1> with an <H3> heading. It confuses some browsers—and some people—when you jump around like that.

Aligning Headings

More advanced browsers such as Netscape allow you to align your headings using the ALIGN attribute. You have the following alignment options:

- HTML Left (the default)
- HTML Center
- HTML Right
- HTML Justify

For instance, to center the main heading, use the command:

```
<H1 ALIGN="center">The Heading</H1>
```

LESSON #3: PARAGRAPHS

A Web browser needs to know when one paragraph ends and when a new one begins. It's essential to separate one paragraph from the next. You can do this by using a plain <P> element. This element may appear at either the beginning or end of each paragraph—you don't need to designate both. For example, the text in Figure 11-3 was achieved by the code:

```
Once upon a time there was a little man. When I say little, I mean
little. I mean real little. He was so little most people would just
pass him by. Others would see him and smirk. He felt belittled.<P>
So this little man made himself a little plan. He decided he wasn't
going to take the belittling anymore. He was going to show those
giants a thing or two!<P>
The little man waited for a foggy day. The clouds came down low. Most
everybody was blinded, but not the little man. He could walk between
the confused people's legs, and walk he did. Eventually he reached his
destination: The chainsaw store!<P>
```


Google/YouTube Exhibit 1005/1105
Page 00278

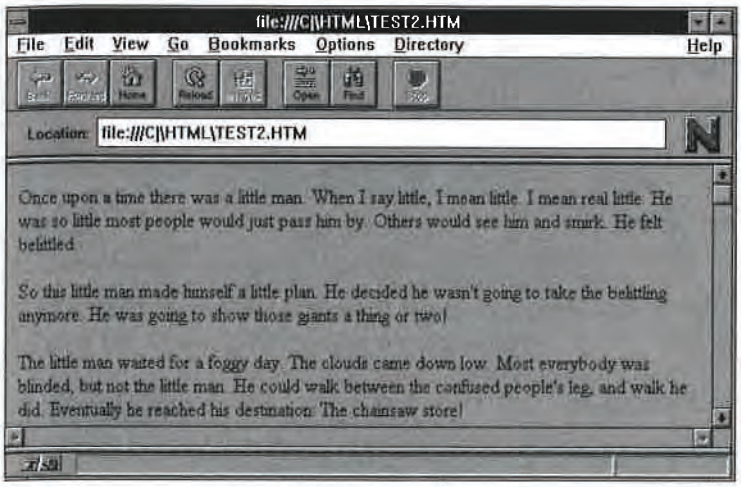


Figure 11-3 Picking a peck of paragraphs

NOTE: Remember, any extraneous blank spaces or lines in your HTML document are usually just ignored.

The latest HTML browsers let you define exactly how your paragraphs should look by using <P> at the beginning of your paragraph and </P> at the end. Eventually, this HTML+ paragraph *container* will allow you to define formatting attributes for each paragraph. For example, you could give each paragraph a unique ID, making it easy to reference, such as:

```
<P ID="Here">
```

More importantly, you could center align, left align, right align, justify, or indent each paragraph. For example, you would start each paragraph with the tag:

```
<P ALIGN="justify">
```

As of now, most Web browsers do not support these enhanced paragraph markers.

In general, it doesn't really matter whether you use one empty <P> marker or the <P></P> container. Use whichever technique seems most intuitive to you.

STYLE TIP: Some browsers will indent the beginning of paragraphs, others will leave a blank line or two. In any case, you don't need to worry about indentation. HTML takes care of this for you.

Google/YouTube Exhibit 1005/1105
Page 00279

You do *not* need to place paragraph markings around, in, or after nonbody text, such as:

- HTML Headings
- HTML Lists
- HTML Preformatted text
- HTML Addresses
- HTML Blockquotes
- HTML Horizontal rule lines

These elements have their own types of separators. For example, when you define a heading, all Web browsers know to add appropriate spacing automatically.

LESSON #4: LINE BREAKS

Now your paragraphs are nice and tidy. But what if you want to put text on specific lines within the same paragraph? For example, although things like postal addresses, poems, and songs are stored in paragraphs, you want to break them up into lines. The solution is the <BR> element.

For example, if your HTML document looks like

```
Hello there, dude.
How are you today?
I surely hope you are okay.<P>
That's good to hear!
```

Then, when seen over a Web browser, the text appears like:

```
Hello there, dude. How are you today? I surely hope you are okay.
That's good to hear!
```

However, if you insert the <BR> element after each line, as follows:

```
Hello there, dude. <BR>
How are you today? <BR>
I surely hope you are okay.<P>
That's good to hear!
```

Then the Web text appears on separate lines:

```
Hello there, dude.
How are you today?
I surely hope you are okay.

That's good to hear!
```


Google/YouTube Exhibit 1005/1105
Page 00280

No Break

If you have a line of text that absolutely, positively should not be broken up, you can designate this using the <NOBR> element, as follows:

```
<NOBR>
This is a piece of text that, for whatever reason, we want all on the
same line.
</NOBR>
```

As of now, only Netscape supports this function.

STYLE TIP: Be careful with this! If your no-break text is too long, it'll scroll right off the screen. This can be a pain to read.

Word Break

If you have a specific no-break section and know exactly where you wouldn't *mind* the text breaking, you can insert the <WBR> element. A word break (<WBR>) does not *force* a break the way <BR> does. Instead, it tells Netscape that, should a break be necessary, this is where it should be done.

You can use word breaks to achieve very precise blocks of text.

STYLE TIP: In general, text rolls over onto the new line once it reaches the end of the window. Be careful, because some HTML commands may break up text that is meant to be together. More advanced browsers like Netscape will only break up a line when it comes across empty space such as a space or a tab.

LESSON #5: TEXT STYLE

Most HTML browsers will display the following text emphasis styles:

HTML Bold

HTML Italic

HTML Underlined

HTML Preformatted Text

Highlighted text flows along with any neighboring text. You can also mix and match these elements to achieve, say, bold and italicized text.

There are two general style types: physical and logical. Physical styles tell the browser exactly what the text should look like. Logical styles, on the other hand, explain the *kind* of text being marked and let the browser do

Google/YouTube Exhibit 1005/1105
Page 00281

whatever it does with that kind of text. The next two subsections explain the differences, and you can use whichever type you feel most comfortable with.

All styles have a start tag and an end tag. To highlight a word, phrase, or entire paragraph, surround the text by tags. For example, to create boldface text you'd type:

```
<B>This text will be bold,</B> but this text won't.
```

STYLE TIP: You can usually mix and match emphasis elements to achieve, say, bold and italicized text. You should note, however, that browsers have their own rules for interpreting overlapping styles. Some browsers interpret only the innermost element, which would make

```
<B><I>Bold and Italicized Text</B></I>
```

appear in italics but not boldface. In general, you should immediately surround any text you wish to highlight with its proper physical or logical style elements.

Physical Styles

Physical styles tell the Web browser exactly how a piece of text should appear. Most word processors make use of physical styles. However, each browser will implement a physical style in its own way, and not every browser supports all physical styles.

The supported physical styles are:

HTML <B> **Bold text** </B>

HTML <I> *Italicized text* </I>

HTML <TT> `Typewriter fixed-width font text` </TT>

HTML <U> Underlined text </U>

HTML+ has specifications for the following additional styles. Most of these are currently supported by browsers such as Mosaic and Netscape:

HTML ^{Superscript}

HTML _{Subscript}

HTML <S> ~~Strikethrough~~ </S>

HTML <BLINK> Blinking text </BLINK>

HTML <CHANGE> Change bar </CHANGE>

Google/YouTube Exhibit 1005/1105
Page 00282

STYLE TIP: Try not to use too many different types of emphasis in a document, or your text will appear peppered. Stick to one or two styles and be consistent about what type of text you use them for.

Logical Styles

Logical styles, in most cases, get the same results as physical styles. However, a logical style will usually be more consistent from browser to browser. A logical style is more like a description of the enclosed text and is useful for writing scholarly journals, manuals, or other articles involving many types of print.

Here are the most popular logical styles:

HTML <DFN>: A word being defined. This word may later appear in a glossary. (Usually displayed as italics.)

HTML <EM>: Emphasized text. Use this to give words some oomph. (Usually displayed as italics.)

HTML <CITE>: A citation. Used around titles of books, journals, movies, poems, etc. (Usually displayed as italics.)

HTML <CODE>: Computer code. If you include a program's source code listing, you may want to surround it by this element. (Usually displayed as a fixed-width font.)

HTML <KBD>: Keyboard entry. If you tell a user to type a specific set of commands, surround the commands with this element. (Usually displayed as a bold, fixed-width font.)

HTML <SAMP>: Sample status messages. (Usually displayed as a fixed-width font.)

HTML <STRONG>: Strong emphasis. Good for warnings or other text that you want to leap off the screen. (Usually displayed as bold.)

HTML <VAR>: A variable. If you show a command, you can use this element to designate parts of the command that the user should replace with a legal parameter. (Usually displayed as fixed-width italics.)

HTML <Q>: An inline quote. (Only Netscape supports this tag, surrounding the text by proper open and closed quotation marks.)

Google/YouTube Exhibit 1005/1105
Page 00283

For example, if you wanted to emulate the following text:

```
To change directories in Unix, use the cd command, as follows:
cd directory-name
```

You would type the HTML sequence:

```
To change <DFN>directories</DFN> in Unix, use the cd command, as
follows:<P>
<KBD>cd <VAR>directory-name</VAR></KBD><P>
```

LESSON #6: SECTION STYLE

Giving individual words a particular emphasis is a great way to clarify your text. HTML also offers the ability to mark entire paragraphs or text sections with a particular style.

Preformatted Text

If you have a table, indented program listing, a free-form poem, or a tiny piece of ASCII art, you want it to appear over the Web the same way it appears on your screen. The *<PRE>* element allows you to mark a section of text as preformatted. This text is then printed using the same line breaks, and fixed-width characters. For example the poem:

```
<PRE>
This is a poem,
                    I'm not sure why I wrote it.
But it's a poem nonetheless,
                    Try not to over-quote it.
</PRE>
```

Would appear exactly as you entered it:

```
This is a poem,
                    I'm not sure why I wrote it.
But it's a poem nonetheless,
                    Try not to over-quote it.
```

STYLE TIP: Hyperlinks can be used within <PRE> sections without any problems. You should avoid using other HTML tags within preformatted text, however, since there's no telling how they'll appear. Since line breaks are automatically interpreted, there's no need for <P> paragraph tags or <BR> line break tags. You should also avoid tabs, since every Web browser may have different tab settings. Use several blank spaces instead, which will always line up in a fixed-width font.

Google/YouTube Exhibit 1005/1105
Page 00284

The Joy of Fixed-Width Fonts

Fixed-width fonts make each character the exact same size. With most other fonts, spaces, periods, and dashes are much shorter than alphanumeric characters—this usually gives the text a nice cosmetic appearance, but it can also make it hard to align different lines. For example, <PRE> lets you draw this cute little animal head:

```
/\___/\
.     .
\     /
 \   /
  \ /
   0
```

If you don't use the preformatted text style, such a picture would come out skewed:

```
/\___/\
.   .
\   /
 \ /
 \/
  O
```

WIDTH

You can use the *WIDTH* attribute to let a Web browser know how long each line of text should be. The default is 80 lines. For example:

```
<PRE WIDTH=40>
This text will automatically be wrapped around after every forty char-
acters. This can be useful if you're trying to create a small block of
text, or some other similar effect.
</PRE>
```

will produce:

```
This text will automatically be wrapped
around after every forty characters.
This can be useful if you're trying to
create a small block of text, or some
other similar effect.
```


Google/YouTube Exhibit 1005/1105
Page 00285

Blockquotes

The *<BLOCKQUOTE>* element is useful for any sort of quoted text. This text will usually appear italicized and centered as a block in the middle of the page, as shown in Figure 11-4.

For example, if you're writing a paper about René Descartes, you could include:

```
Descartes writes:
<BLOCKQUOTE>
Thus I lived, in appearance, just like those who have nothing to do
but live a pleasant and innocent life and attempt to obtain the plea-
sures without the vices, to enjoy their leisure without ennui, and to
occupy their time with all the respectable amusements available. But
in reality I never desisted from my design and continued to achieve
greater acquaintance with truth.
</BLOCKQUOTE>
```

The latest specification of HTML (version 3) has abbreviated the <BLOCKQUOTE> element to just <QUOTE> and </QUOTE>. These shortened elements do not work with many browsers yet.

Addresses

You should surround any home address, office address, e-mail address, signature, or other personal information with the *<ADDRESS>* element. An address is generally put at the bottom of a document as a signature, or at the top as a sort of byline.

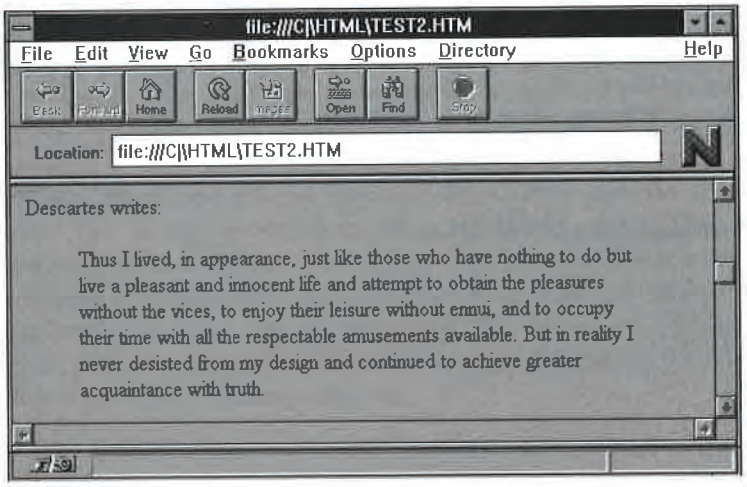


Figure 11-4 A nice way to quote a source

Google/YouTube Exhibit 1005/1105
Page 00286

An address is usually printed in italics. It may also be aligned to the right or otherwise indented. For example, you might sign your documents:

```
<ADDRESS>
John Smith<BR>
123 No Way<BR>
Metropolis, IN 12345<BR>
TEL: (000)555-1234
</ADDRESS>
```

STYLE TIP: It's a good idea to include your e-mail address at the bottom of every Web page you create, so interested parties can reach you easily. You may want to use the mailto: URL (supported by most of the latest browsers) to turn your address into an automatic e-mailer. See Chapter 16 for more information.

Styles to Watch Out For

The latest HTML specification includes several additional styles, such as footnotes, margin notes, abstracts, notes, and iconized blocks. While no commercial browsers support these styles yet, they probably soon will.

Footnotes

If you have a source, or some additional information that may ruin the flow of your narrative, you'll soon be able to designate it as a footnote with the *<FOOTNOTE>* element. A footnote may appear at the end of the text, or may even show up in a pop-up box.

For example, the code:

```
Most Biblical sources give no exact year for Armageddon.
<FOOTNOTE>
A handful of mystics throughout the ages, such as Nostradamus, have
marked the year 2000 to herald the end of the world.
</FOOTNOTE>
This "judgment day" is supposed to be characterized by natural disas-
ters, fire and brimstone falling from the sky, and general turmoil.
```

Would create something that looked like the following:

```
Most Biblical sources give no exact year for Armageddon.* This "judg-
ment day" is supposed to be characterized by natural disasters, fire
and brimstone falling from the sky, and general turmoil.
```

You could then click on the asterisk to read the additional information.

Margin Notes

The *<MARGIN>* element will be similar to a footnote. Marginized text appears to the left or right of the current paragraph, or as a separate

Google/YouTube Exhibit 1005/1105
Page 00287

paragraph. You'll be able to use this method to stick important warnings, notes, or tips in the margins.

For example:

```
The Colt .38 Special revolver contains the typical 6 bullets.
<MARGIN>
When loading a gun be sure to face it away from you!
</MARGIN>
```

Abstract

The *<ABSTRACT>* element will be useful when you write a scholarly paper. If you surround the summary or article abstract by this element, you should get an abstract that looks like the one in a formal paper: in a small font, centered and justified beneath the paper's title. For example:

```
<H1>The Vicissitudes Of Neo-Templar Restructuring in a Postmodern
Era</H1>
<ABSTRACT>
I have no idea what this paper is about but I'm sure it's interesting.
No, really!
</ABSTRACT>
```

Note

You'll be able to use the *<NOTE>* element for notes, tips, or warnings. A note will then be set off in a special box or bold font. For example:

```
<NOTE>Be sure not to touch the two wires together!</NOTE>
```

Role

The *<ROLE>* element will mark a specific piece of text with an icon. You'll be able to give a role the following attributes, each of which displays a different icon:

- *Simple*: No icon
- *Tip*: A pointing finger
- *Note*: The blue information circle
- *Warning*: A yellow traffic warning sign
- *Error*: A stop sign

For example, the command:

```
<ROLE TIP>Counting to ten is much easier if you use your
fingers.</ROLE>
```


Google/YouTube Exhibit 1005/1105
Page 00288

You'll also be able to use the *SRC* attribute to include your own icons. For example:

```
<ROLE SRC="caution.gif">Be careful not to touch the wires
together!</ROLE>
```

See the next chapter for details about including graphics.

LESSON #7: FONT SIZE

As of now, only the Netscape browser allows you to adjust font size. Adjusting fonts lets you design all sorts of special effects, as shown in Figure 11-5.

To change a font's size, just use the command:

```
<FONT SIZE=size>
```

The *size* value must be between 1 and 7. In general, a basic font is size 3. So, if you wanted to create a word that appeared to be shrinking, you could type:

```
<FONT SIZE=7>S<FONT SIZE=6>h<FONT SIZE=5>r<FONT SIZE=4>i<FONT
SIZE=3>n<FONT SIZE=2>k
```

You can use the </FONT> closing element to return to the default base font. For instance:

```
<FONT SIZE=7>This is big.</FONT> Let's go back to normal.
```

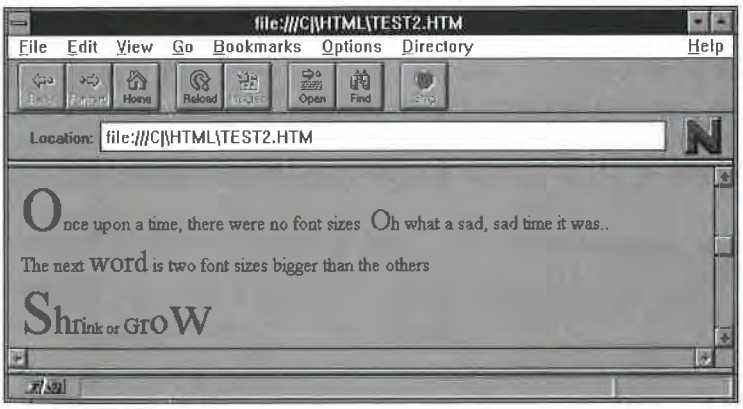


Figure 11-5 Changing font size to spice up your Web page

Google/YouTube Exhibit 1005/1105
Page 00289

STYLE TIP: Be careful when embedding different-sized fonts alongside other text, especially if the text is more than one line long. The disparity in font size will mess up the vertical spacing between the lines.

Relative Font Size

You can also use the plus or minus character to change a font relative to the document's base font. For example, if you wanted a word to be two sizes bigger than the rest of the text, you would type:

```
The next <FONT SIZE=+2>word</FONT> is two font sizes bigger than the
others.
```

The Base Font

The default font size is 3. If you want a document (or a section) to appear in a larger or smaller font, you can set the base font using the <BASEFONT> element. For example, to draw all text in the largest possible font, type:

```
<BASEFONT SIZE=7>
```

LESSON #8: CENTERING

Perhaps the most typically used style of text alignment is centering. Centered text is flashy, professional, and easy to read.

In previous sections, you learned about how you could alter the alignment of a paragraph or a heading by using the

```
<P ALIGN="center">
```

or

```
<H1 ALIGN="center">
```

attributes. This format is a bit unwieldy, however, and does not center everything you'd want it to.

Netscape has come up with a more convenient *<CENTER>* element. You can use this element to center any style, font, or manner of text. You can even use <CENTER> to center graphic images. Centering is simple:

```
<CENTER>
This text will be centered.<BR>
As will this!<P>
And this!<P>
</CENTER>
```


Google/YouTube Exhibit 1005/1105
Page 00290

LESSON #9: COMMENTS

Putting comments in your HTML code helps others steal from you more effectively. The public structure of the Web encourages people to borrow each other's Web pages and improve on them, customizing them to their own needs. Everybody learns from everybody else.

Anything placed within comments will not be printed by a Web browser. The only way for somebody to see your comments is to download a listing of your HTML source.

To begin a comment, use the

```
<!--
```

tag. At the end of each comment use the

```
-->
```

tag.

A typical comment, then, would look like this:

```
<!-- This section jumps to a random home page. -->
```

Most browsers do not allow you to nest comments (use a comment within a comment). They also get very confused if you use a double-dash (--) anywhere in your comment. And although Netscape and other advanced browsers let you comment out several lines of actual HTML markup, other browsers will read commands inside of comments anyway and output a big mess.

STYLE TIP: In general, then, try to restrict your comments to simple one-liners preceding each section. If you absolutely need to use multiline comments, put a <!-- at the start and a --> at the end of each line.

LESSON #10: LISTS

There are several types of HTML lists:

- HTML *Ordered:* a numbered list
- HTML *Unordered:* a bulleted list
- HTML *Definition:* a list with a term followed by a few words or sentences of description
- HTML *Plain:* a list with no numbers, bullets, or definitions

Google/YouTube Exhibit 1005/1105
Page 00291

HTML *Menu:* a short list of items, like a plain list, but tighter

HTML *Directory:* a thin listing of many items in several columns

Lists can be mixed, matched, nested, and combined. In other words, you can have an unordered list within an ordered list, as shown in Figure 11-6.

STYLE TIP: You usually add a period to the end of a list item if it's a complete sentence. If not, lists usually look better with no punctuation.

Ordered Lists

One of the most common ways to express a number of items is to give them all numbers. These numerical, or ordered, lists are great for describing a series of instructions or set of problems.

An ordered list begins with the *<OL>* element and ends with *</OL>*. Each list item itself is preceded by the empty *<LI>* element. For example, to make this list:

```
1. Remove the cat from the box.
2. Pet the cat.
3. Feed the cat.
4. Place the cat back in the box.
```

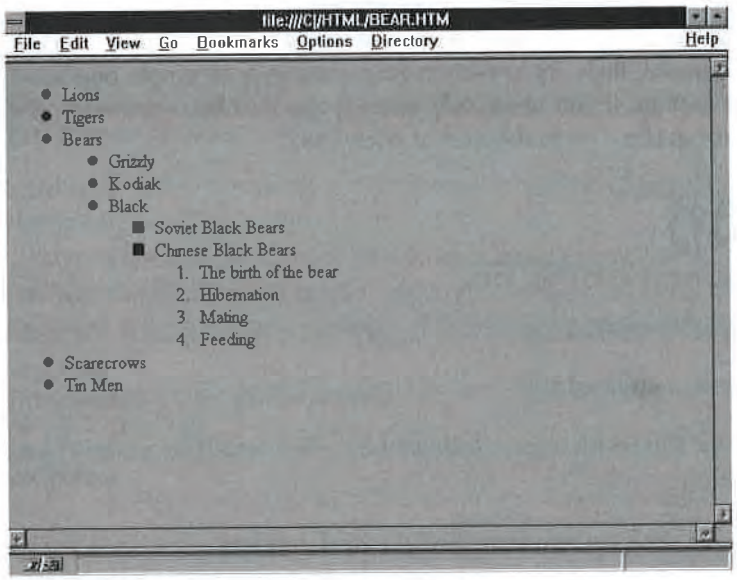


Figure 11-6 Nesting lists together to achieve an outline effect

Google/YouTube Exhibit 1005/1105
Page 00292

You would use the following HTML instructions:

```
<OL>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```

STYLE TIP: You can create a list without <LI> list elements to indent a block of text. This trick works with most browsers. For example, if you want to indent a special tip or idea, you could use the following code:

```
<UL>
This text will be indented, no matter how long the paragraph stretches
on. Pretty cool, eh?
</UL>
```

List Type

By default, an ordered list will count using standard numbers starting from 1, continuing with 2, *ad infinitum*. More advanced versions of HTML, such as Netscape's Mozilla, allow you to define the type of counting system you want. Use the *TYPE* attribute, as follows:

```
<OL TYPE=t>
```

In place of *t* you can fill in one of the following:

HTML *A:* capital letters

HTML *a:* lowercase letters

HTML *I:* large Roman numerals

HTML *i:* small Roman numerals

HTML *1:* standard numbers (the default)

So, for example, if you used the HTML sequence:

```
<OL TYPE=a>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```

Your list would appear as follows:

```
a. Remove the cat from the box.
b. Pet the cat.
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00293

Continued from previous page

```
c. Feed the cat.
d. Place the cat back in the box.
```

Why Start at One?

In some cases, you don't want your ordered list to begin with the number 1. You may be continuing a list from a previous section, or discussing a group of items from another list.

Using the *START* attribute, you can begin a list from any number you wish. For example, type

```
<OL START=4>
```

to begin your list with the number 4. The actual value of the list will still depend on the TYPE attribute. So, for example, if you designate:

```
<OL TYPE=I START=10>
  <LI>Remove the cat from the box.
  <LI>Pet the cat.
  <LI>Feed the cat.
  <LI>Place the cat back in the box.
</OL>
```

You'll get a list that looks like this:

```
X. Remove the cat from the box.
XI. Pet the cat.
XII. Feed the cat.
XIII. Place the cat back in the box.
```

Unordered Lists

The bulleted, or unordered, list is one of the best ways to express a series of points quickly. You can use bullets to list features, names, things in an inventory, or just about anything else. An unordered list begins with *<UL>* and ends with *</UL>*. Just like the ordered list, each list element is preceded by the *<LI>* element.

To create the statement

```
The biggest tourist sites in New York City are:
- The Statue of Liberty
- The World Trade Center
- The Empire State Building
- Rockefeller Center
- Central Park
- The Metropolitan Museum of Art
```

You would use this sequence:

```
The biggest tourist sites in New York City are:
<UL>
  <LI> The Statue of Liberty
  <LI> The World Trade Center
```


Google/YouTube Exhibit 1005/1105
Page 00294

```
  <LI> The Empire State Building
  <LI> Rockefeller Center
  <LI> Central Park
  <LI> The Metropolitan Museum of Art
</UL>
```

Nesting Unordered Lists

You can easily create lists within lists, if you like, to achieve an "outline" effect. You can even mix and match unordered and ordered lists:

```
<UL>
  <LI> an item
  <LI> another item
  <LI> here's a nested list:
   <UL>
     <LI> a nested item
     <LI> another nested list:
     <OL>
       <LI> ordered item 1
       <LI> ordered item 2
     </OL>
   </UL>
  <LI> the last item
</UL>
```

STYLE TIP: Although you can nest as many lists together as you wish, try to curb the nesting to three levels. Beyond this, a list becomes hard to read.

In most cases, your browser draws a small black circle as the first-level bullet. Lynx uses asterisks, and a browser like Netscape uses a small solid disc. As you nest lists, the bullets become indented and the style of bullet changes. Figure 11-6 shows a sample Netscape nested list: circular bullets mark the first and second unordered levels, and squares mark the next two levels.

STYLE TIP: As with headings, no nested list item should stand alone. It always looks much better to make each level of your list have two or more entries. Why else would it be called a list?

Pick Your Own Bullet

Using the TYPE attribute, Netscape and several other HTML+ browsers allow you to designate different styles of bullets. You have three choices, no matter which level of the unordered list you're at:

HTML disc

HTML circle

HTML square

Google/YouTube Exhibit 1005/1105
Page 00295

So, to make a short list preceded by two squares:

```
<UL TYPE=square>
  <LI> Item 1
  <LI> Item 2
</UL>
```

Customizing Lists

You now have quite a bit of control over how your lists can look. But Netscape supports an even further level of list customizing. By adding the TYPE or VALUE attribute to the *<LI>* element, you can combine bullets and numbers within the same list, combine various types of numbers, or use various types of bullets.

For example, to make a list that counts backward by twos, using different types of numbers:

```
<OL START=8>
  <LI>Remove the cat from the box.
  <LI VALUE=6 TYPE=I>Pet the cat.
  <LI VALUE=4 TYPE=a>Feed the cat.
  <LI VALUE=2 TYPE=i>Place the cat back in the box.
</OL>
```

This code would produce:

```
8. Remove the cat from the box.
VI. Pet the cat.
d. Feed the cat.
ii. Place the cat back in the box.
```

To make a list with three different types of bullets, you could write:

```
<UL>
   <LI TYPE=disc> Item 1
   <LI TYPE=circle> Item 2
   <LI TYPE=square> Item 3
</UL>
```

Definition Lists

Definition lists are useful for any sort of listing that contains a piece of additional information for each list item. This is great for glossaries, detailed outlines, or even play scripts (where each speech appears as a "definition" for the name of the character saying the words). The list itself starts with the *<DL>* element and closes with the *</DL>* element.

Each item in a definition list has two parts, the term—designated by *<DT>*—and the definition—*<DD>*. The term—usually one word or phrase—must fit on a single line. The definition itself can take up as many lines as you like.

Google/YouTube Exhibit 1005/1105
Page 00296

Figure 11-7 shows a sample definition list. Here's the markup to create that list:

```
<DL>
  <DT> <B>The Statue of Liberty</B>
   <DD> This symbol of liberty and goodwill to all women and men was
donated to the United States by France. Situated on Liberty Island in
the New York Harbor, the statue is a sight which greeted American
immigrants for decades.
  <DT> <B>The World Trade Center</B>
   <DD> The tallest building in New York, and the second-tallest
building in the world. Also known as the Twin Towers.
  <DT> <B>The Empire State Building</B>
   <DD> Once the world's tallest building, it now ranks third. The
needle-like architecture makes this building a distinct New York land-
mark.
  <DT> <B>The Metropolitan Museum of Art</B>
   <DD> One of the world's premiere art museums, spanning most every
era and culture.
</DL>
```

STYLE TIP: It generally looks good to set the definition term in bold, italic, or some other sort of enhanced print. Note how each <DT> element in the above code is also surrounded by the <B> attribute.

The <DL> element also has the optional *COMPACT* attribute. You can use this to create a list that appears condensed on some browsers.

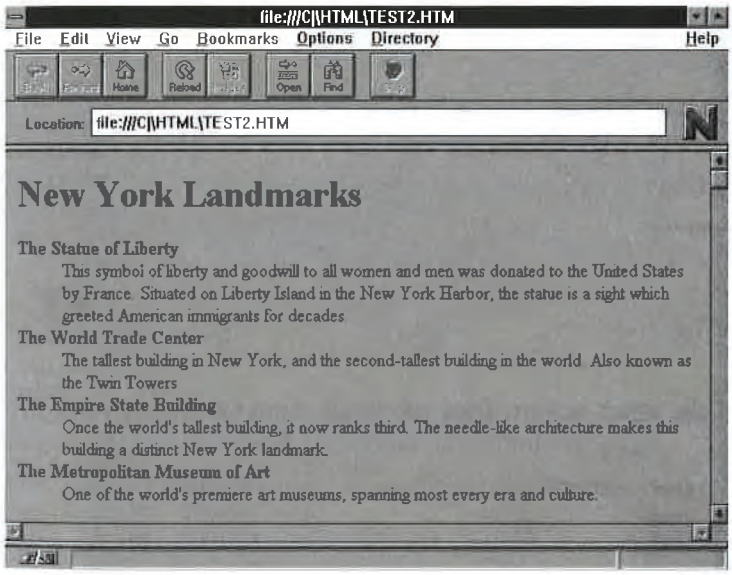


Figure 11-7 A standard definition list

Google/YouTube Exhibit 1005/1105
Page 00297

STYLE TIP: Officially, you're not supposed to use more than one definition (<DD>) for each term (<DT>). If you want a definition that is longer than one paragraph, just use the <P> element and continue writing. Some browsers may get confused if you create a definition without an associated term. You can, however, list a term without including a definition for it.

Plain Lists

If you want to list items without using any fancy bullets, numbers, or definitions, you can easily create a plain list. Simply add the *PLAIN* attribute to any unordered list container. For example:

```
<UL PLAIN>
  <LI>First item
  <LI>Second item
  <LI>The Third
</UL>
```

A plain list can either shoot straight down the screen, or wrap over from line to line. This way, a list of twenty elements will be ordered in columns across the screen rather than wasting many pages. To wrap a list, use the WRAP attribute, with one of the following tags:

HTML vert

HTML horiz

For example, you could create a list that went across the screen horizontally:

```
Item1   Item2   Item3   Item4   Item5   Item6
Item7   Item8   Item9   Item10
```

by using the markup:

```
<UL PLAIN WRAP=horiz>
  <LI>Item1
  <LI>Item2
  ...
  <LI>Item10
</UL>
```

If you specified the wrap as vert, then the items would be listed vertically instead:

```
Item1   Item4   Item7   Item10
Item2   Item5   Item8
Item3   Item6   Item9
```


Google/YouTube Exhibit 1005/1105
Page 00298

The <MENU> list and the <DIR> list, described in the next two sections, are a shorthand representation for popular plain lists. Try to use <MENU> and <DIR> whenever you can, since some browsers do not support the PLAIN attribute.

STYLE TIP: You can use the COMPACT attribute to give your plain lists a tighter appearance.

Menu Lists

A menu list looks very similar to an unordered list, except it usually consists of smaller paragraphs. The <MENU> element generally gets the same results as a plain list. You can use it to create a list that appears tighter, smaller, and with more white space:

```
<MENU>
  <LI>Caviar
  <LI>Champagne
  <LI>Beef Wellington
  <LI>Cheez Whiz
</MENU>
```

Directory Lists

The *<DIR>* element gives you a quick list of short words or phrases that wrap across the screen. In most cases, <DIR> gets the same results as the <UL PLAIN WRAP=vert> element. Each item in a directory list should be less than 20 characters long:

```
<DIR>
  <LI>John
  <LI>Mary
  <LI>Sue
  <LI>Bob
  <LI>Billy
  <LI>Kathy
  <LI>Frank
  <LI>Tim
  <LI>Elliot
  <LI>Mark
  <LI>Carlos
</DIR>
```

This would create something similar to the following:

```
John    Billy   Elliot
Mary    Kathy   Mark
Sue     Frank   Carlos
Bob     Tim
```


Google/YouTube Exhibit 1005/1105
Page 00299

Some browsers make no distinction between plain lists, menu lists, and directory lists.

LESSON #11: HORIZONTAL LINES

Having lots of text together on a screen can be difficult to read. Luckily, most every Web browser supports horizontal rule lines. Anytime you write <HR> the browser draws a thin line completely across the screen. You can use these lines to mark the beginning or end of sections, to make headings look snazzy, to begin a footnote section, or to separate text from inline graphics or icons.

In Netscape and Mosaic the horizontal rule looks like a thin engraved line. Lynx uses a string of dashes.

STYLE TIP: Do not overuse horizontal rules. Too many lines muddle up the page with empty space, and make things hard to read.

In Thickness and in Health

Netscape and several other HTML+ browsers allow you to adjust the thickness of each horizontal line. Usually the line is 1 unit thick, but you can make it as thick as you like. Simply use the *SIZE* attribute. To make a line 5 units thick, type:

```
<HR SIZE=5>
```

Setting the Line's Width

What if you don't want your line to stretch all the way across the screen? Many times it's nice to have lines that are just as long as a nearby heading. Another nice Netscape feature allows you determine a line's width by using the *WIDTH* attribute.

If you want, you can determine the exact width in pixels. When you decide how long to make a line, keep in mind that an average VGA screen is about 640 pixels wide. Here's what the element looks like:

```
<HR WIDTH=200>
```

Or, even more easily, you can just determine what percentage of the screen the line should take up:

```
<HR WIDTH=50%>
```


Google/YouTube Exhibit 1005/1105
Page 00300

Aligning Your Lines

Finally, Netscape gives you the ability to easily align your lines. Since lines can be any width, why not push them against the right margin or center them? By setting the *ALIGN* attribute you can achieve some nice results.

There are three ways to align your lines:

HTML Left (the default)

HTML Right

HTML Center

For example, this element creates a half-sized line in the center of the screen:

```
<HR WIDTH=50% ALIGN=center>
```

Using a Solid Bar

The standard Netscape horizontal line looks as if it was carved into the monitor with a laser beam. If you want a more traditional solid line, simply specify the *NOSHADE* attribute as follows:

```
<HR NOSHADE>
```

Figure 11-8 shows the results of setting various thicknesses, widths, alignments, and solidities, as follows:

```
<HR>
<HR SIZE=2>
<HR SIZE=5>
<HR SIZE=3 WIDTH=50%>
<HR SIZE=3 WIDTH=50% ALIGN=right>
<HR SIZE=3 WIDTH=50% ALIGN=left>
<HR SIZE=4 WIDTH=60% ALIGN=center>
<HR SIZE=6 WIDTH=40% ALIGN=center>
<HR SIZE=8 WIDTH=10% ALIGN=center>
<HR WIDTH=10% ALIGN=center NOSHADE>
<HR NOSHADE>
<HR SIZE=4 WIDTH=60% ALIGN=center NOSHADE>
<HR SIZE=6 WIDTH=40% ALIGN=center NOSHADE>
<HR SIZE=8 WIDTH=10% ALIGN=center NOSHADE>
```


Google/YouTube Exhibit 1005/1105
Page 00301

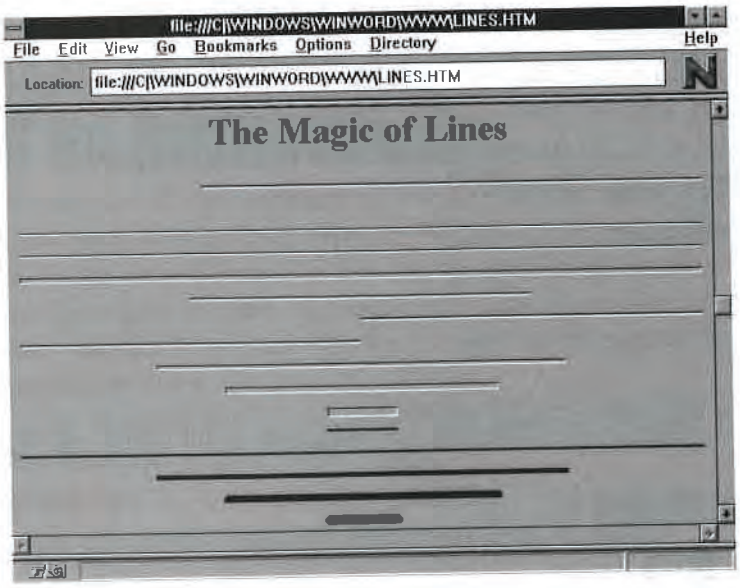


Figure 11-8 Going crazy with customized lines

LESSON #12: TABLES

There are three ways to create tables:

HTML Using preformatted text

HTML Using graphics

HTML Using HTML+ commands

Text Tables

Every browser has the ability to use preformatted text. Remember, this type of fixed-width text is designated by the <PRE> tag and closed by the </PRE> element.

You can create your table by hand, then, using ASCII characters, as follows:

```
+--------+----------+-------+
+  Name  |  Date    | Score |
+========+==========+=======+
| Ralph  | January  | 10    |
| George | February | 20    |
| Mary   | March    | 15    |
| Sara   | April    | 25    |
+--------+----------+-------+
Table 3: Each contestant's final score.
```


Google/YouTube Exhibit 1005/1105
Page 00302

This may not be the most professional look in the world, but it gets the job done.

Graphical Tables

You can use any software you'd like to create your table. Then capture the screen, convert the image into a GIF graphic file, and display this image on your Web page. See the next chapter for the specifics.

HTML+ Tables

The latest version of Mosaic and Netscape fully supports HTML+ table commands. Eventually, most browsers should support this feature. HTML table elements allow you to quickly slap together a very professional and sleek-looking table.

STYLE TIP: Since table commands are a relatively new feature, be sure to include a link to a standard preformatted text table. This way, people with browsers who can't view tables can still see your data.

You designate the start of a table using the *<TABLE>* element and end a table with *</TABLE>*. Within a table you can specify captions using the *<CAPTION>* element. Begin each row with the *<TR>* element. Header cells are designated using the *<TH>* element, and data cells are marked by the *<TD>* element.

Each cell can contain a word or a phrase, or entire paragraphs, lists, and headers—even graphical images. Figure 11-9 shows a sample table, which was created using the following code:

```
<TABLE BORDER>
  <CAPTION>Table 3: Each contestant's final score.</CAPTION>
  <TR><TH>NAME</TH><TH>DATE</TH><TH>SCORE</TH></TR>
  <TR><TD>Ralph</TD><TD>January</TD><TD>10</TD></TR>
  <TR><TD>George</TD><TD>February</TD><TD>20</TD></TR>
  <TR><TD>Mary</TD><TD>March</TD><TD>15</TD></TR>
  <TR><TD>Sara</TD><TD>April</TD><TD>25</TD></TR>
</TABLE>
```

Table Cells

Each row in a table begins with the <TR> element. If you like, you can end rows with </TR> though this is optional.

Any text that you want to head up your table begins with the <TH> element, and may end with the </TH> element. This text appears in a bigger, bolder font. Each cell of the table's actual data begins with the <TD> element, and may end with the optional </TD> element. Headings are usually created at the top or to the left of the data.

Google/YouTube Exhibit 1005/1105
Page 00303

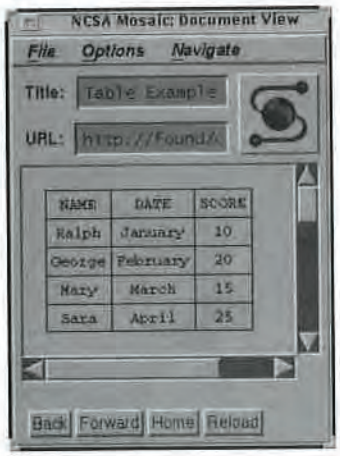


Figure 11-9 A simple but great-looking table created with HTML

NOTE: Rows can have different numbers of cells. The table is drawn accordingly.

STYLE TIP: When your Web browser comes across a table, it tries to figure out the table's size before it fills in any values. If the table is too wide to fit on the screen, it usually wraps text over within each cell. In this way, each cell acts as its own little window. Since this might make a table look awkward, try to limit the size of each cell. If you have too much data in one particular cell, you may want to use two separate tables.

A table's headings and data do not have to be plain text. You can place nearly anything within a table:

- Images
- Formatted text
- Various text styles
- Lists
- Hyperlink anchors
- Forms

Google/YouTube Exhibit 1005/1105
Page 00304

For example, you could make a table as complicated as the following:

```
<TABLE BORDER>
  <CAPTION ALIGN=top>Table 6: Birds</CAPTION>
  <TR><TH>Famous Birds</TH><TH>Color</TH></TR>  <TR>  <TD>   <OL>
  <I><LI> The Roadrunner</I>
  <I><LI> Big Bird</I>
  <I><LI> Chilly Willie</I>
  <I><LI> Tweety Bird <IMG SRC= "Tweety.gif"></I>
  </OL>  </TD>  <TD>   <UL>
  <LI> Green and blue    <LI> Yellow     <LI> Black and white    <LI>Yellow
  </UL>  </TD>  </TR>
</TABLE>
```

The Caption

Always designate the *<CAPTION>* element as the first command in your table. The caption is usually placed directly above the table in a separate font. You can also position the caption using the ALIGN attribute. For example:

```
<CAPTION ALIGN=bottom>Table 5. A neat table.</CAPTION>
```

would print the caption at the bottom of the table. To print the caption at the top, you'd use:

```
<CAPTION ALIGN=top>Table 5. A neat table.</CAPTION>
```

The Border

If you want to give your table a border, be sure to specify the *BORDER* attribute:

```
<TABLE BORDER>
```

STYLE TIP: In general, you can keep small tables free of borders. However, large tables can be hard to read unless each cell has a border.

Alignment

Data is usually centered within each cell. You can use the ALIGN attribute, however, to specify the horizontal alignment of cell headings or cell data:

HTML Left

HTML Center (the default)

HTML Right

Use *VALIGN* to specify the vertical alignment:

HTML Top

Google/YouTube Exhibit 1005/1105
Page 00305

HTML Middle (default)

HTML Bottom

For example, to place the words "Top left" at the top left corner of a cell:

```
<TD ALIGN=LEFT VALIGN=TOP>Top left</TD>
```

NOWRAP

Use NOWRAP to specify whether or not text within cells should wrap around. If you don't want text-wrapping, use markup similar to the following:

```
<TD NOWRAP>
```

Merging Cells

Two cells can be merged to form one cell; this way you can create one header cell for two columns. To make a cell span more than one row, use the *ROWSPAN* attribute; use *COLSPAN* to make the cell larger than one column. You can make any cell heading or cell data as wide or as tall as you wish.

For example, the code:

```
<TABLE BORDER>
  <CAPTION>Wide rows</CAPTION>
  <TR><TH ROWSPAN=2></TH>
      <TH ROWSPAN=2>NUMBER OF<BR>CONTESTANTS</TH>
      <TH COLSPAN=2>SCORE</TH>
  </TR>
  <TR><TH>HIGHEST</TH>
      <TH>LOWEST</TH>
  </TR>
  <TR><TH ALIGN=LEFT>MALES</TH>
      <TD>20</TD>
      <TD>101</TD>
      <TD>23</TD>
  <TR><TH ALIGN=LEFT>FEMALES</TH>
      <TD>823</TD>
      <TD>103</TD>
      <TD>12</TD>
</TABLE>
```

produces the table in Figure 11-10.

In other words, if you designate a cell with ROWSPAN=2, then it becomes two units high, merging with the cell directly below it. You only need to create one heading for a set of merged cells. If you make a cell with COLSPAN=2 then it becomes two units wide, merging with the cell to the immediate right. Your Web browser automatically centers all headings, making merged cells easy to read.

Google/YouTube Exhibit 1005/1105
Page 00306

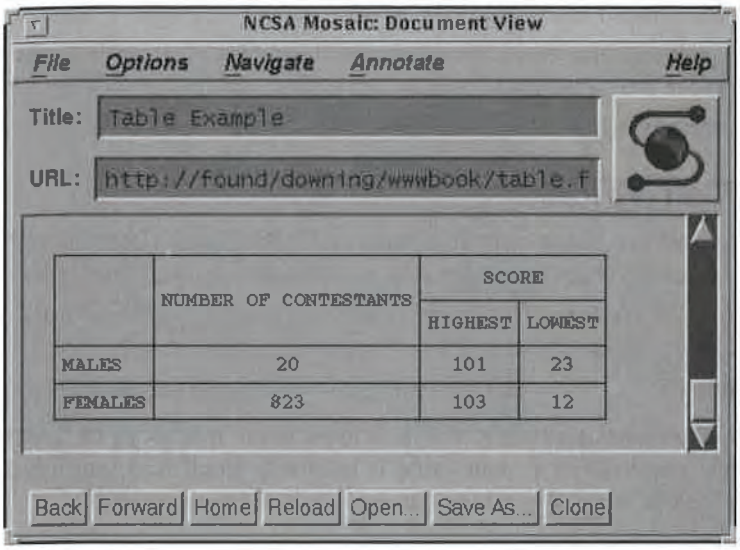


Figure 11-10 A table with merged cells

In cases where you have labels along the top and side of a table, you can create an empty cell in the upper-left corner just by typing <TH> twice.

When merging a column of cells, you can force an entry to take up more than one line by using the <BR> element.

STYLE TIP: Be sure you don't create row and column spans that cause cells to overlap. The resulting table is usually pretty hard to read.

Netscape's Table Extensions

Netscape creates lovely-looking tables, with three-dimensational borders, colors, and tight layouts. Some additional Netscape attributes let you customize the look of a table even further. For example, Mozilla HTML allows you to assign the table's BORDER attribute a value, giving different tables different looks. To create a table with a border four units thick:

```
<TABLE BORDER=4>
```

Spacing and Padding

Netscape also allows a CELLSPACING and CELLPADDING attribute as part of your <TABLE> element. By default, there are two units of space between each cell. If you want to tighten or loosen up a table, you can use markup similar to the following:

```
<TABLE CELLSPACING=4>
```


Google/YouTube Exhibit 1005/1105
Page 00307

Padding is the space between a cell's border and it's contents. Netscape usually pads a cell by one unit. If you want your cell's contents to be surrounded by more free space, you could use markup similar to:

```
<TABLE CELLPADDING=3>
```

Table or Cell Width

Finally, Netscape allows you to set the exact WIDTH of your table or of each cell. You can set a value (in pixels) or a percentage. For example, if you want your table to go across half the Web page, you could use the command:

```
<TABLE WIDTH=50%>
```

STYLE TIP: Usually Netscape tries to create a table width that looks as good as possible. Only use WIDTH if your table is relatively small and you have an exact layout look in mind. Big tables can end up looking squashed.

You can also use the WIDTH attribute within the <TH> or <TD> elements. This lets you determine the size of a given cell. If you want a cell to take up a quarter of the table, you could use the markup:

```
<TD WIDTH=25%>Big ol' Cell</TD>
```

Design Hints

You might find it helpful to design your tables using a sort of graph, as shown in Figure 11-11. Each square on the graph is a cell. The thick lines represent the sizes of each cell. The bold print represents headings. You can then go through each cell with HTML commands, using the appropriate COLSPAN and ROWSPAN attributes and filling the cells with appropriate data. Each time you design a cell, cross it out, along with any other affected cells in the table. This prevents you from declaring the same cell twice, skewing the entire table.

For example, here's the code for a table based on the graph in Figure 11-11. Note that the caption doesn't show up on the graph, because it doesn't affect cell size or contents.

```
<TABLE>
  <CAPTION ALIGN=bottom>Each month's winners.</CAPTION>
  <TR><TH ROWSPAN=2><TH COLSPAN=2>SIZE<TH ROWSPAN=2>BEST<BR>SCORE
  <TR><TH>HEIGHT<TH>WEIGHT
  <TR><TH>JOHN<TD>5'10"<TD>150<TD>983
  <TR><TH>SARA<TD>6'0"<TD>164<TD>901
</TABLE>
```


Google/YouTube Exhibit 1005/1105
Page 00308

	Size		
	Height	Weight	Best Score
John	5'10"	150	983
Sara	6'0"	164	901

Figure 11-11 Designing a complicated HTML table

LESSON #13: HYPERTEXT

Let's review how hypertext works: You click on a word or phrase or image and then you're magically transported to a wonderful new world. In other words, the Web browser needs to know two pieces of information about each hypertext link:

HTML Which part of the document is the link?

HTML Where does the link lead?

The answer to the first question is the *<A> link* element. The answer to the second question is a URL (covered in detail in Chapter 1) and an *anchor.*

The Anchor Element

The anchor element puts the hyper in hypermedia. A link designates that the next piece of text or graphics is actually a hypertext doorway to a strange, new, and wonderful place. You designate an anchor or link using the

```
<A>
```

element.

Google/YouTube Exhibit 1005/1105
Page 00309

When your link text is finished, you designate that using the closing element:

```
</A>
```

Link text can take many formats. In some cases it is the exact address, file name, or location of the place the user will be zapped to, as in the top half of Figure 11-12.

In most cases, though, a link can be much more subtle, as in the bottom half of Figure 11-12. The word "aquarium" can whisk the user to an online store specializing in aquariums and tropical fish supplies. The style of your document link is completely up to you.

STYLE TIP: Try to create links that don't muck up the flow of the text. People may convert your hypertext document into a plain text document so they can refer to it offline. Your text should stand on its own. You might want to avoid words like "click your mouse on this sentence to read about the history of skyscrapers." Every Web user should know how to click. You're also being "mouseist"– some sensitive person who doesn't use a mouse may be offended. Instead, say something like, "There's an excellent article on the history of skyscrapers by T.O. Tall"–and make the word "skyscraper" your link.

Links do not have to be plain body text. You can generally turn any sort of text–headings, lists, italicized or bold text, or anything else into a link. It's perfectly fine to put an anchor within another HTML element. For example:

```
<H2><A HREF = "fishstore.html">Fish Stores</A></H2>
```

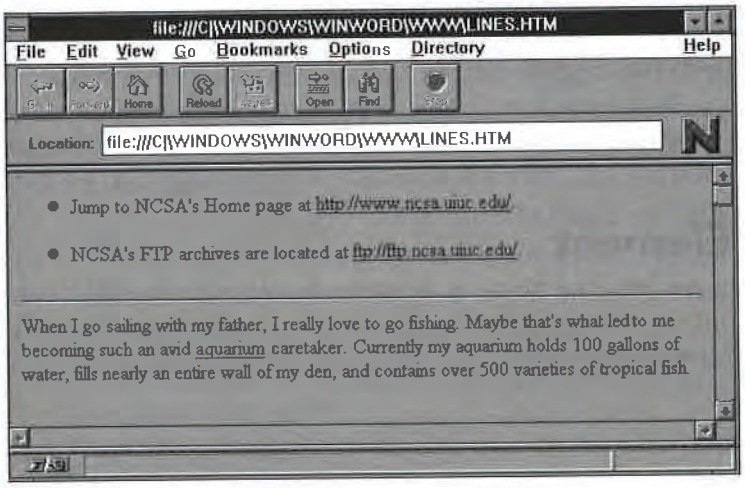


Figure 11-12 You can make anchors large and obvious (top) or subtle (bottom)

Google/YouTube Exhibit 1005/1105
Page 00310

You should not, however, embed other HTML elements within an anchor, since some browsers may become confused. This is a no-no:

```
<A HREF = "fishstore.html"><H2>Fish Stores</H2></A>
```

You can even make a graphic into a hyperlink. The next chapter talks about how to do this.

The anchor element has a number of optional attributes, including:

HTML *HREF:* jump to another Web page or resource

HTML *NAME:* Destination for jumps from another part of the current document

HTML *TITLE:* Title of document

HTML *REL* and *REV:* Relationship between objects

HTML *METHODS:* How to link

An anchor is meaningless, however, unless it has either an HREF or a NAME.

Links: HREF

The HREF attribute is the hyperlink reference: the place where you want to jump to. The anchor takes the form

```
<A HREF="url.html">clickable link text goes here</A>
```

Replace "url.html" with the full URL of the destination document (inside the quotes). Any text or graphics between the <A> and the </A> becomes an anchor, and most browsers will highlight it in blue. The user can then click on this blue text and automatically jump to the HREF location. See Figure 11-13 for a breakdown of a complete link.

WARNING: Do not create an anchor that doesn't lead to a valid location. Users will get confused and fed up clicking on links that don't exist. If you're not sure of a URL, you can either use empty quotes or just underline your link text: <U>Link Text</U>. You can then come back later and easily insert the real link. Also be sure to periodically check your remote links, since hypertext can often go *stale* as pages move or are erased.

HREFs often include *absolute URLs*—the specific addresses of distant Web documents, pictures, sounds, FTP files, gopher menus, telnet sites, newsgroups, e-mail, WAIS databases, and more. Any URL on the Internet

Google/YouTube Exhibit 1005/1105
Page 00311

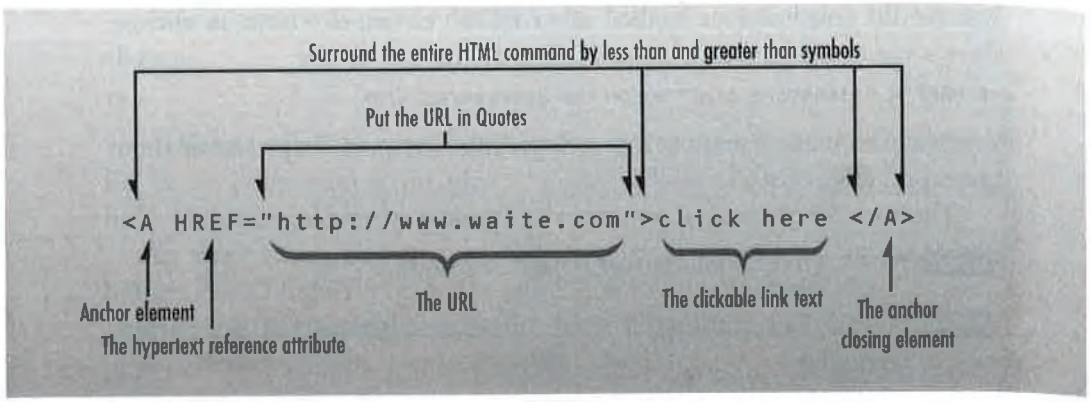


Figure 11-13 The elements in a hypertext link

can be jumped to. Most URLs are https, leading to another Web server. For example:

```
<A HREF="http://www.cern.edu/dsd/sd">CERN's home page.</A>
```

You can also jump to an FTP site, however:

```
<A HREF="ftp://ftp.netcom.com">Netcom's public FTP archives.</A>
```

Or even telnet to a remote computer:

```
<A HREF="telnet://world.com">Visit the World system.</A>
```

Most HREFs, however, are expressed as *relative URLs*—identifying documents, images, or other files that are available in the current directory. You can also indicate a relative URL location if you'd like to jump to an HTML document that is stored somewhere on the current Web server. For example, you can have users easily return to your own home page by using the anchor:

```
<A HREF="index.html">Click here to return to the home page.</A>
```

Or, if the file is stored in a different directory, you could indicate the entire pathname:

```
You should visit my friend <A HREF="/web/smith/public-
html/home.html">John Smith's</A> home page.
```

NOTE: Unix directories are separated by slashes, whereas DOS directories are separated by backslashes. If your HTML server is on a Windows or DOS PC, express filenames the way you normally would. If you are using a Unix machine, be sure to use a Unix directory structure. You may need to check with your system administrator if you're interested in using files in a directory different from your own. See Chapter 21 for detailed information about Web servers.

Google/YouTube Exhibit 1005/1105
Page 00312

Relative URLS

When you can, stick to relative URLs. They're easier to type and easier to locate. All relative URLs consist of a simple filename. That file—whether it be a graphic, sound, movie, text file, or another HTML document—must be stored in the same directory as your Web page.

Relative URLs are especially useful if you must move your Web page to another location. For example, if your Internet provider goes out of business, you can transfer your entire Web directory to another machine. All relative URLs will still be valid—you won't need to retype a thing. Simply notify people of your new Web address.

When writing a relative URL, you don't need any slashes as long as the document you're linking to is in the current directory. If you call a URL that is in a different directory on the same machine, simply include the file's full pathname, which usually begins with a slash.

WARNING: Be careful not to use partial pathnames. Such a setup may load up when you use it, but won't work for remote users.

Absolute URLS

You should link to an absolute URL when:

HTML You didn't create the document being jumped to.

HTML The document, for space or copyright reasons, is stored on a distant server.

HTML You're linking to a document that often moves around.

In most cases, you create a link to a distant URL because it fits in with the theme of your page. For example, if I'm interested in tango dancing, I may include a link to the official Tango Page in France. If you're involved with fractal research and there's an interesting movie about fractal animations in a colleague's directory at another university, use that person's complete URL.

STYLE TIP: Don't be shy when it comes to using absolute URLs. Distant hyperlinks are what makes the Web a web. Many people create a special Web page with links to their favorite places. In fact, you can convert your Lynx, Netscape, or Mosaic bookmark lists into HTML. Turn to Chapter 17 to find out how.

An absolute URL must take the form

```
protocol://www.domain/directory/filename
```

See the section on URLs in Chapter 1 for details.

Google/YouTube Exhibit 1005/1105
Page 00313

Anchors: The Name Attribute

Suppose you have a hundred-page document. Are you going to force readers to scroll down through the entire thing if they're only interested in one section?

The NAME attribute turns the anchored text into an actual hypertext destination. In other words, if you designate a piece of text with a certain name, you can build in explicit jumps to it. This attribute takes the form:

```
<A NAME="label">Optional Text</A>
```

In the corresponding HREF, precede the anchor name with the pound sign (#), so the attribute takes the form:

```
<A HREF="#label">Put your jump text here</A>
```

Once you designate a named anchor, you can jump to it from anywhere within the document (an *internal link*), or even from another document (an *external link*). See Figure 11-14.

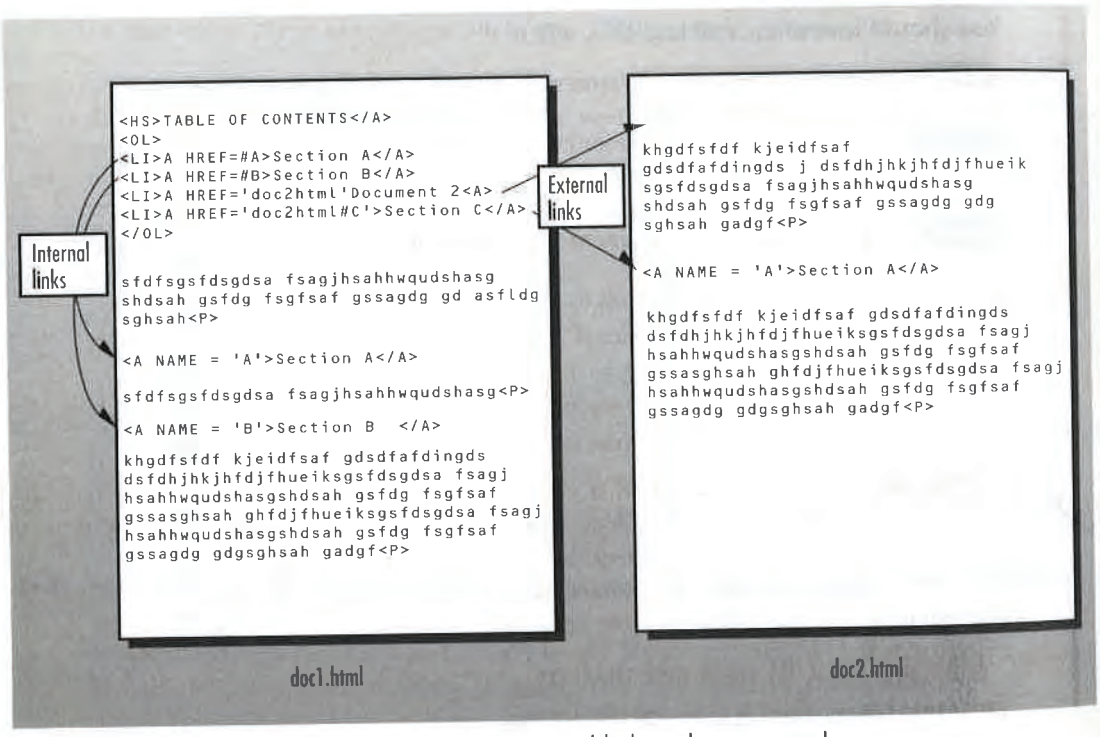


Figure 11-14 The difference between an internal link and an external one

Google/YouTube Exhibit 1005/1105
Page 00314

STYLE TIP: Since some browsers can get confused by messy anchors, be sure to create anchors at the left margin of a given line. Try to limit your anchors to one word. Anchor names are case sensitive, so be sure to be consistent.

Internal Links: Creating a Table of Contents

The NAME attribute comes in most useful if you set up a table of contents as shown in Figure 11-15. Each line of this table can be a link to a heading that appears later in the same document.

For example, if you have a heading buried deep in some long document:

```
<H2>Judaism</H2>
```

you can mark it as a link by giving it an anchor name:

```
<H2><A NAME="jew">Judaism</A></H2>
```

You can then create a jump from the table of contents by setting up the entries there in a form like this:

```
<OL>
  <LI><A HREF="#christ">Christianity</A>
  <LI><A HREF="#jew">Judaism</A>
  <LI><A HREF="#hindu">Hinduism</A>
</OL>
```

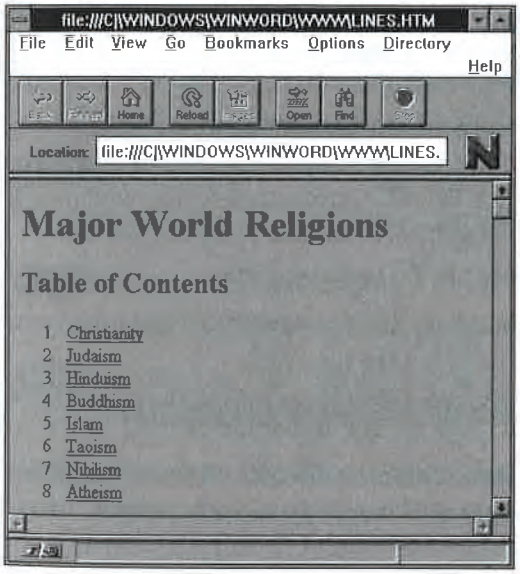


Figure 11-15 A hypertable of contents

Google/YouTube Exhibit 1005/1105
Page 00315

When someone clicks on the word Judaism in your table of contents they'll be warped to the appropriate section of your document.

NOTE: The <LI> element precedes the <A> link because most tables of contents are created using ordered lists.

External Links: The Guts of Another Document

Normally, when you jump to another document using the HREF attribute, you are taken to the beginning of the text. If the name attribute is also included, however, you can begin reading a document from any particular point. This is very useful. Simply append the *#NAME* text to the end of the URL.

Suppose, for example, you're creating a document about the Middle East. It might be useful to create a link to your document about Major World Religions. When you got to talking about Israel's Jewish populace, for example, you could include the following hyperlink:

```
<A HREF="religions.html#jew">Click here to read some more about
Judaism</A>.
```

Assuming there's a properly labeled <A NAME="jew"> section in the *religions.html* document, your readers will be warped directly there.

STYLE TIP: If you have many Web pages dealing with the same topic, you may want to create one master Table Of Contents page with external links to each section of each document. You could then include a button at the bottom of every Web page that returns the user to your table of contents. This makes a lot of material easy to flip through.

You can even create external links to distant absolute URLs. For example, if you find a huge article about marine life on some distant system, you can scan that article's HTML source code until you find a NAME tag. For example, you may find:

```
<H1><A NAME="sharks">Sharks: Deep Sea Killers</A></H1>
```

If you happen to be discussing sharks within your own Web page, you can create a link to the shark section of the marine life article:

```
An <A HREF="http://www.sea.org/marinelife.html#sharks>excellent
article</A> about sharks is published by the SEA Organization.
```

NOTE: In the latest HTML+ specification, the NAME attribute has been changed to ID, and is usually a part of the <P> paragraph marker, rather than the <A> anchor. For instance:

```
<P ID="jew">
```


Google/YouTube Exhibit 1005/1105
Page 00316

However, for the time being, most browsers do not support this. You should stick to the NAME attribute.

Title

The *TITLE* attribute is optional, and usually not necessary. TITLE should be used only when creating a link to a non-HTML document. Every HTML document should have its own title. However, telnet sites, Gopher menus, FTP files, and text files do not have official Web titles. Inserting a title in the anchor displays a title at the top of the browser's screen.

This takes the form:

```
<A HREF="plaintext.txt" TITLE="A Plain Text Document">Click here</A>
```

A TITLE is usually informational, acting more as a comment than a useful tag.

Defining Relationships

The REL and REV attributes are used within advanced HTML documents. These two markers define the exact relationship between the linked document and the current document. REL and REV are only used *in addition to* the HREF attribute.

Most hypertext links consist of a *source document* (yours) and the *destination document* (the URL specified in HREF).

REL

The REL attribute expresses a forward relationship between the source and the destination. You can use REL as follows:

```
<A HREF="http://www.here.there/path/crazy.html" REL="Useindex">Click
here</A>
```

Here are the most useful relationships you can create between documents:

HTML Acyclic: You cannot return to the source document from the destination document.

HTML Annotation: The destination document just contains notes, edits, or small additions to the source document.

HTML Approves: The destination document has been approved by the source.

HTML Embed: The destination document is included in the source document. Some browsers use this to display two documents at once.

Google/YouTube Exhibit 1005/1105
Page 00317

HTML History: The destination document lists the many versions of the source.

HTML Includes: The source document includes the destination within some sort of group.

HTML Interested: The owner of the source document is interested in the destination.

HTML Made: The source document is a home page or description of the destination's author.

HTML Owns: The person who owns the current source document also owns the destination document.

HTML Precedes: The source document precedes the destination document. This is useful if you have a book or manual consisting of a string of many Web pages. Often, you can click the browser's Next or Previous arrows to navigate. Each document can be preceded by only one other document.

HTML Present: Whenever the source document is presented, the destination document must also be retrieved.

HTML Refutes: The destination document refutes an argument made in the source. See Supports.

HTML Reply: The destination document is a reply to a question or issue raised in the source.

HTML Search: The destination document should be searched, not shown. Most browsers will ask the user to type in a search term. This allows you to create an index or database.

HTML Supersedes: The destination document is a previous version of the source.

HTML Supports: The destination document supports an argument made in the source. See Refutes.

HTML UseGlossary: The destination has the glossary definition of terms that are listed in the source.

Google/YouTube Exhibit 1005/1105
Page 00318

HTML Useindex: The destination document is a search index.

Most REL tags are pretty much arbitrary and can be used by you to keep track of your Web documents.

REV

The REV attribute is the exact opposite of REL. For example, a link from the source to the destination with:

```
<A HREF="document2.html" REL="Precedes">Click here to view the next
document in the series.</A>
```

Is the same as a link from the destination to the source with:

```
<A HREF="document1.html" REV="Precedes">Click here to view the
previous document in the series.</A>
```

Methods

You almost never need to worry about this attribute. The METHODS tag tells the httpd server which method types the linked-to URL supports.

LESSON #14: RELATIONSHIPS AND HYPERPATHS

You now know how to jump to another document. But what if you want the jump to be automatic? You can create an ordered list of several documents, forming a sort of guided tour of your Web pages.

Once you create a hyperpath, a user would have two ways of accessing adjacent documents along the path:

HTML Clicking on the anchor, as usual

HTML Pressing the Next or Previous button in the browser's toolbar

STYLE TIP: Many browsers won't support forward and backward links. If you're presenting a multipage document, it's always a good idea to include a set of buttons or phrases at the bottom of your page that can take the user to and fro. For example:

```
[Table Of Contents] [Next] [Previous]
```

Which can be created using HTML similar to the following:

```
<A HREF="toc.html" REL="Precedes"> [Table Of Contents] </A>
<A HREF="page5.html" REL="Precedes"> [Next] </A>
<A HREF="page3.html" REV="Precedes"> [Previous] </A><P>
```


Google/YouTube Exhibit 1005/1105
Page 00319

Using Link

HTML has a <LINK> element which uses the REL and REV attributes, similarly to the <A> element. For example:

```
<LINK HREF=URL REL="next">
```

The REL attribute takes arguments similar to:

- Contents: The destination document is a Table of Contents.
- Next: The destination document is the next document in the sequence.
- Previous: The destination document is the previous document in the sequence.
- Parent: The destination document is the parent of the current document.
- Help: The destination document has lots of help for topics in the current document.

The best way to make one document automatically follow another is to use the <LINK> element's REL attribute. The LINK is usually placed in the document's <HEAD>. For example, if you have three documents: doc1.html, doc2.html, and doc3.html, you could put these links at the beginning of your third document:

```
<HEAD>
<TITLE>Sample Document 3 of 10</TITLE>
<LINK HREF="doc2.html" REL="previous">
<LINK HREF="doc4.html" REL="next">
<LINK HREF="toc.html" REL="contents">
</HEAD>
```

and this set of links at the beginning of document 4:

```
<HEAD>
<TITLE>Sample Document 4 of 10</TITLE>
<LINK HREF="doc3.html" REL="previous">
<LINK HREF="doc5.html" REL="next">
<LINK HREF="toc.html" REL="contents">
</HEAD>
```

You could continue doing this for all the documents. Some browsers then allow you to press the Next or Previous buttons to cycle between these ten documents. In the future, browsers may even include an automatic Table of Contents button.

Google/YouTube Exhibit 1005/1105
Page 00320

Nodes

The latest specification of HTML also has commands that will make it easy to develop complex hypertext paths. To create stepping stones along a hypertext river, use the <A> anchor's *NODE* attribute. You can create as many nodes as you like. You usually want to put your nodes in a list for a user to see:

```
<UL>
  <LI><A HREF="node1.html" NODE>The first stop on our tour.
  <LI><A HREF="node2.html" NODE>The second stop.
</UL>
```

A browser reads these nodes, in order, and places them in its own list. A user can then use the Next and Previous button to cycle between the nodes.

NOTE: Modern browsers do not yet support the NODES command.

Path

Eventually, you'll be able to use NODES to create a special HTML document which lines up a specific path. If you want to call this path document, you could use the anchor's *PATH* attribute. For example, if you have a set of paths listed in the file paths2.html, you could use the command:

```
<A HREF="paths2.html" PATH>Another path</A>
```

The second path would then be added onto any existing nodes.

LESSON #15: MATHEMATICAL EQUATIONS

As of now, there are no widespread Web browsers that can handle equations. However, the proposed new HTML version 3 format has some great mathematical additions. Since the next generation of browsers will most likely implement these elements, it's worth going over them.

HTML+ has elements for all basic mathematical functions. To designate an equation, you would type the *$* element. To end the equation, use *$*.

A browser that supports mathematical symbols would have a set of common escape characters, such as:

HTML *∫*: Integral symbol

HTML *∞*: Infinity

HTML *&tanh;*: Tangent

Google/YouTube Exhibit 1005/1105
Page 00321

HTML *α*: Greek Alpha

HTML *β*: Greek Beta

HTML *θ*: Greek Theta

HTML *κ*: Greek Kappa

You can create an equation, then, using a command similar to the following:

```
<math>
  F(t) = &int;<sub>n</sub><sup>&infin;</sup> j<sup>t</sup> dt
</math>
```

which would create the equation in Figure 11-16. The subscript and superscript would automatically center the appropriate numbers over and under the integral sign.

Netscape and Mosaic already support <SUB> subscripts and <SUP> superscripts, making it possible to create simple exponents and such.

The HTML+ math specification also calls for the *<BOX>* element to group a set of numbers of equations together. The *<OVER>* element could be used to create fractions with a numerator and a denominator. An *<ARRAY>* element would print out nice-looking matrices. A *<ROOT>* element would be used to express various roots.

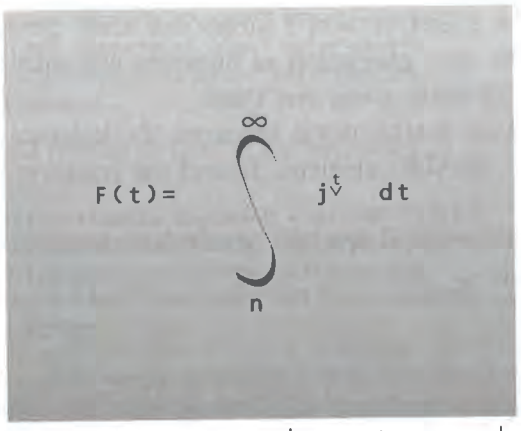


Figure 11-16 A sample equation created by HTML+

Google/YouTube Exhibit 1005/1105
Page 00322

You can read more about HTML+ math commands and other HTML+ specifications by jumping to:

http://www11.w3.org/hypertext/WWW/MarkUp/HTMLPlus/

LESSON #16: DYNAMIC DOCUMENTS: PUSH OR PULL?

The latest version of Netscape Navigator supports dynamic documents by using the *server push* and *client pull*. This lets your Web browser receieve new data periodically. For example, a Web page with the latest weather map could be updated every ten minutes, or a Web page with stocks could be updated any time a stock's price changed. A video camera can even be used to send the latest frame to your Web browser, forming a live animation or movie.

Server Push

Server push is when the Web server delivers information to a Web page. Unlike a standard Web page, however, the connection between the server and the Web client remains intact. This way, whenever the server decides to update information, the Web page is immediately reloaded.

You can use server push to create very detailed sets of dynamic documents. You can even dynamically change a single image within a Web page, providing video or animation (albeit with a very slow frame rate) within an otherwise unchanging page.

To find out more about how to create CGI scripts which promote server pushing, check out the following URL:

http://home.netscape.com/home/demo/1.1b1/pushpull.html

Client Pull

Client pull is similar to server push, but gives the Web page itself more power. A Web page can be programmed to tell a server to reload its data after a given amount of time. The server waits and then automatically gives the Web page the new information.

Client pull is accomplished by using the HTTP-EQUIV="Refresh" attribute in the META element (an HTML3 element that simulates a document's header). The number of seconds to wait is specificed using the CONTENT attribute. For example, the following markup should be the first line of your HTML document if you want the Web page to reload itself after 3 seconds:

```
<META HTTP-EQUIV="Refresh" CONTENT=3>
```


Google/YouTube Exhibit 1005/1105
Page 00323

The Web page will continue to reload every three seconds.

NOTE: You can use a CONTENT of 0 seconds to load up the given document as quickly as possible.

To load up a different document after a set amount of seconds, simply specify a full URL using the URL attribute. For example, to load the page at *http://www.smartpants.edu/sample.html* after a minute, use the following markup as the first line in your HTML document:

```
<META HTTP-EQUIV="Refresh" CONTENT=60
URL=http://www.smartpants.edu/sample.html>
```

You can use this trick to flip between two documents, animate a sequence of images, or create an automatic "slide show" of many Web pages.

NOTE: To stop a dynamic document while it is loading, a user needs to either close the window, click on a hyperlink, or use the history list to return to a previous document.

LESSON #17: POLISHING

Proofread your pages for spelling and grammar errors. The page will be seen by millions of adoring fans around the world. Don't publish something flawed.

You should also double-check your HTML syntax, keeping a sharp eye out for these common problems:

- Unpaired elements—that is, start-something elements with no corresponding closings (it's easy to forget the slash "/" in </H1>).
- Missing opening or closing quotes around URLs (<A HREF="http://www.sillyme.com>)
- Never-ending comments (<!--How come nothing else on my Web page is getting displayed?!?!?)
- HTML elements with no closing greater-than bracket (<A NAME="hello")
- Using the less-than, greater-than, or ampersand sign as part of your text (<, >, &).
- Special characters with no semicolon (&gt)

Google/YouTube Exhibit 1005/1105
Page 00324

HTML URLs with uppercase letters when the system wants lowercase (<A HREF="HTTP://A.EDU/C/D/E/FILE.HTML">) or vice versa

There are several programs and services that run through your HTML code, checking for such errors. You can read about them in Chapter 20, Where to Place Your HTML Documents.

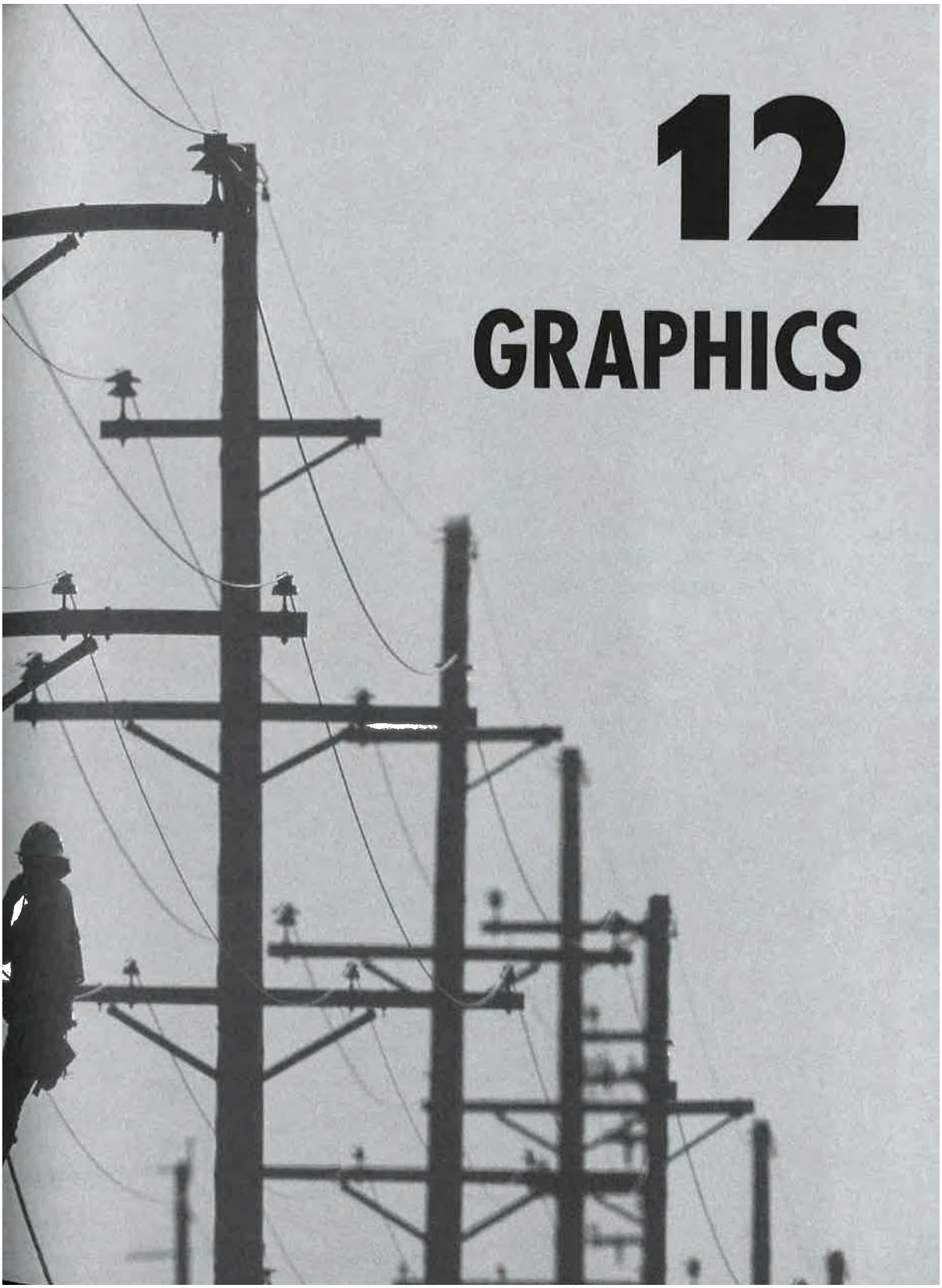
STYLE TIP: Remember, a Web page is always a work in progress. Polish it and polish it again 'til it shines.

WHAT NOW?

Your Web page already looks pretty good, doesn't it? Well, just wait until you add graphics. Flip to the next chapter. Now that you can create hyperlinks, you can read about interesting things to link to in Chapter 15.

If you can hardly wait to put your pages on the Web, check out Chapter 20, Where to Place Your HTML Documents.

Google/YouTube Exhibit 1005/1105
Page 00325



12 GRAPHICS

12

A magazine without illustrations of any kind. A newspaper devoid of news photos. A report empty of graphs. Plain, cold text against a plain white background. In a word: Boring.

Before the Web, there was no easy way of viewing graphics over the Internet. Not only do images add spice to online text, but they often replace it. Here's just a tiny sample of common Web graphics:

- HTML Huge billboard-like display ads
- HTML Logos
- HTML Borders, lines, or other page-enhancing designs
- HTML Signatures
- HTML Icon buttons
- HTML Custom fonts
- HTML Photos, drawings, paintings, or other helpful illustrations
- HTML Charts, graphs, and tables

Basically, using graphics, a Web page can look exactly the way you want it. If you can sketch, scan, or design it, the Web can shoot it all over the world. This chapter covers everything you need to know about grabbing, creating, converting, designing, and placing graphics over the Web.

GRAPHIC FORMATS

There are dozens of ways to store graphics. Some formats deal with black and white images, others can hold photographic-quality pictures, while others contain video. See Table 2-1 in Chapter 2 for a list of the most common graphic types. Since graphics are generally stored as a plain sequence of numbers, they can easily be shared across various types of computers and operating systems. Graphics viewers can usually read in several types of images and then write out the image in the format of your choosing.

The Web generally deals with two types of images: GIFs and JPEGs.

GIF

The Graphical Interchange Format is a very basic table of colors. This table can hold most any resolution of graphics, though it is limited to 256 colors. Since GIFs use a very modest form of compression, GIFs are generally quite large. You should use GIFs for the following types of images:

- HTML Icons, buttons, or bullets for lists
- HTML Small illustrations

Google/YouTube Exhibit 1005/1105
Page 00328

HTML Graphical lines or separator bars

HTML Charts, tables, or graphs you want to place alongside your Web text

HTML Thumbnails (see the Thumbnails section, later in this chapter)

JPEG

The JPEG format squeezes and compresses images, and thus graphics take up much less room to store (though they take longer to interpret). JPEG can also deal with images containing millions of colors. JPEG is known as a lossy format because the compressed images generally lose some data and sharpness. The general rule of thumb is to use JPEGs for:

HTML Photographic-quality or other "real life" images

HTML Large images

HTML Complex images that use more than 256 colors

HTML Images you don't need to include directly on the Web page

When you store a JPEG, you can usually designate the amount of compression you'd like to use. The more you compress, the more data the image loses.

GRAPHIC VIEWERS

Every multimedia browser can display some images alongside blocks of text. However, there are many types of graphics that your browser can't load up on-the-fly. To see these you'll need a separate graphic viewer.

Graphic viewers can do more than display images, they can help manipulate them. Before you start to work with graphics, you should install two important Windows viewers: LView and WinGIF. Each browser allows you to configure viewers for popular types of files.

LView

The most popular and versatile Windows viewer is probably LView Pro, by Leonardo Loureiro. LView can only run on a 386 or better, and a Super VGA card and monitor are recommended. There is also a commercial LView version that needs a 486 running 32-bit Windows or Windows NT. LView can read or write the following graphics formats:

Google/YouTube Exhibit 1005/1105
Page 00329

HTML .JPG JPEG JFIF

HTML .GIF GIF87a GIF-89a

HTML .TGA Truevision Targa

HTML .BMP Windows and OS/2

HTML .PCX Zsoft and Microsoft Paint Pixelmap

HTML .PPM and .PGM by PBMPLUS

HTML .TIF TIFF

To install LView Pro, copy the entire \WINDOWS\VIEWERS\LVIEWP directory from your CD-ROM and place it on your own hard disk. You can now configure Mosaic or Netscape to use LView as the native graphic viewer. You should also set up an LView icon in the Windows Program Manager, since it's a program you'll probably use often.

Tweaking the Image

Depending on the image you load and the type of video hardware you have, you may want to adjust LView's settings. Load up a sample image by selecting File, Open. Select the graphic format by clicking on the List Files Of Type box.

If the image doesn't look very good, this may just be a limitation of Windows. Try tweaking the image to look its best by selecting tools from the Retouch menu. You can perform a Gamma Correction (if all colors seem slightly skewed), an Interactive RGB session, Color Balance, Contrast, etc.

Flipping Out

LView makes it easy to rotate, flip, or even take the photographic negative of an image. Select Flip Horizontal!, Flip Vertical!, Rotate Left!, or Rotate Right! from the Edit menu.

Adding Text

Another useful feature of LView is the ability to add text onto an image. This lets you scribble in company names beneath logos, descriptions on top of buttons, or captions or credits alongside photos.

Using the mouse, click on a spot where you want your text to begin. Holding down the mouse button, create a rectangle to show LView where

Google/YouTube Exhibit 1005/1105
Page 00330

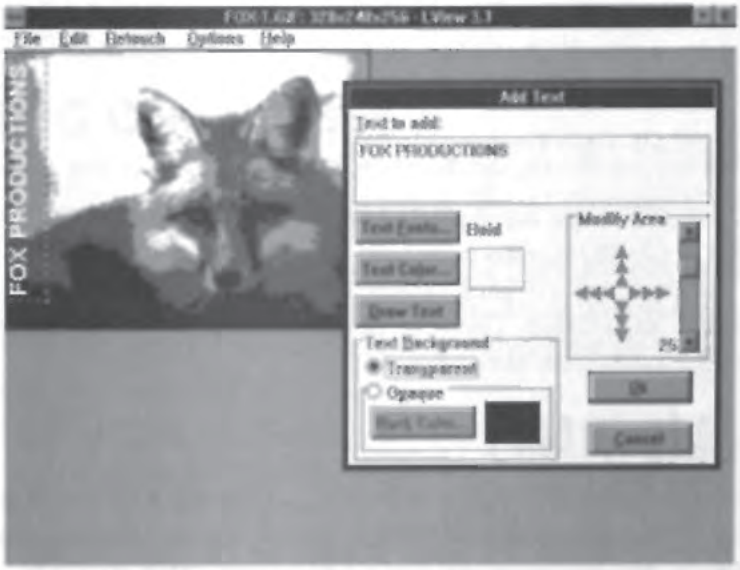


Figure 12-1 Putting text on an image

your text should be. Let go of the mouse button. Select Edit, Add Text. The Add Text dialog appears, as in Figure 12-1.

Click Text Fonts to choose a font's Name and Points size. Select the Bold, Italic, Underline, or Strikeout style if desired. You can also align the text Left, Right, or Centered. If you like, you can even create vertical text. Click the Vertical Up button to create vertical text that runs up the screen, click Vertical Down for text running downward.

Click Foreground and pick any text color from the palette. In most cases you want your text's Background to be Transparent; be sure Transparent Text Background is checked. If not, click Background and select the text's background color. As you play around with different ideas, click Exec to preview what the print will look like. When the text appears the way you want, click OK.

To finalize the text, click anywhere on your image.

WinGIF

WinGIF is an easy-to-use graphics viewer that also contains some basic image manipulation commands. WinGIF can handle most popular graphics formats:

HTML .GIF GIF 87a GIF89a

Google/YouTube Exhibit 1005/1105
Page 00331

HTML .BMP Windows

HTML .RLE

HTML .PCX Microsoft Paint Pixelmap

To install WinGIF, copy the entire \WINDOWS\VIEWERS\WINGIF directory from the CD-ROM onto your hard disk. WinGIF is a basic, no-frills viewer. If the image on your screen is difficult to make out, try selecting Options, Quick Dither. This uses Windows colors to emulate what the image should actually look like.

LESSON #1: INLINED IMAGES

Inlined images are any sort of graphics embedded within the Web text. For example, Figure 12-2 contains several inlined images. The photo, the icon buttons, the bullets, and the horizontal line are all custom-made graphics.

Almost all inlined graphics are in the popular GIF format, though some are stored in the JPEG or X-Windows binary map style. Every multimedia browser can display GIFs along with text, and most can display XBM images. Some browsers, like Netscape, can also inline JPEGs. Other graphic formats

Figure 12-2 A Web page just wouldn't be much of a page without inlined graphics.

Google/YouTube Exhibit 1005/1105
Page 00332

can be downloaded and shown using a viewer. Chapters 4, 5, and 6 discuss how to install a graphical viewer with SlipKnot, Mosaic, or Netscape.

The IMG Command

Once you have a nice collection of GIF images, choose the ones you'll want to use. Copy all the necessary image files to the same directory as your HTML document.

To insert an image in your Web page, use the HTML *<IMG>* command at the place you want the image to appear. Designate the name of the source image file using the SRC attribute. For example, typing

```
<IMG SRC="image.gif">
```

immediately displays the image in the IMAGE.GIF file.

The "image.gif" filename is a relative URL and must be located in the current directory. You can also use images from different directories, by designating the entire pathname:

```
<IMG SRC="/design/icons/image.gif">
```

Distant Images

Using full URLs, you can even grab images from anywhere else on the Internet. Suppose you're designing a Web page about astronomy. You browse around and find a great photo of Jupiter at the Smartpants University archives. If the file is publicly available, you can request it, download it, and place it in your own directory. Otherwise, you can just insert it using this command:

```
<IMG SRC="http://www.smartpants.edu/astronomy/planets/Jupiter.gif">
```

NOTE: Since distant images involve extra Internet connections, loading them is generally slower than using a local image.

Hyperimages

One of the nicest things about inlined images is the ability to use them as buttons. You can develop your own interactive screens full of icons. To create a hyperimage, use the HTML anchor command as usual, inserting an image in the middle:

```
<A HREF="jumphere.html"> <IMG SRC="jump.gif"></A>A really cool Web
page.
```

Most browsers draw a blue border around a hyperimage to let people know they can click it.

Google/YouTube Exhibit 1005/1105
Page 00333

Figure 12-3 A set of buttons with border sizes ranging from size 1 to 5

Crossing the Border

More advanced versions of HTML, such as Netscape's Mozilla, allow you to specify the size of an image's border using the BORDER attribute:

```
<IMG SRC="image.gif" BORDER=2>
```

Generally, Web browsers automatically draw blue borders of size 1 around hyperimages. If you want, however, you can draw a border around any image at all. Various sizes of borders can add a nice cosmetic effect to your Web pages. For example, if you have several buttons on your page, you may want to surround the default button with a thicker border than the others. Figure 12-3 shows some sample borders.

NOTE: If you set a hyperimage's border to 0, users have no way of knowing that the image is, indeed, an anchor. You may or may not desire this effect. For example, if the image is quite obviously a button, most users will try pressing it, whether or not it's surrounded by a blue border.

LESSON #2: ALIGNMENT

Sprinkling images around your text is definitely a great way to snazz things up. In most cases, an image is accompanied by some sort of text. This may be a caption, a heading, or just the body of an article. If you randomly insert an image, there's no telling how adjacent text will appear.

Most browsers allow you to suggest an alignment location for your inlined images. All you need is the *ALIGN* attribute:

```
<IMG ALIGN="top" SRC="/image.gif">This text appears next to the top of
the image.
```


Google/YouTube Exhibit 1005/1105
Page 00334

Figure 12-4 Top, middle, and bottom-aligned images

The adjacent text then appears toward the top of the image. Likewise, you can specify *ALIGN="bottom"* or *ALIGN="middle"*. By bottom-aligning an image, text appears next to the bottom of the image:

```
<IMG ALIGN="bottom" SRC="image.gif">This text appears next to the
bottom of the image.
```

Similarly, you can middle-align the text:

```
<IMG ALIGN="middle" SRC="image.gif">This text appears next to the
middle of the image.
```

Figure 12-4 shows three images, each using a different type of alignment.

Absolute Alignment

If you want, you can get *really* precise with your alignment options. This is useful if you're developing a Web page that uses several images nestled together. You may also want to use absolute alignment if you're dealing with fancy text made up of various fonts or styles.

NOTE: Only advanced browsers such as Netscape support these additional ALIGN attributes.

The absolute alignment attributes are:

Google/YouTube Exhibit 1005/1105
Page 00335

HTML *texttop:* Aligns the image with the top of the tallest piece of *text* in the given line. This is different from the standard *ALIGN="top"* attribute, which lines up the image with the tallest *item* in the line, whatever it may be.

HTML *absmiddle:* This lines up the exact middle of the current line with the exact middle of the image.

HTML *baseline:* Essentially the same as *ALIGN="bottom"*.

HTML *absbottom:* Aligns the exact bottom of the image with the exact bottom of the current line.

For example, the command

```
<IMG ALIGN="absmiddle" SRC="photo.gif"><IMG ALIGN="absmiddle"
SRC="photo2.gif">
```

is a great way to precisely align two images to each other.

Floating Alignment

Most versions of HTML+, including Netscape's Mozilla, support more useful types of image alignment. These alignment techniques make use of a *floating image*, which allows you to align the image itself, not just the surrounding text. Your graphic floats in the area you specify (against the right margin, the left margin, or even smack in the middle of the page) and any nearby text lines or other images flow smoothly alongside.

Right and left alignment of graphics is a recently introduced function, allowing you to create newspaper-like text that wraps around your images. For example, the Web page shown in Figure 12-5 was created using the following code:

```
<IMG ALIGN="right" SRC="dev1.gif" VSPACE=1 HSPACE=2>
<CENTER>
<H2>Aligning</H2>
</CENTER>
Notice how nicely this text wraps around the photograph. You can wrap
around text of any sort, size, or style. You can even wrap other
images. It's like magic! The text continues here. When we reach the
bottom of the photo, the text continues as usual. You can use right
and left image alignment to make your pages look like professional
newspaper or magazine articles. You would, of course, have to have
more interesting text (and a handsomer photo) in place of what's
here.<P>
```


Google/YouTube Exhibit 1005/1105
Page 00336

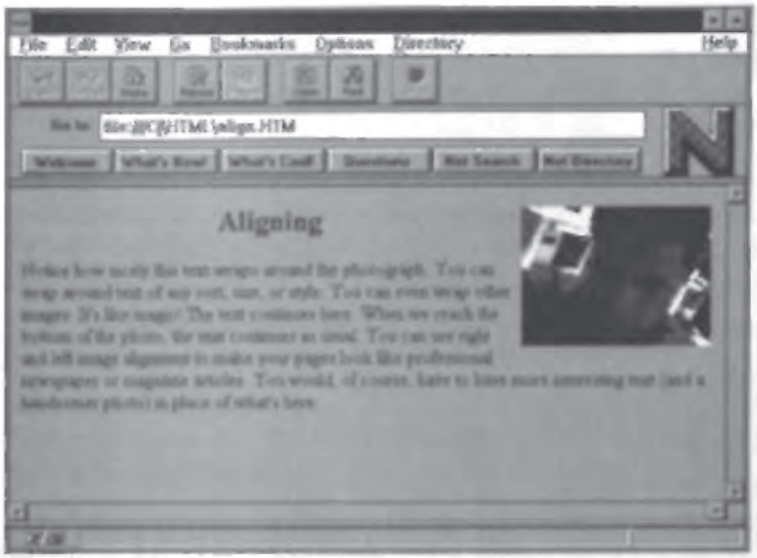


Figure 12-5 An image aligned to the right

Mino in the Gutter

You probably don't want your wrap-around text crowding up against a neighboring image. Mozilla allows you specify a *gutter*, forcing text to keep away from an image. You can specify the vertical space above and below an image by using the *VSPACE* attribute. The *HSPACE* attribute takes care of space to the left and right of the image.

For example, the command

```
<IMG SRC="image.gif" ALIGN="left" VSPACE=5 HSPACE=5>
```

pushes the surrounding text five spaces away.

Clearing Space After Line Breaks

Thanks to floating images, lots of text can now share the same space as a graphic. However, what if you want to place a certain piece of text next to an image (such as a caption), and place other text beneath the image?

The <BR> line break element won't do the trick alone. For example,

```
<IMG SRC="image.gif" ALIGN="left">This text here should be to the left
of the image.<BR>
But what about this text here? We want this to be below the image!<P>
```


Google/YouTube Exhibit 1005/1105
Page 00337

results in all the text being placed to the immediate right of the image (if the image is large enough).

Netscape makes use of the *CLEAR* attribute to solve this problem. You can designate

```
<BR CLEAR=left>
```

to break the line and continue writing text only when there are no more images against the left margin. For example, we can fix the above text by using the HTML sequence:

```
<IMG SRC="image.gif" ALIGN="left">This text here should be to the left
of the image.
<BR CLEAR=left>
But what about this text here? We want this to be below the image! And
now it is!<P>
```

You can also designate

```
<BR CLEAR=right>
```

for images that are aligned against the right margin. If you want to continue only after *both* the right and left margins are clear, use the command:

```
<BR CLEAR=both>
```

LESSON #3: TEXT ALTERNATIVES

Graphics are great for those who can see them, but what about the rest of the world? Due to slow modems, limited Internet connections, or sheer preference, thousands of people surf the Web using Lynx or other text-based browsers. Is it fair to just forget about these folks?

In most cases, text browsers simply skip images. Lynx displays a replacement tag instead, surrounded by brackets. The default is

```
[IMAGE]
```

Using the *ALT* attribute, you can suggest an alternate word or symbol to be shown. For example, if your HTML code was:

```
<IMG ALT=" [Hot Air Balloon] " SRC="/balloon.gif">
```

Lynx would display

```
[Hot Air Balloon]
```

STYLE TIP: Often, inlined graphics are just whimsical ways to spice up a document, but are otherwise optional. If an image is not important, it's a good idea to tell the text-based browser to ignore it. If you don't want any alternate text at all, just use a blank:

```
<IMG ALT=" " SRC="image.gif">
```


Google/YouTube Exhibit 1005/1105
Page 00338

Alternate text is especially important for hyperimages, such as icons or buttons. If all a user sees is the word [IMAGE], how is she supposed to know whether or not it's worth clicking on? For example, if you have a row of icons as in Figure 12-2:

```
<IMG ALT=" [Literary] " SRC="joyce.gif">
<IMG ALT=" [Art] " SRC="gizmo.gif">
<IMG ALT=" [Computers] " SRC="computer.gif">
<IMG ALT=" [VR] " SRC="dancer.gif">
<IMG ALT=" [Misc] " SRC="girl.gif">
```

A text-bound user would then see:

```
[Literary] [Art] [Computers] [VR] [Misc]
```

If you wish, you can use the ALT attribute alone, without specifying a source. Browsers of the future may even display icons for special key words, such as "Home," "Next," or "Previous."

LESSON #4: IMAGE SIZING

Images can take a long time to load, even with browsers like Netscape that support multiple image loads. This is because the browser must read through the image, determining its height and width, before it can lay out the rest of the document. The browser first draws a properly sized box and then it fills the image in, row by row.

What if the browser knew the exact size of a graphical element before it so much as loaded a single byte? The answer: Performance could be increased by up to 50 percent.

Netscape lets you solve this problem with the *HEIGHT* and *WIDTH* attributes. You specify the image's size in pixels, as follows:

```
<IMG SRC="image.gif" WIDTH=150 HEIGHT=200>
```

Netscape then uses this information to predraw the image box. That lets it load the rest of the Web page immediately.

NOTE: This speed-up technique only works for the Netscape browser. Of course, other browsers will just skip this sizing information, so including it could never hurt.

To determine an image's size, load it up using LView. Select Edit, Resize or press CTRL-R. The Resize dialog box appears, with the image's size—width times height—printed in the Current Size or Current Dimensions box.

Google/YouTube Exhibit 1005/1105
Page 00339

Figure 12-6 Using auto-scaling to show several sizes of the same image

Auto-Scaling

The sizing attributes can serve another function: scaling your images as they're loaded. This allows you to use several sizes of the same image. See Figure 12-6 for an example.

All you need to do is use WIDTH and HEIGHT specs that are different from the image's actual dimensions. The image can then be ballooned, stretched, or shrunken. If you want the image to appear with similar proportions, be sure to keep up the *aspect ratio* by multiplying the height and width by the same number. For example, if you have an image that is 25x30, you can load it in at four times its size by using the command:

```
<IMG SRC="smallimg.gif" WIDTH=100 HEIGHT=120>
```

Scaling is also useful for patterns, small photos, or other images that take up a lot of disk space. You can quickly load up a tiny image and have it fill the entire screen.

NOTE: Browsers other than Netscape will display a scaled image at its usual size.

Google/YouTube Exhibit 1005/1105
Page 00340

Percentage Auto-Scaling

If you don't feel like calculating out the exact size of a scaled image's height and width in pixels, you can have Netscape do the work for you. Simply specify the HEIGHT or WIDTH as a percentage of the available screen. This way, the image always appears relative to the rest of the Netscape page. If you make your Netscape window bigger, the image will grow bigger; if you compress your window, the image will shrink.

In general, you want to scale an image according to how wide it is. So, if you have a small 25 by 25 image, you can triple its height and have it fill half the screen by using the command:

```
<IMG SRC="smallimg.gif" WIDTH="50%" HEIGHT=75>
```

If you only type in one dimension attribute, the image will always keep the same aspect ratio. For example, use the command

```
<IMG SRC="fox.gif" WIDTH="75%">
```

The image's width fills up three-quarters of the screen, as shown in Figure 12-7. The height will change as the size of the screen changes, always corresponding to the dimensions of the original image.

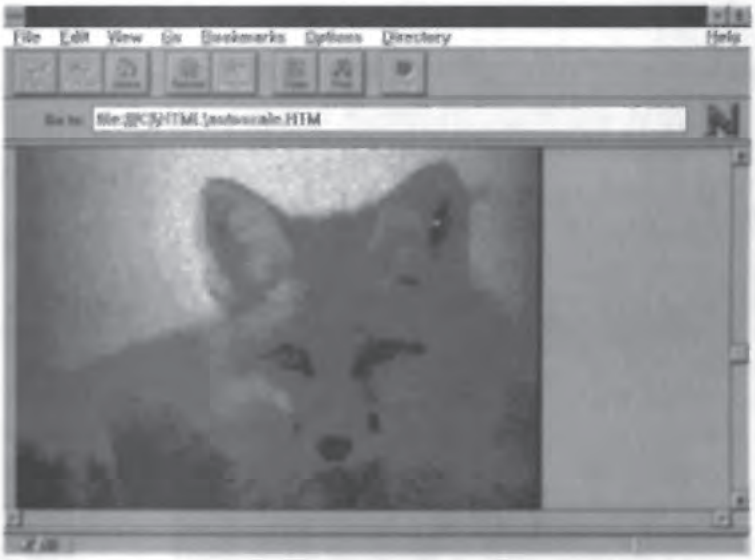


Figure 12-7 Using percentage auto-scaling to make small images leap off the screen

Google/YouTube Exhibit 1005/1105
Page 00341

LESSON #5: TRANSPARENT GIFS

Sneak a peek at Figure 12-8. Which fox looks better, the block-like, splotched-out one on the top or the floating, snazzy, cute little one on the bottom? (Can you tell the question is loaded?)

When dealing with logos, buttons, icons, or any other simple graphics, *transparent background images* are the only way to go. Most browsers support transparent images, automatically setting the image's background color to match the color of the browser's window—no matter what it is.

To make an image transparent, it must be in the latest GIF format—*GIF89a*. This format has the ability to mark a single color as the transparent one. Making an image transparent may take a bit of work, but the sleek results often speak for themselves.

STEP 1: Isolate the Background

The first thing you need to do to create a transparent GIF is to paint the image's background all one color. Most icons, buttons, and such are already set against a solid white or black background, in which case no isolating is necessary. If an image's background color isn't used anywhere else in the image, you can skip immediately to Step 3. However, if you'd like to make a photograph transparent (to design a floating face, for example), this process is much more difficult.

Figure 12-8 Which image is foxier?

Google/YouTube Exhibit 1005/1105
Page 00342

Professional tools such as Adobe Photoshop are built to create just such effects. However, you can often do a pretty good job of background isolation by using something as simple as the Windows Paintbrush program.

First, use WinGIF or IView to convert your image to the PCX or BMP format. To convert a file, you need to load it (using File, Open) and then save it under a different filename and file type (using File, Save As).

You can now load up the image using the Windows Paintbrush, which should be located in the Accessories group in your Program Manager. Select File, Open. The image appears on your screen. You now need to cut away the image's background, or any other parts of the picture that you wish to be transparent.

Select the Polygon Cut tool—the scissors cutting away the star. Remember kindergarten, where you had to cut out a doll outline from construction paper? Those skills are now put to the test. Click on the place where you'd like to start cutting and hold down the mouse button. Holding your hand steady and moving the mouse very slowly, you can now draw a line around the entire image. Pretend you're holding an X-acto knife, or directing a laser beam. Be sure to include the borders of the image as part of your outline. At the end, you should have a shape that corresponds to the part of the picture you wish to cut away. Figure 12-9, for example, shows the results of isolating a fox' face.

Figure 12-9 Cutting away an image's background

Google/YouTube Exhibit 1005/1105
Page 00343

To cut away what you've just selected, choose Edit, Cut or press CTRL-X. Your image should appear to be floating against a white background. If not, select Edit, Undo and try again.

If the image is bigger than your screen, you need to use the scroll bars to deal with the rest of the picture. Continue scrolling and cutting until the entire background is cleared. You can preview the entire image by selecting View, View Picture or pressing CTRL-P.

NOTE: Most every drawing package has similar tools, and most can handle more colors or graphics types than Windows Paintbrush. Experiment until you find the package most comfortable for you.

STEP 2: Selecting a Background Color

You're now ready to fill in a unique background color. Try to pick a basic, primary color that isn't used anywhere else in the image. Usually solid black or solid white works just fine. If not, select yellow, red, green, or blue.

STYLE TIP: If possible, use light grey as your background color. Not only is it the easiest color to work with, but if a browser doesn't support transparent images, grey usually blends in well with that browser's background anyway.

Once you've decided on a color, double-click on the color from the palette at the bottom of the screen. A dialog box appears showing you the color's Red, Green, and Blue *(RGB)* composition values. For example, black is comprised of 0,0,0. White is 255,255,255. Green is 0,255,0. Blue is 0,0,255. For simplicity, try to stick to colors that are based on combinations of 0 and 255. Write down or memorize this *RGB triplet* value and then click OK.

Click the Fill-in tool, shaped like a paint roller. You can now click anywhere along the white background. The background becomes your new color. Continuing with the fox example, we could fill in the background with Blue (whose RGB value is 0,0,255) since the color Blue appears nowhere else in the image.

STYLE TIP: Be sure the image's background color isn't used anywhere else in the image, otherwise your transparent image will seem to have holes in it. Of course, you may want this effect if you're using, for example, the image of a doughnut or a car with see-through windows.

When your image stands out against a solid background, select File, Save As to save it as a PCX or 256-color BMP bitmap. You now need to use WinGIF or IView to convert the file back to the GIF format. You should now have an image file named TRAN-FOX.GIF or something similar.

Google/YouTube Exhibit 1005/1105
Page 00344

STYLE TIP: When defining a background color, you may also want to draw a border around the image using that same color. You can do this easily using any drawing package's rectangle tool. This way, there will be a small transparent border around your entire image. Since most browsers draw a blue border around hyperimages, this invisible border will give your image the appearance of floating within a button.

STEP 3: Making the Background Transparent

Now comes the final step: Transforming your uniform-background GIF87a image into a transparent GIF89a image. If your computer can run LView Pro, then this step is easy. If not, you'll need to use a small DOS command called GIFTRANS.EXE.

LView Pro

Load up the image you just created by selecting File, Open. You now need to tell LView which color should be transparent. Select Options, Background Color. You see the Select Color Palette Entry window, shown in Figure 12-10.

You now need to try to find the background color from the palette. Depending on your image, this palette index may be 16 colors, 256, or more. If you isolated the background color using Window Paintbrush, then your background color is probably located in the top row. Click the background color. To be sure you've selected the right color, check the Mask Selection Using box. Every color *other* than the background color is shown in black (or white). A silhouette of your image should appear. Everything other than this silhouette now acts as a transparent background.

If you picked the right color, select OK. If not, guess again. Sometimes an image is made up of fifty shades of blue, and it can be difficult to find the right one. You can use the Current Selection Info RGB at the bottom of the dialog box to help you find the actual background color. Keep selecting possible colors until the RGB index matches the one you found in Step 2. Mask the image to be sure and then select OK.

NOTE: If you want your image to be interlaced, be sure to check Options, Save GIFs Interlaced. Lesson #9, later in this chapter, explains interlaced images.

You're now ready to save your new image. Select File, Save As. Be sure to Save File as Type GIF89a—*not* GIF87a. Once you save a GIF89a file, LView automatically designates the background color as the transparent

Google/YouTube Exhibit 1005/1105
Page 00345

Figure 12-10 Masking out the background color to be sure you have the right one

color. That's all there is to it. If you saved your file as FOX2.GIF, then, you could put the image on your Web page using the markup:

```
<IMG SRC="fox2.gif">
```

GIFTRANS

Another way of designating a transparent background color is a utility called GIFTRANS. GIFTRANS can be downloaded from the site listed in Appendix D. Unzip the file and copy GIFTRANS.EXE to any convenient directory along your path, such as the C:\WINDOWS directory.

GIFTRANS is a simple DOS filter that converts standard GIF87a images into GIF89a format. To exit to the DOS prompt, select the DOS icon from the Main group in the Windows Program Manager. Move to the directory where your TRANS.GIF file is stored. For example, if the file is in the C:\HTML\IMAGES directory, type

```
cd c:\html\images ENTER
```

at the DOS prompt.

You can now use GIFTRANS, which takes the form:

```
giftrans -t index source.GIF > destination.GIF
```


Google/YouTube Exhibit 1005/1105
Page 00346

Instead of *index* you'll need to type in the background color's RGB value in the hexadecimal form:

```
#rrggbb
```

Here's why we told you to only use RGB values of 0 or 255. The hexadecimal representation of 0 is *00*, the hexadecimal representation of 255 is *ff*. Your index, then, is comprised of three *00*s or *ff*s. Table 12-1 contains a short listing of sample color values:

Table 12-1 RGB index values for various colors

Color	RGB Value	Hexadecimal Index
White	255,255,255	#ffffff
Black	0,0,0	#000000
Red	255,0,0	#ff0000
Green	0,255,0	#00ff00
Blue	0,0,255	#0000ff
Yellow	255,255,0	#ffff00
Magenta	255,0,255	#ff00ff
Dark red	191,0,0	#bf0000
Dark yellow	191,191,0	#bfbf00
Dark green	0,191,0	#00bf00
Medium blue	0,191,191	#00bfbf
Light blue	0,255,255	#00ffff
Dark blue	0,0,191	#0000bf
Dark magenta	191,0,191	#bf00bf
Dark grey	128,128,128	#808080
Grey	192,192,192	#c0c0c0

NOTE: If you're familiar with hexadecimal notation, feel free to use any value at all as your background color.

For example, to convert our TRAN-FOX.GIF image into a transparent image, we'd type:

```
giftrans -t #0000ff tran-fox.gif > fox2.gif
```

We use the index #0000ff because the background color was Blue. The final product is now stored in the FOX2.GIF file. So, the next time we felt like sticking a transparent fox head in our Web page, we'd type:

```
<IMG SRC="fox2.gif">
```


Google/YouTube Exhibit 1005/1105
Page 00347

LESSON #6: CREATING SPACERS

Suppose you want to position two or more images next to each other. How can you determine the exact spacing? For example, suppose you want to create a pyramid of images, as in Figure 12-11. How does the Web browser know? The solution is to use transparent images to create, in essence, an invisible image. This image can then act as a *spacer*.

Spacers should only be one pixel high–like an invisible horizontal line. That way, they'll load in lightning fast while still serving their purpose. You can use LView to create a spacer:

1. Select File, New.
2. Type the image's width in pixels in the first box of the New Image Dialog, and type 1 for the image's height in the second box.
3. By default, the line should be drawn in the background index color. Select Options, Background Color to be sure.
4. Select File, Save and save your spacer. Type in a filename such as SPACE100.GIF. Be sure to specify the GIF89a format.
5. You can now use the spacer as you would any other image. For example, to create the pyramid shown in Figure 12-11:

```
<IMG SRC="space50.gif"><IMG SRC="gizmo.gif"><BR>
<IMG SRC="girl.gif"><IMG SRC="space50.gif"><IMG SRC="joyce.gif"><P>
```

Figure 12-11 A pyramid of images

Google/YouTube Exhibit 1005/1105
Page 00348

LESSON #7: PERKS, GIZMOS, AND FRILLS

Many browsers do a good job with things like <HR> horizontal lines, <UL> bullets, and other tiny graphics. But why not customize these yourself? Look at the difference between Figure 12-12 and Figure 12-13.

Bullets

The CD-ROM has a whole collection of bullets stored in the \CLIPMDIA\BULLETS directory. Normally, the HTML code for a bulleted list looks like:

```
<UL>
  <LI>One
  <LI>Two
  <LI>Three
</UL>
```

To insert your own bullets, you'd typically use code like the following sample instead of the standard list format:

```
<UL>
<IMG SRC="bullet.gif" ALT="*"> One<BR>
<IMG SRC="bullet.gif" ALT="*"> Two<BR>
<IMG SRC="bullet.gif" ALT="*"> Three<P>
</UL>
```

Figure 12-12 A Web page using standard built-in graphical elements

Google/YouTube Exhibit 1005/1105
Page 00349

Figure 12-13 A totally rad Web page with unique graphics of its very own

You can also get fancy with custom bullets, making your own nested definition lists, as in Figure 12-14. Here's the code for it:

```
Things you'll need to pursue a career in kickboxing:
<DL>
<DT><IMG SRC="BLUECUBE.GIF" ALT="*"> <B>Shin Pads</B>
<DT><IMG SRC="BLUECUBE.GIF" ALT="*"> <B>Arm Pads</B>
<DT><IMG SRC="BLUECUBE.GIF" ALT="*"> <B>A Uniform</B><DD>Various
styles of martial arts use different types of uniforms. Many modern
kickboxing studios are very liberal about this, accepting sweatpants
and tank top or any other sort of loose-fitting outfit. There are
three types of more traditional uniforms, however:
<DL>
<DT><IMG SRC="GRENBULL.GIF" ALT="+"> Cotton V-Neck
<DT><IMG SRC="GRENBULL.GIF" ALT="+"> Open-fronted Japanese Hogu
<DT><IMG SRC="GRENBULL.GIF" ALT="+"> Black Ninjitsu-Wrap Style
</DL>
<DT><IMG  SRC="BLUECUBE.GIF"  ALT="*"> <B>Men: A Cup</B>   <DD>Never
leave home without it!
<DT><IMG SRC="BLUECUBE.GIF" ALT="*"> <B>Women: A Sports Bra</B>
</DL>
```

Buttons / Icons

Buttons can take a plain Web page and make it feel like a software program of its very own. By giving the user a set of buttons to press, you make access

Google/YouTube Exhibit 1005/1105
Page 00350

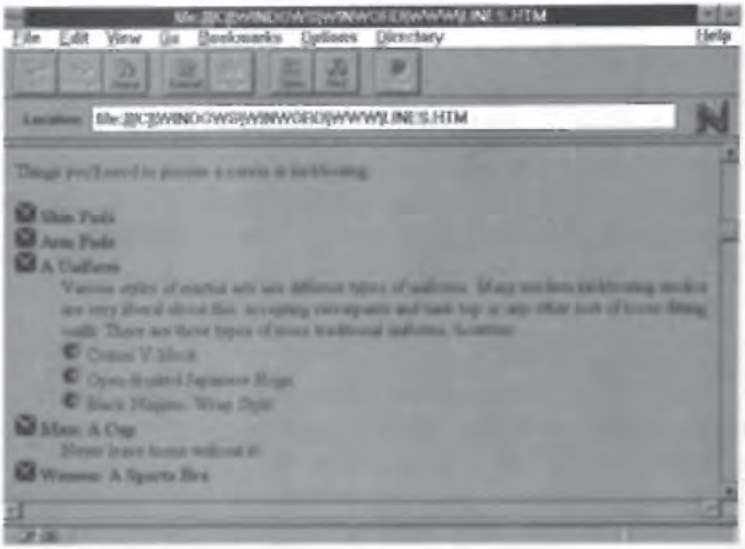


Figure 12-14 Shooting up the page with custom bullets

to your information more interesting and more customized: users can select the specific items they're interested in.

You'll find a large selection of standard buttons and icons in the CD-ROM \CLIPMDIA\BUTTONS directory.

You can arrange buttons any way your heart desires. However, there are two basic layouts:

HTML The toolbar: a row of buttons stretching across the screen. See Figure 12-3.

HTML The list: A vertical list of icons or buttons alongside some sort of description.

STYLE TIP: It's a good idea to include some sort of description next to each button, since the functions of certain icons may not be obvious.

Code for a toolbar of buttons would typically take the form:

```
<A HREF="home.html"><IMG SRC="home.gif" ALT="[HI"></A>
<IMG SRC="spacer.gif">
<A HREF="forward.html"><IMG SRC="next.gif" ALT="[>>]"></A>
<IMG SRC="spacer.gif">
<A HREF="back.html"><IMG SRC="previous.gif" ALT="[<<]"></A><P>
<PRE>[Home]     [ >> ]     [ << ]</PRE>
```


Google/YouTube Exhibit 1005/1105
Page 00351

Notice the small spacer (see Lesson #6) between each image. If you want your icons to appear right next to each other, remove the spacer. Also notice that the descriptive text along the bottom has to be carefully spaced out so it appears beneath each icon. Preformatted text is usually the best way to accomplish this.

Code for a list of buttons, on the other hand, would look something like:

```
Click one of the following buttons to move to another Web page:
<A HREF="home.html"><IMG SRC="home.gif" ALT="[H]">Home</A><P>
<A HREF="forward.html"><IMG SRC="next.gif" ALT="[>>]">Next</A><P>
<A HREF="back.html"><IMG SRC="previous.gif" ALT="[<<]">Previous</A><P>
```

STYLE TIP: Be sure to use the ALT attribute to name your buttons. For example, [H] can lead to a home page and [>>] can lead to the next page. This way, text-based users can "press" your buttons too.

Bars

The HTML <HR> command is a great way to separate sections, or to designate the end of a document. However, this line is always gray, straight, and monotonous. Why not use your own colors or line designs? The CD-ROM has a bunch of sample bars in the \CLIPMDIA\BARS directory.

To use a bar, just type

```
<CENTER>  <IMG SRC="bar.gif"><P>
</CENTER>
```

STYLE TIP: You can use Netscape's image sizing or alignment commands to center bars in the middle of the screen, offset them to the right or left, make them act as vertical graphical lines, or even separate columns of text.

LESSON #8: MAKING GRAPHICS HUMBLE

It's tempting to use oodles of graphics in your Web pages. Be wary, however, of overloading the Web with data. Graphic files are usually pretty large. If someone has a slow modem or Internet connection, the page may take almost hours to load. Although some users may use their browsers to turn off graphics, others may just get fed up and skip your page altogether.

There are several ways of making your graphics gentler, kinder, and faster to load:

- HTML Reduce the resolution of the image by shrinking it.
- HTML Use compressed graphic types like JPEGs, perhaps with GIF thumbnails.

Google/YouTube Exhibit 1005/1105
Page 00352

HTML Use Netscape's flip trick.

STYLE TIP: Many PC users have monitors that can display only 256 colors (as opposed to the millions and millions of colors available with an enhanced graphics card). For them, complex images will not appear as crisp or defined as you'd like. When posting images on the Web, try to use the clearest, most basic images you can and steer clear of images using tons of different colors. Even with high-powered monitors, there's a potential problem when you try to display two complex color images at once. If one image has 256 shades of blue and the other uses 256 shades of red, which colors will actually be displayed? Windows will usually try to share the colors, making both images seem splotchy and discolored. If you must put two complex images near each other, try to stick to a similar color scheme.

Shrink Me

You can easily use LView or WinGIF to change the size of any given image. Using either program, select File, Open and load up the image.

STYLE TIP: It's a good idea to change an image's size if it seems out of place, or unwieldy. You should also resize images to give your Web page some uniformity. If you have several buttons, for example, it looks much nicer to make them all the exact same size.

LView

If you're using LView, select Edit, Resize or press CTRL-R. The Resize dialog box appears. Type in either a new width or a new height. As long as the Preserve Aspect Ratio check box is checked, the image will keep its scale. If you'd like to stretch an image, you can deselect the Preserve Aspect Ratio checkbox and type in any values you wish. If you're going to be changing an image's aspect ratio, however, it's a good idea to crop the image first.

Suppose, for example, you want to take a full-sized rectangular image and make it into a square 50x50 button. First, you'd crop the image to make it square. Move your mouse to the top corner of the square, hold down the mouse button, and then create a box around the part of the image you want to keep. Release the mouse button. There should be a perforated box drawn around your image, as in Figure 12-15. Now select Edit, Crop! to throw the rest of the image away.

You can now resize the image. Select Edit, Resize or press CTRL-R. Deselect the Preserve Aspect Ratio checkbox and fill in 50s in the two boxes. The result is a postage-stamp-sized image that you can then save (File, Save As) as a GIF and use throughout your HTML documents.

Google/YouTube Exhibit 1005/1105
Page 00353

Figure 12-15 Cropping an image

WinGIF

Resizing an image works pretty much the same in WinGIF. Hold down the mouse to select a rectangular part of the image. As you create the cropping square in WinGIF, the square's dimensions are printed in the middle of the screen. This is a nice feature, allowing you to create perfectly scaled crops.

Once your cropping square looks the way you want it, let go of the mouse button. Select Edit, Trim to crop it. Select Edit, Resize or press F4. The Resize dialog box appears. Type in the image's new width and/or height. If you don't want to maintain the aspect ratio, deselect the Resize To Scale checkbox.

Thumbnails

GIF images are large and often take too long to load in. A better way to share photos or other detailed artwork would be to use the JPEG format. Unfortunately, most browsers cannot inline this format with the rest of the text. A solution to this problem is *thumbnails*, or tiny previews of larger or more intricate images. Using hyperimages, you can set up thumbnails to lead the way to actual full-sized images. Thumbnails take only a second or two to load. Figure 12-16 shows a few sample thumbnails.

Google/YouTube Exhibit 1005/1105
Page 00354

Figure 12-16 Twiddling through thumbnails

For example, if I had a high-resolution, thousand-color GIF image of New York City, it would take a while for any Web browser to display it. It would be much smarter to store the picture as a JPEG. Using LView or WinGIF, load up the picture of New York and resize it to 100x100 (or thereabouts). If you're using many thumbnails on one page, you may want to make them even smaller. Save this new image as a separate GIF.

To place a thumbnail in your HTML document, you'd typically use the form:

```
<A HREF="bigimage.jpg"> <IMG SRC="littleimage.gif" ALT="Big Image"
ALIGN=MIDDLE></A>
Click on this thumbnail image to retrieve the full high-resolution
JPEG.
```

Users could then click on littleimage.gif to load up the bigimage.jpg. If you want, you can emulate thumbnails using Netscape's sizing attribute, as follows:

```
<A HREF="bigimage.jpg"> <IMG SRC="bigimage.jpg" ALT="Big Image"
ALIGN=MIDDLE WIDTH=10%></A>
```

Click on this thumbnail image to retrieve the full high-resolution JPEG.
The image, however, will be full sized on other browsers.

Google/YouTube Exhibit 1005/1105
Page 00355

The Flip Trick

Netscape Mozilla has come up with an excellent function that allows you to quickly load in a low-resolution preview of an image, depending on the capabilities of your browser. This way, users with Netscape can see a low-resolution JPEG image as soon as your Web page loads. Eventually, this low-res JPEG will be replaced by a higher resolution GIF.

NOTE: For people who don't use Netscape, the high-res GIF is simply loaded as usual.

The *LOWSRC* attribute makes this possible. Create two copies of your high-resolution images, such as photographs or other full-screen images: one GIF and one JPEG. To call up the images, then, use the HTML command:

```
<IMG SRC="highres.gif" LOWSRC="lowres.jpg">
```

Netscape loads up small *lowres.jpg* first. After it gets the rest of the document loaded, it replaces the image with the *highres.gif* file. This acts kind of like an instant thumbnail. A user who likes the image will stay on your page and wait for the high-res image to fade in. Otherwise, she'll be off to another part of the Web.

NOTE: Remember, you can specify the *WIDTH* and *HEIGHT* attributes, in which case both low-res and high-res images will load in to the same scale. If you don't specify the image size, the high-res image will load in at the same scale as the low-res image regardless of what you meant it to be, since the rest of the document has already been laid out.

STYLE TIP: You can use this trick to perform some neat special effects, such as limited animation. Create the first frame as the LOWSRC and the second frame as the SRC. The second frame will appear to fade in from the first frame. This effect turns out even nicer if the GIF is interlaced (see the next section).

For example, if you have a 500,000-byte 800x600 GIF of New York City and a tiny 5,000-byte 50x50 JPEG copy, you can quickly load up a blurry, scaled-up version of the JPEG and then gradually replace it with the actual image:

```
<IMG SRC="nyc.gif" LOWSRC="nyc.jpg" WIDTH=800 HEIGHT=600>
```


Google/YouTube Exhibit 1005/1105
Page 00356

LESSON #9: INTERLACED GIF IMAGES

Different browsers load up graphics in different ways. Mosaic, for example, loads one image at a time, only displaying the image once the entire thing is loaded. Netscape and similar browsers are much more sophisticated, featuring *incremental* graphic loading. Netscape reads in all images at once, displaying each image as it loads. You can watch an image take shape like a curtain dropping, starting from the top, line by line.

The GIF format can store image data in an *interlaced* style. All this technobabble means is that a graphic appears to fade in as it loads. Instead of a curtain dropping, an interlaced GIF looks more like a set of venetian blinds opening. The latest version of Netscape goes even further, offering the *progressive enhancement* effect. As an interlaced GIF loads, Netscape fills in the blanks with an educated guess of what the color should be. This gives the impression of a blurry image, constantly being sharpened.

Interlaced GIFs do not load faster or more accurately than noninterlaced GIFs, but many people prefer them because they make more sense sooner. For example, it takes a quarter of the time for a large image to appear—in primitive form—on the screen. Often, this is enough for the user to decide whether it's worth waiting for the rest of the image.

The WinGIF program makes it easy to create interlaced images. Simply follow these steps:

1. Load up the GIF (or other image format) you want to interlace.
2. Select File, Save.
3. Click the Format>> button.
4. Select the GIF radio button.
5. Be sure the Interlace GIF box is checked, as in Figure 12-17.
6. Make sure the proper filename is filled in (the GIF extension is automatically added).
7. Click Save.
8. You may be asked if you'd like to overwrite the existing file. Click OK.

Google/YouTube Exhibit 1005/1105
Page 00357

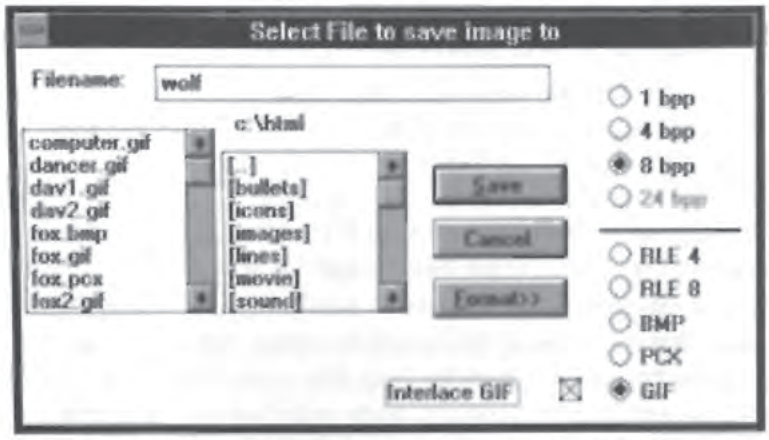


Figure 12-17 Saving a GIF in the interlaced style to create extra-snazzy Web images

You can also interlace images with LView Pro. Just be sure the Options, Save GIFs interlaced is checked before you save any GIF.

LESSON #10: CLICKABLE IMAGEMAPS

So far we've seen graphics do some pretty amazing things, acting as text aids, design elements, and even buttons. But how about creating an entire custom Web page consisting of one huge graphic. How about having that graphic be smart enough to know what part of it is being clicked, and smart enough to perform several distinct actions.

An example: You load up a Web page and see the front view of what seems to be a shopping mall. You click on the doorway of the Sporting Goods store and can now shop for baseballs or tennis shoes. If you clicked on the Software store you could've browsed the latest CD-ROMs.

Another example: You see a floor plan of a museum. You click on Renaissance and images by Leonardo DaVinci and Raphael suddenly appear. You click on Roman Sculpture and a huge close-up of Zeus fills your screen.

Yet another: The graphic on your screen seems to be of several photographs, each of a different person, all tossed together on a table. You click on Mary's face and are transported to her home page. Bob's face takes you to his page. Likewise with Ralph.

One last one: A map of the world fills your screen. You click on a red dot near London and are transported to your company's British offices.

Google/YouTube Exhibit 1005/1105
Page 00358

Click on Denver and you visit the Mile High City. Click on the Mexico City dot and the next thing you know pages full of Spanish fill your screen.

As you can see, these *imagemaps* allow your creativity to jump off the deep end. Although the above scenarios could've been accomplished using several different hyperimages, the interface just wouldn't have been as intuitive or as fun.

Imagemaps allow you to make a hyperimage interactive. It's much like a patchwork of many little hyperimages, all strung together. Depending on where on the graphic you click, you'll be linked to different URLs. Given a graphic, you can use imagemaps to designate polygons, circles, or rectangles as their own anchors.

NOTE: If you want to use imagemaps, you must have a true httpd Web server. Since some of your imagemap information must be stored in the Web server's private directories, ask your system administrator if he or she will help you set up an imagemap before you begin.

STYLE TIP: Imagemaps won't work with a non-graphical browser such as lynx. Always be sure to offer an alternate menu of hyperlinks below your imagemap. See Step 5, later in this chapter.

STEP 1: Create the Image

There is no one way to create imagemaps. You may want to use one big image, such as a floor plan or a map, or you may want to combine several smaller images together. Although you can use a simple tool like the Windows Paintbrush, you can design much nicer and professional images using a commercial package such as Adobe Photoshop or Illustrator.

One common method of imagemap design is to create several distinct buttons or images. You can then load these all onto one canvas and scatter them around, combine them, draw arrows, or create other logical connections. Browse through Chapter 8 for some more sample imagemap ideas.

Another idea is to take a photograph of any typical object associated with your subject, such as a filing cabinet, an office desk, or even an entire building. You can then scan in the image and create hot spots for various drawers, doors, windows, appliances, etc.

In any case, you generally want to label each hot spot clearly. Write some descriptive text beneath, atop, or embedded in each hot spot. You can use LView to add pretty clear horizontal or vertical text to most images. If you have a tool like Adobe Photoshop or Adobe Illustrator, you can even create *anti-aliased* text that blends right into the image, making the print seem flawless.

Google/YouTube Exhibit 1005/1105
Page 00359

STYLE TIP: Design your map so the boundaries of clickable items are pretty obvious. Space your hot spots to make it easy for the user to tell where to click. Try not to overlap too many hot spots.

STEP 2: Create the Map

The key to imagemaps is the map file itself. When you click an imagemap, your browser sends the coordinates of that click to a special imagemap program. This software then performs the following steps:

1. It notes exactly where on the image the mouse was clicked.
2. It looks up these *hot-spot* coordinates in a map file.
3. It jumps to the URL specified in the map file.

Maps files always have the *.map* extension. They can be stored anywhere, but it's usually most convenient to just place them in the same directory with your HTML code and the image itself. The map file looks something like this:

```
default http://www.smartpants.edu/righthere.html
#The Home Page
poly (10,75) (29,65) (21,57) (29,42) (36,43) (45,35) (65,41) (56,21)
(69,7) (87,11) (105,11) (121,16) (137,26) (159,21) (174,66) (167,98)
(163,110) (157,110) (144,135) (122,142) (124,152) (112,152) (111,157)
(106,151) (77,151) (73,148) (65,148) (52,137) (47,140) (43,138)
(31,143) (6,79) http://www.smartpants.edu/home.html
#The Computers Page
rect (500,95) (588,234) http://www.smartpants.edu/computers.html
#The Graphics Page
circle (223,85) 30 http://www.smartpants.edu/graphics.edu
```

For NCSA servers, the map file looks more like:

```
default http://www.smartpants.edu/righthere.html
#The Home Page
poly http://www.smartpants.edu/home.html 24,125 17,124 17,270 113,272
115,172 205,172 203,126 150,124 144,136 122,144 125,157 108,153 69,152
59,144 57,137 42,137 35,139 26,126
#The Computers Page
rect http://www.smartpants.edu/computers.html 500,95 588,234
#The Graphics Page
circle http://www.smartpants.edu/graphics.edu 67,50 48,30
```

In either case, the file contains:

HTML Comments (prefaced by the # character)

HTML A default URL for clicks that don't fall into any specified range

Google/YouTube Exhibit 1005/1105
Page 00360

HTML Hot-spot circles with affiliated URLs

HTML Hot-spot rectangles with affiliated URLs

HTML Hot-spot polygons with affiliated URLs

How in the world, you might wonder, do you know where to specify the coordinates of a given image's hot spots? Luckily, there's a Windows program that does it all for you: MapEdit.

MapEdit 1.1.2

To create an accurate imagemap, you'd need to be able to point to the image itself and know exactly which coordinate is which. Luckily, Thomas Boutell created the MapEdit program, which does just that, and more. All in all, MapEdit makes imagemaps a snap.

You'll find MapEdit in the \WINDOWS\TOOLS\MAPEDIT directory of the enclosed CD-ROM. Copy it to any directory on your hard disk, then set up an icon for MapEdit in the Windows Program Manager.

NOTE: MapEdit is not in the public domain. Be sure to register for MapEdit if you do end up using it. Commercial users need to pay $25, but nonprofit and education users can just send the author a postcard. Read the licensing terms included with the software for more information.

MapEdit loads up your GIF image and places it in a neat window. You can then use the mouse to draw rectangles, circles, or intricate polygons around certain parts of the image. You can then specify a URL for each of these hot spots.

Loading an Image

Once you've created a suitable image for your imagemap, you can load it into MapEdit by selecting File, Open/Create. The Open/Create Map dialog box appears, as in Figure 12-18. Type the GIF filename in the box or click Browse to search your hard disk for the proper file.

If you're creating a new map, just type any name (with the *.map* extension). In general, it's easier to keep track of things if you use the same name for your map as you do for your image.

NOTE: You can also edit an existing imagemap by designating its name in the Map Filename box.

Google/YouTube Exhibit 1005/1105
Page 00361

Figure 12-18 Designate the name of the GIF image and the map file

By default, MapEdit creates an NCSA map file. If your system is using the CERN Web server, then click the CERN radio button.

When you're done, click OK. Click OK again when the system asks if you'd like to create the given map file. It takes quite a while for MapEdit to load your GIF into memory. Eventually, it appears on the screen. If your image is larger than the screen, you may need to adjust the window or use the scrollbars.

The Default URL

If somebody clicks a *cold* area of your imagemap (one that isn't covered by a hot spot), what happens? If you don't designate a default URL, then most browsers display a message telling the user that the area they clicked has no link.

It's a good idea to set up a default URL instead, which can take the user to the most likely or popular page. Select File, Edit Default URL. The Default URL dialog box appears. Fill in the default and click OK.

Drawing Hot Spots

Creating polygonal, circular, or rectangular hot spots is simple. For example, select Tools, Rectangle to designate a rectangular region. Click the left mouse button on one of the rectangle's corners and then move the mouse to the rectangle's opposite corner. If you want to abort this rectangle, click the left mouse button again. Otherwise click the *right* mouse button.

The Object URL dialog box appears. Fill in this hot spot's URL (the way you would designate HREF in an anchor) in the URL for clicks on the object box, as shown in Figure 12-19. You can also type in a description of the URL or other comments in the Comments box. When done, click OK.

Google/YouTube Exhibit 1005/1105
Page 00362

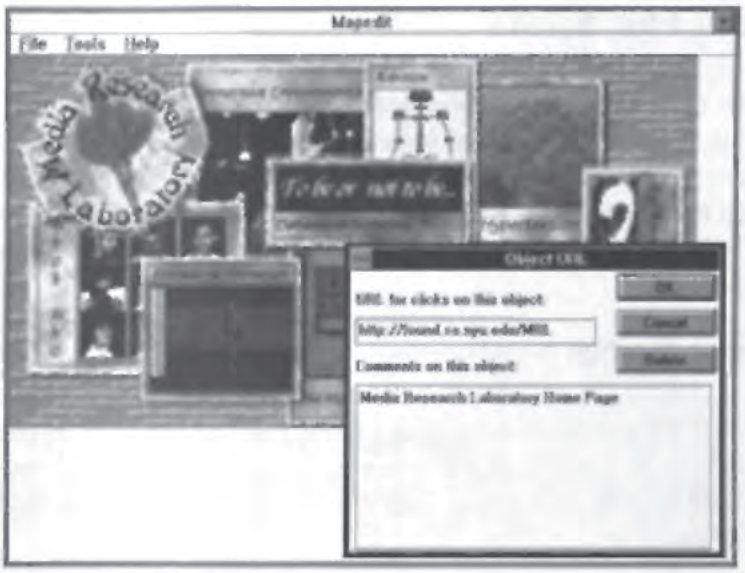


Figure 12-19 Making a spot hot

NOTE: If you don't know the URL yet, just type in some sort of guess. Later, you can use MapEdit to change the URL. See the Editing Hot Spots section of this chapter.

Creating a circle works the same. Just select Tools, Circle. Move the mouse cursor to where you want the center of your circle to be and click the left mouse button. Now move the mouse, lengthening the circle's radius. When you have the circle you want, press the *right* mouse button. Fill in the Object URL dialog box, as before.

Finally, you can create polygons with as many sides as you wish. This allows you to specify very specific shapes and patterns. Click on where you want the polygon to begin. You can now draw little lines connected to each other until your polygon is formed. Just keep clicking at strategic spots around the area you wish to define: a blob, a starfish, whatever. Continue until you've outlined the entire area. To cancel the polygon, press the left mouse button again. If you like the way it looks, click the *right* mouse button and fill in the Object URL dialog box.

NOTE: If hot spots overlap, the first one you created will control where the user goes.

Google/YouTube Exhibit 1005/1105
Page 00363

Testing the Hot Spots

MapEdit allows you to try before you buy. Just select Tools, Test+Edit. Now click anywhere on the image, the way a user at your Web page would. Anytime you click on a hot rectangle, circle, or polygon, the URL window pops up, showing you the link you've just defined. The current hot spot is also highlighted in green.

Click OK to continue testing. MapEdit remains in testing mode until you choose a new Tool.

Editing Hot Spots

If you test a hot spot and it doesn't work the way you wanted it to, you can edit it on-the-fly. All you have to do is type the new information in the Object URL dialog box. You can delete it just as easily. Just click the Delete button in the Object URL dialog box, and the hot spot and all associated information is tossed out.

Saving the Map File

When your imagemap works just the way you want it to, you can save the map file. Select File, Save. The map file you first specified is created or updated.

That's all there is to it. Keep track of where your map file is located; you'll need it for the next steps.

NOTE: You can use the File, Save As command to save your map to a different file, or in a different format.

STEP 3: Write the HTML

To use an imagemap, just create a suitable GIF image and display it as usual using the IMG element. However, you need to add the *ISMAP* attribute letting the browser know that the image is a clickable map. Once ISMAP is declared, your browser automatically passes the coordinates you clicked on to the imagemap program.

For example, if your image is called *entrance.gif* and your map file is *entrance.map*, you would type:

```
<A HREF="http://www.smartpants.edu/cgi-bin/imagemap/entrance">
<IMG SRC="entrance.gif" ISMAP></A>
```

The anchor, then, should point to the imagemap script itself. If you do not know where your imagemap program is located, ask your system

Google/YouTube Exhibit 1005/1105
Page 00364

administrator. The entrance "subdirectory" designates the name of your imagemap program, as designated in the *imagemap.conf* file. See the next section for instructions on setting up your imagemap configuration file.

NOTE: With some systems, instead of designating just the name of your map, you need to include the map's exact location.

STYLE TIP: In general, imagemaps should be centered, without borders. As such, a typical imagemap command would look like:

```
<CENTER><A HREF="http://found.cs.nyu.edu/cgi-
bin/imagemap/homepage"><IMG BORDER = 0
SRC="http://found.cs.nyu.edu/MRL/Homepage2.gif" ISMAP></A><P>
</CENTER>
```

STEP 4: Set Up the Server

Most of the action with imagemaps takes place in the background. Your http server (usually on a Unix system) performs the tasks using cgi-bin programs. See Chapter 15 for details about CGI and Chapter 21 for information about Web servers. However, you don't really need to understand how servers work—you just need to have access to one.

NOTE: If you don't have access to your own Web server, ask your system administrator to perform the following steps for you. Most commercial Internet providers should be happy to set up their imagemap software to include your personal map.

To use imagemaps, you need the *imagemap cgi* program. If this program isn't already compiled and installed in your cgi-bin directory, be sure to do so. Both CERN and NCSA http servers have versions of this program, usually included with their server. When your Web browser comes across an imagemap (designated by the ISMAP attribute) it automatically runs the imagemap software.

NOTE: If you're using a server other than CERN or NCSA http, check with the server's distributor for details. In most cases, the steps for creating an imagemap are exactly the same.

If you have access to your NCSA http directories, then you can easily set up your imagemap. Simply switch to your server's *conf* subdirectory. In other words, if your server is located at

```
/www/httpd
```

change to the *conf* directory by typing

```
cd /www/http/conf
```


Google/YouTube Exhibit 1005/1105
Page 00365

There should be a file here called *imagemap.conf*. If not, you may need to create it. You now need to add a line to this file telling the server where your imagemap's map file is located. This takes the form

```
mapname: map's full directory
```

For example, if the map's name is *entrance.map* and it's located in the */www/http/htdocs* directory, you would add the line

```
entrance: /www/http/htdocs/entrance.map
```

That's all there is to it, on the server side.

STEP 5: The Text-Based Index

Whenever you use an imagemap, be sure to include a text-based index for those folks who can't—or don't have the time to—display graphics. For example, suppose your imagemap contains a bunch of flags each leading to one of the following areas:

HTML English

HTML Japanese

HTML French

HTML Russian

HTML Hebrew

HTML Arabic

HTML Greek

In this case, you should include a few lines of HTML (or a hyperlink to a separate HTML document) as follows:

```
Here's the text-based index:<P>
<UL>
<LI><A HREF="english.html">English</A>
<LI><A HREF="japanese.html">Japanese</A>
<LI><A HREF="french.html">French</A>
<LI><A HREF="russian.html">Russian</A>
<LI><A HREF="hebrew.html">Hebrew</A>
<LI><A HREF="arabic.html">Arabic</A>
<LI><A HREF="greek.html">Greek</A>
</UL>
```


Google/YouTube Exhibit 1005/1105
Page 00366

LESSON #11: WHAT'S YOUR BACKGROUND?

Hold onto your hat. One of the neatest features in HTML3 has just been announced: the BACKGROUND attribute. This allows you to take a small picture, pattern, or color, and use it as a texture for the entire Web page's background. This way, your Web page can appear to be printed on sandpaper, stucco, metal, rock, water, or against any type of wallpaper you can imagine. See Figure 12-20 for an example.

NOTE: Currently, only Netscape supports the BACKGROUND attribute, though it is within the current HTML3 specification.

The attribute itself is very simple. Just specify the GIF image file for your background in your Web page's <BODY> element:

```
<BODY BACKGROUND="pattern.gif">
```

NOTE: Your background images should be as small as possible, since nothing can be loaded until the background has been painted. Small (less than 150x150) backgrounds can be loaded and rendered quite quickly, however.

STYLE TIP: Try not to mix transparent images and background patterns, since your custom background will not show through the transparent spots.

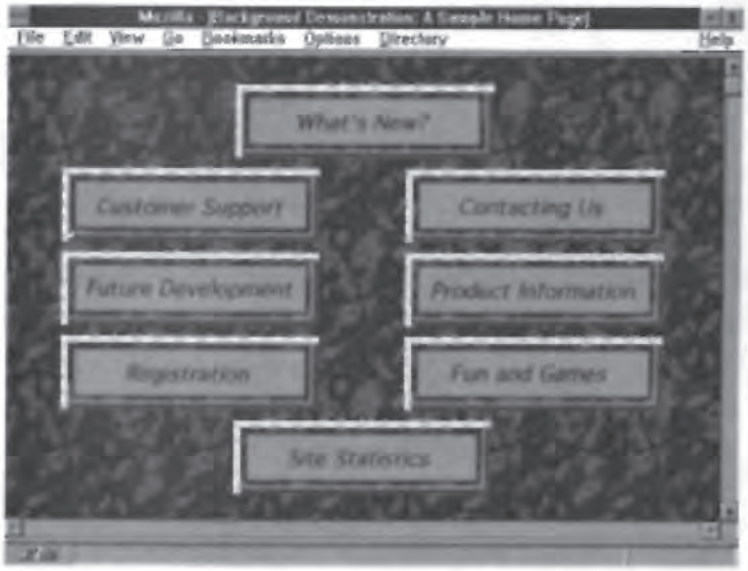


Figure 12-20 A custom Web page background

Google/YouTube Exhibit 1005/1105
Page 00367

Background Color

Netscape has come up with a quick and easy attribute which allows you to change the color of your background. This way, you can use the same pattern for a rainbow of different pages.

The BGCOLOR attribute is also part of the <BODY> element. Simply specify the color as a hexidecimal RGB color triplet, which always begins with the pound sign (#). For example, the color black is #000000 and white is #FFFFFF. See Table 12-1 earlier in this chapter for a listing of several sample colors. For example, to make the background black, you would use the markup:

```
<BODY BACKGROUND="image.gif" BGCOLOR="#000000">
```

Foreground Color

Usually, the text on your Web page is printed in black. However, if your background is dark, black text may very well be illegible. Wisely, Netscape allows you to specify the foreground color.

Text Color

To change the color of the document's standard text, use the TEXT attribute along with an RGB triplet. For example, if you want light-gray text, use the markup:

```
<BODY TEXT="#F0F0F0">
```

Hyperlink Color

Normally, a hyperlink is blue, a visited link is purple, and active links are red. Netscape also allows you to specify the color for each hyperlink, visited link (a link which was already loaded), and active link (a link currently being loaded). Simply use the LINK, VLINK, and ALINK attributes, respectively. These attributes are also expressed within the <BODY> element. For example, if you want yellow links, use the markup:

```
<BODY LINK="#FFFF00">
```

A typical <BODY> element, then, might look something like:

```
<BODY BACKGROUND="pattern.gif" BGCOLOR="#FFFFFF" LINK="#FFFF00"
VLINK="#000000" ALINK="#00FFFF">
```


Google/YouTube Exhibit 1005/1105
Page 00368

Getting Backgrounds

The Netscape Communications Corporation has many sample backgrounds at:

```
http://home.netscape.com/home/bg/index.html
```

You can point to these images directly or copy them and use them as a base for your own backgrounds.

Many groovy backgrounds are also stored on the enclosed CD-ROM in the \CLIPMDIA\BKGRNDS directory.

LESSON #12: EMBEDDING

The latest version of Windows Netscape allows you to use the new <EMBED> element to drop most any kind of document directly in your Web page. As long as a user has a viewer for the document on his or her computer, an embedded file appears as some sort of icon. A user need only click on the icon to see or access the document.

You can use EMBED to insert word-processed documents, movies, sounds, or nearly any type of graphic. For example:

```
<EMBED SRC="bird.tif">
```

You can also use the WIDTH and HEIGHT attributes to fit the embedded object within a certain space. Simply specify the dimensions in pixels. For example, to create an image which appears in a 100X50 box:

```
<EMBED SRC="bird.tif" WIDTH=100 HEIGHT=50>
```

THE SILVER SCREEN

The Web can handle any sort of file—even movies. On a computer, "movies" can be animations, multimedia presentations, film clips, sequences of images, or even just soundtracks. Two main movie/multimedia formats are popular on the Web:

HTML QuickTime .MOV files

HTML Multimedia .MPEG files

Additional types of movies and animations are listed in Table 2-1 in Chapter 2. Most Web browsers don't have built-in movie-display software,

Google/YouTube Exhibit 1005/1105
Page 00369

but you can find movie viewers for most every format, operating system, or computer type. Viewers are discussed further on in this section. In most cases, you should be able to download a supported viewer at the site where you retrieve the movie or animation.

WARNING: Video runs very, very slowly on machines without at least 8 MB of memory. And if your machine doesn't have a Super-VGA monitor or better, the video will probably look pretty splotchy.

Lights, Camera... Action?

To add a movie to your Web page you need to set up an anchor that leads to the movie file. For instance, to play the movie *rocket.mov*, just include the line:

```
<A HREF="rocket.mov">Click here to see the Rocket movie. (2.2M)</A>
```

The rocket.mov file must be stored in the same directory as the rest of your HTML documents.

It's a good idea to use a preview frame from your movie instead of a simple text anchor like the one above. Most video capture tools—such as Adobe Premiere and MovieMaker—allow you to do this. This makes a "poster" for the movie, to help the user decide if it's worthwhile to view or download the movie itself. For example, if you captured a frame of the rocket movie as *rocket.gif*, you would type:

```
<A HREF="rocket.mov"><IMG SRC="rocket.gif"></A>Click here to see the
Rocket movie. (2.2M)
```

STYLE TIP: Since movies are terribly long, be sure to adequately describe each movie. Otherwise, people may waste many minutes downloading movies they don't really care to see. It's also somewhat of a convention to tag how long each movie is, in megabytes. This way, someone with a slow modem will know not to bother checking out your enormous 10-megabyte film.

You can, of course, also include movies that aren't stored in your local HTML directory. You can use movies located anywhere else on your system, or even anywhere else on the Internet. For example, if you come across a movie at the *http://www.movieland.com/rocket.mov* URL, then your link would look like:

```
<A HREF="http://www.movieland.com/rocket.mov">Click here to see the
Rocket movie. (2.2M)</A>
```


Google/YouTube Exhibit 1005/1105
Page 00370

You can browse around the Internet, finding movies that fit the theme of your Web page. When you come across something interesting, you can check the link's complete URL at the bottom of the screen in Mosaic or Netscape simply by touching the link with your mouse arrow. In Lynx, just highlight the link and press [-].

NOTE: A distant movie generally takes longer to load and play, due to the time it takes to leap across the network.

QuickTime

QuickTime is the Macintosh video and sound capture tool. Movie files generally end with the .MOV extension. There are also several Windows quicktime viewers, one of which is found at the site listed in Appendix D. Unzip and copy the files to a convenient directory on your hard disk. You can now configure Mosaic or Netscape to call this viewer whenever they have to present a QuickTime movie.

NOTE: If you grab a QuickTime movie from a Macintosh system, you may have to *flatten* it before you can play it on another operating system. Mac users should flatten movies before putting them on the Web. When you flatten a movie, all the resources, image data, and QuickTime sound are placed into one file. Try to search the Web for the shareware *flattmoov* file. You can also flatten a flick using the commercial Adobe Premiere software.

MPEGPLAY

MPEG stands for the Moving Pictures Experts Group, a body of people who work with digital video compression, storage, and viewing. The MPEG format is perhaps the most popular format of computerized video and multimedia.

There's a very good shareware MPEG player available (called, appropriately, MPEGPLAY) in the CD-ROM's \WINDOWS\VIEWERS\MPEGPLAY directory. Copy the entire directory to your hard disk and configure your Web browser to use MPEGPLAY as your MPEG viewer. See Chapters 4, 5, and 6 for instructions on setting up viewers.

The next time you download a movie from the Web, MPEGPLAY pops up. You can use the VCR-type controls to Rewind, Stop, Advance the movie one frame, or Play the entire movie. These same controls can be selected from the Movie menu. Figure 12-21 shows a sample movie being played.

Google/YouTube Exhibit 1005/1105
Page 00371

Figure 12-21 Watching a Web flick

GRABBING GRAPHICS

Where do images come from? Spend just a few minutes browsing the Web and you'll realize that there's no lack of graphics in cyberspace. There are three basic ways of building up your collection of images: you can create them, capture them, or download them.

WARNING: Be wary of copyrighted images. In most cases, you'll need to give the artist credit before you can display a copyrighted image—and sometimes it'll cost money as well.

Playing Picasso

If you're a bit of an artist you can, of course, design your own images. Tools for this range from the simple Windows Paintbrush to the painterly Adobe Illustrator or Fractal Painter. There are also plenty of computer design firms who can create just the logo, backdrop, or style you've been picturing.

If you're interested in creating video, you'll need more sophisticated resources. Not just a video camera, either; your computer needs a video digitizer card and some sort of video capture software such as Adobe Premiere.

Google/YouTube Exhibit 1005/1105
Page 00372

WARNING: Movie files are huge. Try to edit your movies to be as short as possible while still preserving their quality and impact. You may want to save your movies in black-and-white format, since this often takes up less space.

There are many software packages capable of producing animations or multimedia presentations. Check out Autodesk Animator or Macromedia Director if you're interested in creating professional quality cartoons or other animations.

Capturing

If you come across an image you'd like to use, you can easily capture it. In general, there are three capture methods. Depending on the image you want and the browser you're using, one of these methods should work like a charm.

Load to Disk

Most browsers have a Load to Local Disk option. Turn this option on and then reload the page. As the image loads up, the browser asks for a filename. In this way, you can save any inlined GIF to your hard disk. When you're done capturing, be sure to turn the Local Disk option off.

Loading the Image's URL

If your browser doesn't offer the Load to Local Disk command, there's a slightly more troublesome method: Display the current page's HTML source. Browse through the markup until you find the <IMG> command that corresponds to the image you want to snag. Copy the SRC's URL to your clipboard or write it down. If the SRC is local, you can just append this filename to the current Web page's URL. You can then instruct your browser to load up this specific URL. Select File, Save As and you should be able to save the image onto your hard disk.

For example, suppose you're browsing a page at:

```
http://graphics.smartpants.edu/webstuff/cool.html
```

and you come across a great picture of a pickle. You view the HTML source and find the line:

```
<IMG SRC="pickle.gif">
```

You can download the pickle by opening the URL:

```
http://graphics.smartpants.edu/webstuff/pickle.gif
```


Google/YouTube Exhibit 1005/1105
Page 00373

Full Screen Capture

If you like, you can grab an entire screen itself. Windows has a built-in screen capture utility that allows you store the current screen in the clipboard: Just press ALT-PRT SC.

You can now load up the Windows Paintbrush, or any other paint program, and paste this screen image onto the canvas. You can now work with the image, erasing the parts you don't want, changing colors, or touching things up.

Image Hunting

The enclosed CD-ROM contains a library of images to get you started. General images are in the \CLIPMDIA\IMAGES directory. There's also a directory for bullets, icons, and bars.

If you don't find what you need, you can search cyberspace for a suitable image. The only problem with downloading images from the Internet is that there are too many places to look. There are literally hundreds of Web pages, FTP sites, or other archives full of GIFs, JPEGs, movies, animations, and more.

Table 12-2 lists some of the best places to get general images. Table 12-3 lists archives full of icons, bullets, buttons, bars, and other Web spices. And, finally, Table 12-4 contains a full list of animations, videos, and other movies. Happy hunting!

Table 12-2 Just a few places to find images over the Web

Archive Name	URL
Internet Image Finder	http://www.cm.cf.ac.uk/Misc/wustl.html
Yahoo Server Index	http://akebono.stanford.edu/yahoo/Computers/Multimedia/Pictures/
WWW Power Index	http://www.webcom.com/power/multimedia.html
Archive Image Archive	ftp://archie.au/graphics/gifs
Flinux Image Archive	ftp://flinux.tu-graz.ac.at/pub/graphics
Funet Image Archive	ftp://ftp.funet.fi/pub/pics
Informatik Image Archive	ftp://ftp.informatik.tu-muenchen.de/pub/rec/images
Isca Image Archive	ftp://grind.isca.uiowa.edu/images
Nic Image Archive	ftp://nic.vokom.sk/pub/multimedia
Smithsonian Archive	gopher://bramble.er.usgs.gov/1ftp%3aphoto1.si.edu%40/
Stuttgart Image Archive	ftp://ftp.uni-stuttgart.de/pub/graphics/pictures
Sunet Image Archive	ftp://ftp.sunet.se/pub/pictures
Sunsite Image Archive	file://sunsite.unc.edu/pub/multimedia/pictures/

Archive Name	URL
UIUC Image Archive	http://www.ncsa.uiuc.edu:80/rml/Gifs/
WUArchive Images	ftp://wuarchive.wustl.edu/
Newsgroup Archives	http://web.cnam.fr/Images/Usenet/ebpen/summaries/
Space Image Archive	http://www.jsc.nasa.gov/~mccoy/Images/"> Daniel's Space Image Archive
Another Space Archive	ftp://tezuka.rest.ri.cmu.edu/usr/anon/pub/space
More Space Images	http://info.cern.ch/Space/Overview.html
Astronomy Pictures	http://www.univ-rennes1.fr/ASTRO/astro.english.html
NASA Space Images	ftp://explorer.arc.nasa.gov/pub/SPACE/GIF
Data Visualization	http://www.nhgs.tec.va.us/GovSch/datavisual.html
MRI Images	http://www-mri.uta.edu/"> Magnetic Resonance Imaging
Historical NASA Archive	http://www.ksc.nasa.gov/history/history.html
Ansel Adams Photos	http://bookweb.cwis.uci.edu:8042/AdamsHome.html
America Images	http://rs6.loc.gov/amhome.html
Arizona Images	gopher://miles.library.arizona.edu:70/1
Brazil Images	http://guarani.cos.ufrj.br:8000/Rio/Todos.html
California Images	http://www.water.ca.gov/Calif.images.html
California Photo Museum	http://cmp1.ucr.edu/
Hawaii Images	http://www.mhpcc.edu/tour/Tour.html
Hypatia Images	http://hypatia.gsfc.nasa.gov/Images
Indiana Images	gopher://emif.astro.indiana.edu:70/11/images
New Mexico Images	http://www-swiss.ai.mit.edu/philg/new-mexico/album.html
Utah Images	http://wings.buffalo.edu/staff/plewe/so-utah/images.html
Washington DC Images	gopher://calypso.oit.unc.edu:70/11/sunsite.d/politics.d/multimedia.d
WWW Images Examples	http://www.cit.gu.edu.au/images/Images.html
Index of Banners	http://strauss.ce.cmu.edu:2000/cgi-bin/banner
Fractal Art Gallery	http://www.cnam.fr/fractals.html
Mathemartical Art	http://sunsite.unc.edu/pics/mathgif.html
Aviation Art	http://146.245.2.151/rec/air/air.html

Table 12-3 Icons, buttons, bars, bullets, and flags

Archive Name	URL
Yahoo Server Icons	http://akebono.stanford.edu/yahoo/Computers/World_Wide_Web/Programming/Icons/
Yahoo Server Flags	http://akebono.stanford.edu/yahoo/Reference/Flags/
WWW Power Index	http://www.webcom.com/power/icons.html
Anthony's Icon Library	http://www.cit.gu.edu.au/~anthony/icons/
Daniel's Icon Archive	http://www.jsc.nasa.gov/~mccoy/Icons/index.html

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00375

Continued from previous page

Archive Name	URL
Example WWW Icons	http://www.cit.gu.edu.au/images/Images.html
Icon Archive (Trancy1)	http://ivory.nosc.mil/html/trancy/html/icons-lcc.html
Icon Browser	http://www.di.unipi.it/iconbrowser/icons.html
Index of Flags	http://155.187.10.12/images/flags/
Index of Icons	http://www.bsdi.com/icons/Alcons/
Index of NCSA Icons	http://www.ncsa.uiuc.edu/General/Icons/
Network Services	http://www-ns.rutgers.edu/doc-images
Space Icons	ftp://explorer.arc.nasa.gov/pub/SPACE/LOGOS
Standard Icons	http://www.cs.indiana.edu/elisp/w3/icons.html
Virtual Icon Collection	http://ials.ucsd.edu/y/OhBoy/icons.html
World Flags	http://www.adfa.oz.au/CS/flg/

Table 12-4 Your source for Web movies and animations of all sorts

Archive Name	URL
Thant's Animations Index	http://mambo.ucsc.edu/psl/thant/thant.html
Andrew Tong's Archive	http://www.ugcs.caltech.edu/~werdna/mpegs.html
Index of MPEGs	http://www-psrg.lcs.mit.edu/~rweiss/mpeg.html
MPEG Movie Archive	http://www.acm.uiuc.edu:80/rml/Mpeg/
Another MPEG Archive	http://www.eeb.ele.tue.nl/mpeg/index.html
A 3D Frog	http://george.lbl.gov/ITG.hm.pg.docs/Whole.Frog/Whole.Frog.html
Aquarium Movies	http://www.actwin.com/fish/index.html
Army Computing	http://www.arc.umn.edu/html/ahpcrc.html
Berkeley Quicktimes	http://cmsl.berkeley.edu/Video/QTlist.html
Big Bang Animation	http://www.ncsa.uiuc.edu/evl/cosmic/cosmic.html
Buena Vista Pictures	http://bvp.wdp.com/
Calypso	gopher://calypso.oit.unc.edu:70/11/../.pub/multimedia/animation/mpeg
Cocteau Twins Videos	http://garnet.berkeley.edu:8080/cocteau.html
Collection of Videos	http://www.st.nepean.uws.edu.au/pub/movies/mpeg/
Computer Graphics	http://mambo.ucsc.edu/psl/cg.html
Cornell Visualizations	http://www.tc.cornell.edu/Visualization/vis.html
CRS4 Animation Gallery	http://www.crs4.it:/Animate/Animations.html
Facial Animations	http://www.cs.ubc.ca/nest/imager/contributions/forsey/dragon/top.html
Figure Skating	http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/loosemore-sandra/skate.html
Formula 1 Racing	http://www.obelcrd.co.uk/Formula1/bin.html
Fractal Animations	http://www.cnam.fr/fractals.html

Google/YouTube Exhibit 1005/1105
Page 00376

Archive Name	URL
Galaxy Animations	http://zebu.uoregon.edu/movie.html
Hang Gliding	http://cougar.stanford.edu:7878/MoviePage.html
Japanese Quicktime	ftp://ftp.iij.ad.jp/pub/info-mac/grf/Quicktime
Juggling Movies	http://www.hal.com/services/juggle/animations/
Kaleidospace Movies	http://kspace.com/KM/stream.sys/screenlist.html
Kylie Minogue's Videos	http://www.eia.brad.ac.uk/kylie/index.html
Married with Children	http://www.eia.brad.ac.uk/mwc/index.html
MCNC Quicktime Library	http://www.cnidr.org:5050/quicktime.html
Military Animations	file://dsn-sgi.nosc.mil/pub/movies
NASA's F-16 Project	http://dvol-www.larc.nasa.gov/F16XL/index.html
Pearl Jam	http://fierce.sure.umich.edu/users/galvin/mpeg/pjmpeg.html
Pixel Motion Gallery	http://atea.hypercomp.ns.ca/pix/pix_dragon_gal.html
Planet Earth MPEGs	http://white.nosc.mil/movies.html
Public FTP Archive	http://ftp.arl.mil/images.html
Quicktime Collection	gopher://informns.k12.mn.us/11/best-k12/images/Quicktime%20Movies
Quicktime Guide	http://www.ncsa.uiuc.edu/SDG/Software/MacMosaic/QuickTimeGuide.html
Raytraced MPEGs	http://www.santafe.edu/~nelson/mpeg/
Ren and Stimpy	gopher://kazak.nmsu.edu/11/Public/Ren%20%26%20Stimpy/movies/quicktime
Rob's Multimedia Lab	http://www.ocm.uiuc.edu/rml/Mpeg/
Rugby	http://rugby.phys.uidaho.edu/rugby/Action/edwards.html
Science Theatre	http://www.ncsa.uiuc.edu/SDG/DigitalGallery/DG_science_theater.html
SIGGRAPH Showcase	http://www.ncsa.uiuc.edu/evl/showcase.html
Silicon Graphics Movies	http://www.sgi.com/free/gallery.html#movie
Space Movies	ftp://bezo.lpl.arizona.edu
Space Movies	http://www.univ-rennes1.fr/ASTRO/anim-e.html
Star Trek	http://sherlock.berkeley.edu/docs/info/star_trek/trek.html
Sumex Quicktimes	gopher://sumex-aim.stanford.edu/11/info-mac/Graphic/Quicktime
Sunet Animations	ftp://ftp.sunet.se/pub/movies
Sunsite Multimedia	file://sunsite.unc.edu/pub/multimedia/
Supermodels	http://www.cs.buffalo.edu/~skumar/video.html
Tonga Movies	http://litsun35.epfl.ch:8001/tonga/
The Beatles Quicktime	http://erathost.cfa.ilstu.edu/oratGopherHome/oratClasses/
The MPEG Group	http://www.crs4.it/HTML/LUIGI/MPEG/mpegfaq.html
Tori Amos Videos	http://www.mit.edu:8001/people/nocturne/tori/quicktime.html
USC Quicktime Archives	http://cwis.usc.edu:80/dept/etc/hollynet/movies/
Virtual Reality	http://www.sics.se/dce/dive/dive.html

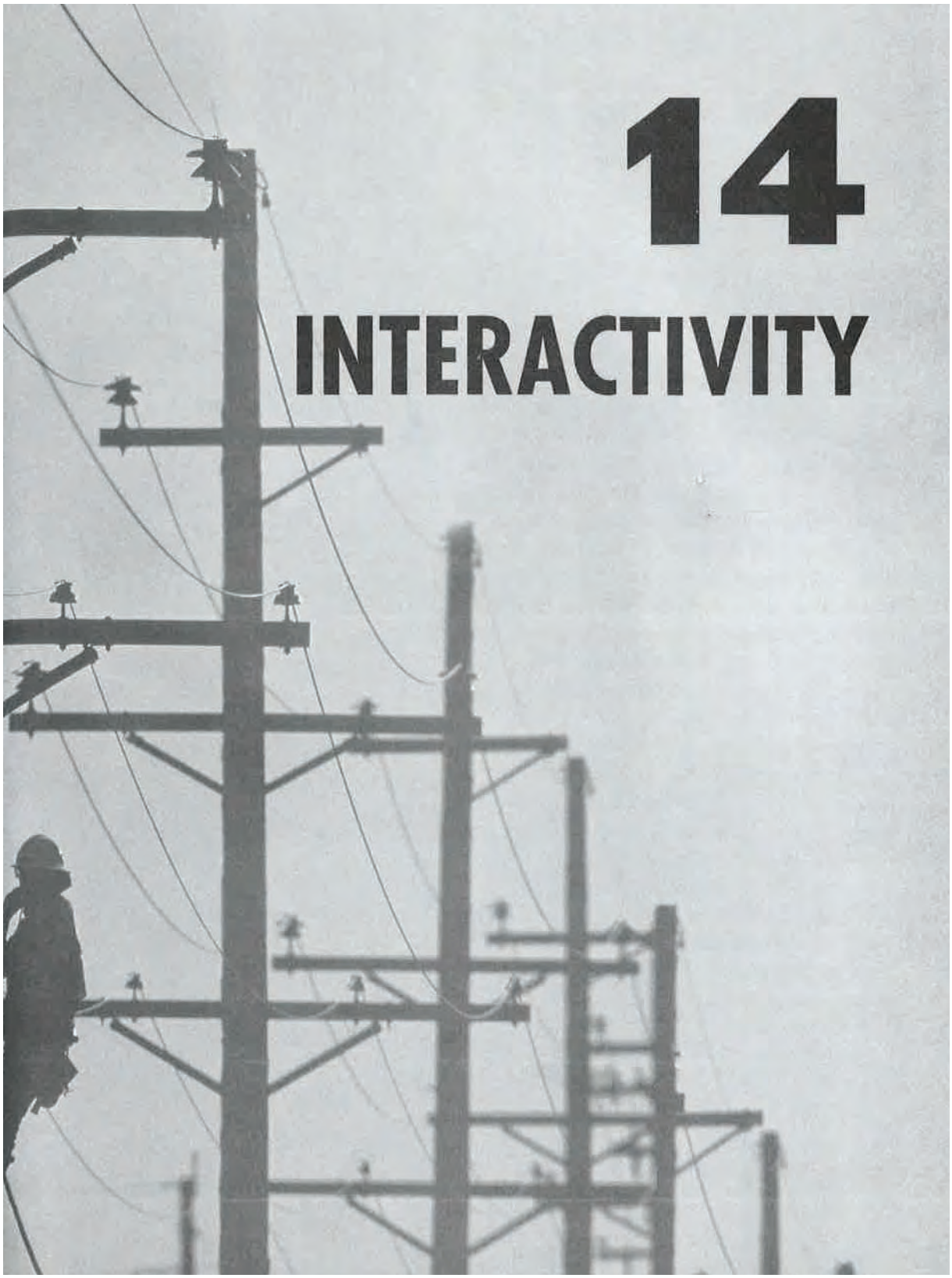
Google/YouTube Exhibit 1005/1105
Page 00377

WHAT NOW?

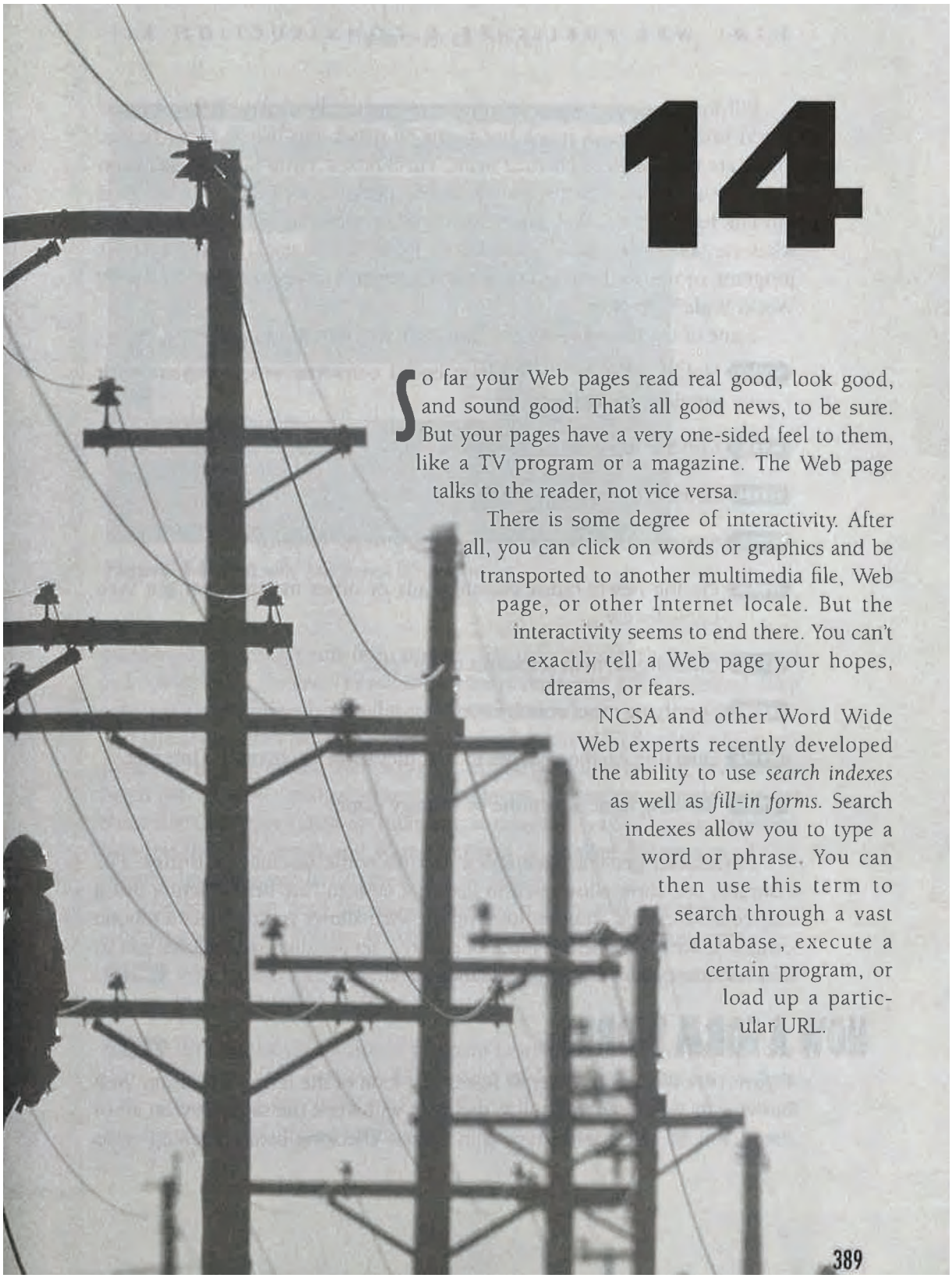
What now, you ask? Well, you can turn to the next chapter and add sound to your already multimedia Web page. Chapter 15 covers CGI Scripts—various programs you can run using your Web page. Many of these programs can be used hand-in-hand with graphics, achieving super-special effects.

If you want to start putting your pages in the world-wide limelight, turn to Chapter 20, Where to Place Your HTML Documents.

Google/YouTube Exhibit 1005/1105
Page 00378



14 INTERACTIVITY



14

So far your Web pages read real good, look good, and sound good. That's all good news, to be sure. But your pages have a very one-sided feel to them, like a TV program or a magazine. The Web page talks to the reader, not vice versa.

There is some degree of interactivity. After all, you can click on words or graphics and be transported to another multimedia file, Web page, or other Internet locale. But the interactivity seems to end there. You can't exactly tell a Web page your hopes, dreams, or fears.

NCSA and other Word Wide Web experts recently developed the ability to use *search indexes* as well as *fill-in forms*. Search indexes allow you to type a word or phrase. You can then use this term to search through a vast database, execute a certain program, or load up a particular URL.

Google/YouTube Exhibit 1005/1105
Page 00380

Fill-in forms give you even more interactive flexibility. Forms on the World Wide Web work much like forms on paper, but they're easier to use. There are text fields, numerical fields, checkboxes, radio buttons, selection lists, pop-up menus, and more. Once you create one of these forms, users can talk back to your Web page. You can then take this information and do whatever you wish with it. You can save it, process it, use it to steer another program, or use the form to automatically create a made-to-order, on-the-fly World Wide Web page.

Some of the places forms and indexes have been used include:

- HTML Holding nearly-live bulletin-board conversations, complete with articles and replies
- HTML Conducting polls or surveys of the World Wide Web community
- HTML Ordering, shipping, or selling products
- HTML Signing up to join online services, groups, or clubs
- HTML Placing newly-typed classified ads or other messages on the Web immediately
- HTML Collecting e-mail responses to surveys
- HTML Specifying exact search criteria for online indexes or databases
- HTML Directing Archie searches to find files somewhere on the Internet
- HTML Running basic adventure or strategy games

Forms can get really complex and do some fascinating things. For example, one form allows you to design a custom "lite-brite" picture using tiny colorful circles. Another form on the Web allows you to direct a robotic camera toward a particular piece of scenery. Yet another form allows you to add text to a collaborative hyperfiction novel.

HOW A FORM WORKS

Figure 14-1 illustrates a sample form. The look of the form varies from Web browser to Web browser, but in theory it will work the same way on all of them. You fill out a form, typing in values, checking boxes, clicking radio

Google/YouTube Exhibit 1005/1105
Page 00381

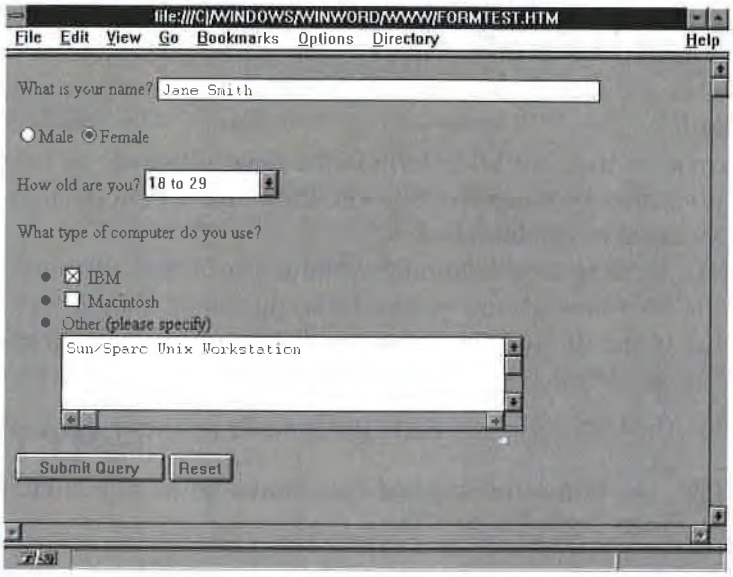


Figure 14-1 A fully functional fill-in form

buttons, or selecting items from menus. You then usually click some sort of Submit or Send button. The form's data is then sent to a specified Web server, converted into a special list of names and values.

The server then passes on these values to a special *CGI script*, which is a program running on the Web server. It can be a shell script, a C program, a batch file, whatever makes sense to the person writing the program. CGI is short for Common Gateway Interface, a standard way of communicating between the Web server and your computer. A detailed description of CGI programming is given in Chapter 15, CGI Scripts.

The script can read the values from the form and analyze, save, or otherwise use them. Writing forms, then, requires two important steps:

HTML Writing the Web page form itself in HTML. This chapter covers the ins and outs, the ups and downs.

HTML Writing the CGI script or program to interpret the form entries. The next chapter discusses CGI scripts in detail, with sections on how to extract data from a form, what to do with that data, and how to create customized forms on-the-fly.

Google/YouTube Exhibit 1005/1105
Page 00382

LESSON #1: A FORM IS BORN

All forms begin, as you might guess, with the <FORM> tag, and end with the </FORM> tag.

You can have more than one fill-in form in the same Web page, as long as each form is located between its own <FORM> and </FORM> tags. Forms cannot be nested or combined.

The <FORM> element means nothing without the proper attributes. Attributes tell the Web browser the type of form, the way it should look, and exactly what it should do. The attributes that work with <FORM> include:

- HTML ACTION: The URL of the computer that actually processes the form
- HTML METHOD: The http server method that should be used to submit the fill-in form
- HTML ENCTYPE: The option to encode the fill-in form's contents before they're sent to the server

STYLE TIP: Not all browsers clearly delineate forms from the rest of the Web page. You should draw a horizontal rule with <HR> before and after each form, to clearly mark it as a separate entity. This is especially important if you have more than one form on the page.

ACTION

The ACTION attribute lets the Web browser know where to send the form's contents. In most cases, this is a specific cgi-bin script or other program. These server scripts are discussed in more detail in the next chapter.

If your server script is called *formprog*, for example, and is stored in the *cgi-bin* directory of the current *www.smartpants.edu* server, then your ACTION attribute would look like:

```
<FORM ACTION="http://www.smartpants.edu/cgi-bin/formprog">
```

The ACTION attribute is optional. If you omit it, then the current Web page's URL will be used.

METHOD

When a form is filled out and submitted, the data can be sent in one of two methods:

Google/YouTube Exhibit 1005/1105
Page 00383

HTML GET: The form's contents are appended to the URL with a question mark, similar to a search query.

HTML POST: The form's contents are sent to the server as a message.

The METHOD attribute lets the Web browser know exactly how the data should be sent. The default is GET. Most modern servers, however, can handle the POST method, which is usually more reliable. The GET method often causes errors when it tries to send too much information. To specify the POST method in HTML, you would type:

```
<FORM ACTION="http://www.smartpants.edu/cgi-bin/formprog" METHOD=POST>
```

ENCTYPE

Use the ENCTYPE attribute if you want to encode the data you send. You can only encode the form's contents if your METHOD is set to POST. As of now, there is only one type of supported encoding (application/x-www-form-urlencoded). In most cases, you do not need to use this attribute.

LESSON #2: INPUT FIELDS

Once you let your Web page know it's dealing with a fill-in form, you can start creating fields. Most fields are specified using the <INPUT> element.

The <INPUT> tag is an empty element (there is no </INPUT> tag, though some people insert one for clarity). You can have as many input fields within a given form as you wish

The most important attribute for any input field is the NAME; a unique name assigned to each field. Input tags can be used to create text fields, checkboxes, radio buttons, submit buttons, or reset buttons. These various fields are specified using the TYPE attribute. Finally, the DISABLED attribute lets you create a field that doesn't actually accept data.

Other possible attributes are used only in specific situations, and are generally optional. These attributes will be discussed later in this chapter, as they become relevant, but here is a quick reference list:

HTML ID: The field ID number

HTML CHARSET: The character set

HTML SIZE: The size of the field

Google/YouTube Exhibit 1005/1105
Page 00384

- MIN: The minimum value
- MAX: The maximum value
- MAXLENGTH: The maximum length
- VALUE: The default value for the field
- CHECKED: The default state of a check box or radio button (checked or unchecked)
- ERROR: The error-testing flag
- SRC: The source filename or URL for an image to be used instead of a button
- ALT: The alternative text for text-based browsers
- ALIGN: The placement of the image (can be set to Top, Middle, or Bottom)

TYPE

Each INPUT tag should include the TYPE attribute. You can choose from the following types of fields:

- Text: A plain text entry field
- Password: A text entry field where any letters the user types are printed as asterisks
- Int: An entry field that only accepts integers
- Check box: A box that can either be checked or unchecked
- Radio: A radio button that precedes several items; the user can choose one, and only one, of these items
- Submit: A button to press when the form is completely filled out
- Reset: A button that erases all fields and snaps everything back to their default values

Google/YouTube Exhibit 1005/1105
Page 00385

HTML Hidden: A field name and value that is not visible to the user. The "hidden" TYPE is useful for sending parameters in a form that are not input by the form user.

Each of these types is discussed in its own section later in this chapter.

NOTE: When you define a TYPE you can enclose in it quotes, such as TYPE="hidden" or just leave it bare, as in TYPE=hidden. Any browser that supports forms should not care. Use whichever style you're most comfortable with.

Other input types are currently being developed for HTML3. Most Web browsers and servers do not support them yet, but it's useful to know what you may be able to do in the near future. These types include:

HTML Float: A field that can hold a floating point (decimal) number

HTML Date: A field that only accepts a legal date

HTML URL: A field that only accepts a legal URL

HTML Range: A control bar, similar to a thermostat control, allowing a user to pick a number between specified low and high values

HTML Audio: An input field that would allow sound bites

HTML File: The filename of a file stored on the user's computer

HTML Scribble: A blank field that would allow the user to scrawl a message using a lightpen, mouse, or image tablet

HTML Image: An input field that would allow an image

NAME

The NAME attribute is mandatory for every type of input field except for the Submit and Reset buttons. Every question you want to ask a fill-in form user must be designated by a unique name, specified by the NAME attribute. This name should ideally describe the field. For example, if you're creating a field for your readers to type their last names, your markup would begin with:

```
<INPUT NAME="lastname">
```


Google/YouTube Exhibit 1005/1105
Page 00386

A field's name is not actually displayed to the user. This information is merely sent to the server script where it can be processed.

DISABLED

If you include the DISABLED attribute with any field, it usually appears grayed out. Users cannot type or select a disabled field. The DISABLED attribute is specified in HTML3. You should note, however, that most current browsers do not support disabled input fields.

Disabled fields are useful for complex forms that you distribute to many different people. If a given question isn't applicable to a specific person or situation, you can gray out the field. This way, people will know that the question *exists*, just not for them.

For example, a user may fill out one form only to be returned another form. In the first form, the user might have specified that he isn't married. The second form could then have a disabled field:

```
<P>What is your spouse's name? <INPUT DISABLED NAME="spousename">
```

STYLE TIP: Disabling makes things easier for the Web developer, as well. Instead of going through an HTML document and deleting unused fields, changing the look of your Web page, you can just gray these fields out.

MIN/MAX

Some browsers allow you to specify a minimum and maximum value for integer fields. This way, you can limit the numbers that people can actually submit. Simply specify a MIN and MAX attribute.

For example, if you want people to enter a number between 90 and 100, you could type:

```
<P>Type a number between 90 and 100: <INPUT NAME="number" TYPE=int
MIN=90 MAX=100>
```

LESSON #3: TEXT FIELDS

A text field is a box where a user can type some information. You can get fancy with text fields, specifying values for them to take if the user doesn't enter anything, masking the input with asterisks for added security, or limiting input to a numerical value. Figure 14-2 shows several types of text fields.

Text Field Variations

To specify a *plain text* field, all you have to do is use the <INPUT> element and name the field. You could use the TYPE=text attribute, but it's easier to

Google/YouTube Exhibit 1005/1105
Page 00387

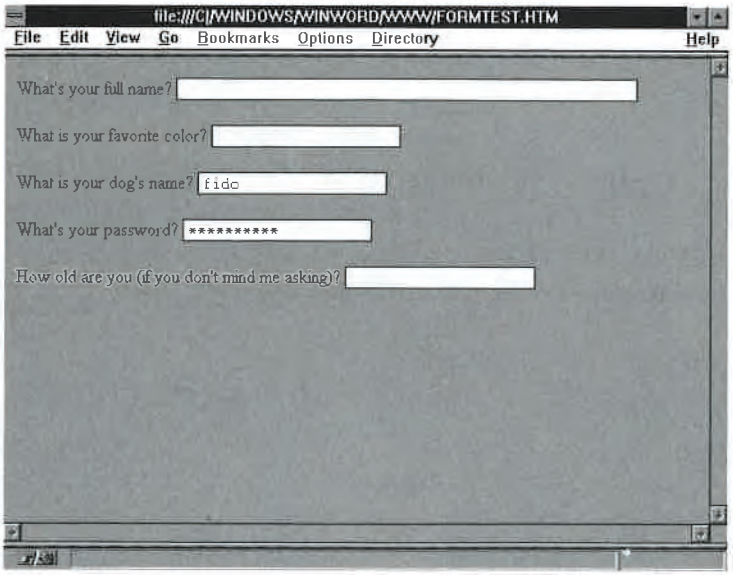


Figure 14-2 Two plain text fields, a text field with a default value, a password field, and an integer field

leave it out; plain text is the default TYPE, so that's what you get unless you specify something else. For example, to create a field that asks for somebody's favorite color, you would type:

```
<P>What is your favorite color? <INPUT NAME="color">
```

When you're dealing with credit card numbers, passwords, or any other secure information, you don't want them left out in the open, so you use *password* fields. As a user types in this kind of field, asterisks hide the values. This way, if somebody is peeping over the typist's shoulders, the intruder won't be able to tell what was just typed. You create a password field by adding TYPE=password to the <INPUT> element. So, to create a field that asks a person for a password, you would type something similar to:

```
<P>What's your password? <INPUT NAME="password" TYPE=password>
```

If you want to make sure you don't get alphabet soup, you can use the last type of "text" field: the *integer* field. This field only accepts integers (whole numbers, negative or positive). To create an integer field, you use TYPE=int in the <INPUT> element. A typical integer field might look like:

```
<P>How old are you (if you don't mind me asking)? <INPUT NAME="age"
TYPE=int>
```


Google/YouTube Exhibit 1005/1105
Page 00388

VALUE

The VALUE attribute lets you assign a default value to a text, password, or integer field. When the form first appears, or when the form is Reset, this value is already written in the input box.

You can also use values to show a user what sort of information he or she needs to type. For example, when you ask for the name of the user's dog, you can include a common name as the default value:

```
<P>What is your dog's name? <INPUT NAME="dogname" VALUE="Fido">
```

SIZE

You can designate how long the input box should be by using the SIZE attribute. The default size is 20 characters. If you want to create an extra-big text entry field, you could type something similar to:

```
<P>What's your full name? <INPUT NAME="name" SIZE=50>
```

If you want your text entry field to be more than one line long, you should use a textarea, which is discussed later in this chapter.

MAXLENGTH

No matter how big your input box is, a user can usually continue typing as much text as he or she wants. The text scrolls within the box. If you want to limit the length of the input, specify this limit with the MAXLENGTH attribute. If a user tries typing beyond your MAXLENGTH, the browser will usually beep or display some sort of error message.

For instance, to limit a number to three digits, you could type:

```
<P>Enter your age: <INPUT NAME="age" TYPE=int SIZE=4 MAXLENGTH=3>
```

STYLE TIP: Since scrolling text is often difficult to read, it's a good idea to set MAXLENGTH less than the SIZE of a field.

LESSON #4: CHECKBOXES OR RADIO BUTTONS

Checkboxes or radio buttons can be used to quickly express a lot of information. Figure 14-3 illustrates several checkboxes and radio buttons.

VALUE

Each radio button should have its own value, as expressed with the VALUE attribute. If the button is selected, this value is then sent to the Web server along with the radio button's NAME.

Google/YouTube Exhibit 1005/1105
Page 00389

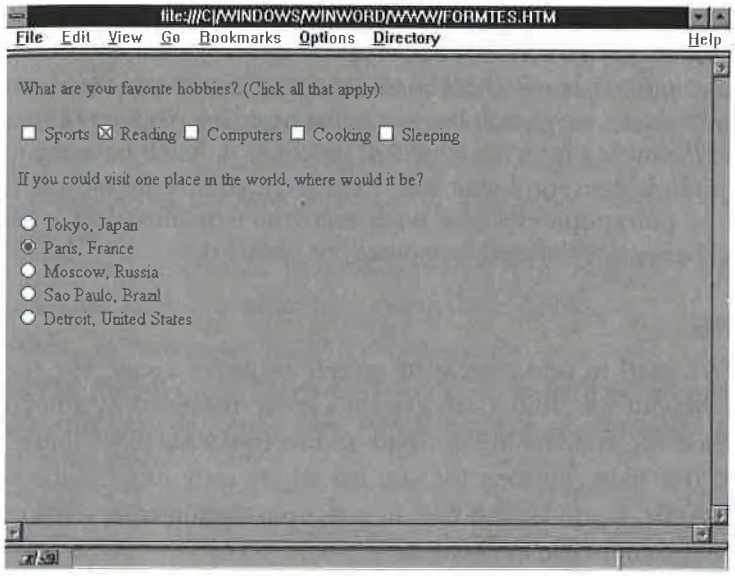


Figure 14-3 A selection of checkboxes and radio buttons

If you want, you can specify a VALUE for each checkbox as well. Checkboxes have two default values (on or off), but you can define other values for them if you wish. Each checkbox must have a NAME. If a checkbox is selected, this name is sent to the Web server along with any given value; if no value was specified, then the word "on" is simply sent. If a box is unselected, then nothing at all is sent to the server. If you desire, you can give several checkboxes the same NAME and different VALUEs. A group of radio buttons should *always* have the same NAME and different VALUEs.

In any case, the VALUE is usually a descriptive word or short phrase. When you create your cgi-script, you will need to refer to this value. Use something obvious.

Checkboxes

A checkbox usually precedes any item or value that is either true or false. For example, you can list several hobbies and allow users to check any hobby they're interested in. If you wanted the hobbies to be listed on the same line, the markup to do this would be:

```
What are your favorite hobbies? (Click all that apply):<P>
  <INPUT NAME="sport" TYPE=checkbox> Sports
  <INPUT NAME="read" TYPE=checkbox> Reading
  <INPUT NAME="computer" TYPE=checkbox> Computers
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00390

Continued from previous page

```
<INPUT NAME="cook" TYPE=checkbox> Cooking
<INPUT NAME="sleep" TYPE=checkbox> Sleeping<P>
```

STYLE TIP: Don't put too many checkboxes or radio buttons on the same line, since some browsers may scroll the text to the next line. You may get a checkbox on a different line from the label that describes it. If you have many checkboxes or radio buttons, or if your labels are long, you should include a line break <BR> or paragraph <P> after each item. You may also want to list a bunch of checkboxes in a bulleted, numbered, or menu list.

Radio Buttons

Radio buttons are used to select between several exclusive items, like the tuning buttons on your car radio. Each item in a group preceded by a radio button should have the same NAME attribute and its own VALUE attribute.

You should use radio buttons for any list where only one choice is allowed. For example, if you want a user to select one option from a list of hot tourist sites, you could use the markup:

```
If you could visit one place in the world, where would it be?<P>
  <INPUT NAME="place" TYPE=radio VALUE="japan"> Tokyo, Japan<BR>
  <INPUT NAME="place" TYPE=radio VALUE="paris"> Paris, France<BR>
  <INPUT NAME="place" TYPE=radio VALUE="moscow"> Moscow, Russia<BR>
  <INPUT NAME="place" TYPE=radio VALUE="saopaulo"> Sao Paulo,
Brazil<BR>
  <INPUT NAME="place" TYPE=radio VALUE="detroit"> Detroit, United
States<BR>
```

You can have as many groups of radio buttons on a form as you wish. For example, the following markup is perfectly legal:

```
Sex: <INPUT NAME="sex" TYPE=radio VALE="male"> Male <INPUT NAME="sex"
TYPE=radio VALE="female"> Female<P>
What amount of schooling have you had:<P>
<INPUT NAME="degree" TYPE=radio VALE="hs"> High School<BR>
<INPUT NAME="degree" TYPE=radio VALE="bs"> College<BR>
<INPUT NAME="degree" TYPE=radio VALE="ma"> Graduate School
(Masters)<BR>
<INPUT NAME="degree" TYPE=radio VALE="phd"> Graduate School
(Doctorate)<BR>
```

Prechecked

The CHECKED attribute allows you to set the default value for any checkbox or radio button. If you want to check any checkbox or select any given radio button, just include the CHECKED attribute:

```
<INPUT NAME="computer" TYPE=checkbox CHECKED> Computers
```

STYLE TIP: You shouldn't use the CHECKED attribute for more than one radio button within a given NAME group.

Google/YouTube Exhibit 1005/1105
Page 00391

LESSON #5: THE RESET AND SUBMIT BUTTONS

No form is complete without a Reset button and a Submit button. The Reset button erases any values already filled in and returns all checkboxes, radio buttons, menus, or textareas to their default values. The Submit button gathers up all the data in the form and sends it off to the server to be processed.

These two buttons are usually placed at the bottom of a form, as shown in Figure 14-4. The markup to create these buttons is:

```
<INPUT TYPE=submit> <INPUT TYPE=reset><P>
```

STYLE TIP: You can place the Submit and Reset buttons anywhere you wish within a form. In some cases, they might look good next to a text entry field or at the top of a form.

As soon as the user clicks the Submit button, the browser sends off the form's entries. You can include several Submit buttons if you wish, each one performing a different action with the same data. For example, you can have a form that asks the user for a search term. You can then have several Submit buttons for various types of databases or search indexes. Each Submit button should have its own NAME attribute.

Customizing the Label

Different browsers use different labels for these buttons. Where some say "Submit," others use "Send," and many use "Submit Query." The Reset button sometimes says "Reset" and sometimes says "Clear."

HTML If you don't want to take your chances with the user's browser, you can use the VALUE attribute to easily replace the labels of these push buttons with your own text.

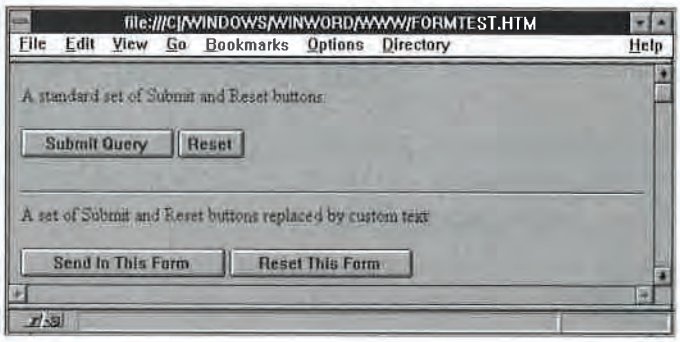


Figure 14-4 The Submit and Reset buttons

Google/YouTube Exhibit 1005/1105
Page 00392

For example, if you want your Submit button to read "Send In This Form," just use the markup:

```
<INPUT TYPE=submit VALUE="Send In This Form">
```

Using Icons

The latest specification of HTML+ allows you to use a special icon instead of the browser's default Submit or Clear buttons. Just specify a SRC, ALT, and ALIGN attribute, the way you would for the <IMG> element. You can even use an imagemap, if you want to get really fancy.

Most browsers do not currently allow you to use an icon. Once iconed buttons become more popular, however, you can use markup similar to the following:

```
<INPUT TYPE=submit SRC="submit.gif" ALIGN=middle ALT="[Submit]">
Submit<P>
```

This would allow you to use the picture of a pointing finger, a green GO sign, or any other graphic you wanted. Such Submit buttons would make a form look spiffy, and be easy for the user to understand.

NOTE: Any SRC attributes within the submit INPUT element are usually just ignored if the browser doesn't support HTML3.

LESSON #6: TEXTAREAS

If you want to create a multiline text entry field, you should use the <TEXTAREA> element. The element requires a </TEXTAREA> closing tag.

A textarea can be used as the message box for e-mail or newsgroup articles, for short answers to questions, or for entire addresses. Textarea fields have both vertical and horizontal scroll bars; you can type as much information as you want.

Much like the INPUT element, you need to specify a NAME attribute. You can also specify the DISABLED attribute if you want the field to be grayed out. If you want your textarea to contain any default text, write it between your <TEXTAREA> and </TEXTAREA> elements. You don't need to specify a line break or paragraph, just press [ENTER] at the end of each line. For example, to create the textarea shown in Figure 14-5, you would type:

```
<TEXTAREA NAME="address" ROWS=5 COLS=50> [ENTER]
John Smith [ENTER]
Biology Department [ENTER]
Smartpants University [ENTER]
Anytown, AW 12345 [ENTER]
</TEXTAREA> [ENTER]
```


Google/YouTube Exhibit 1005/1105
Page 00393

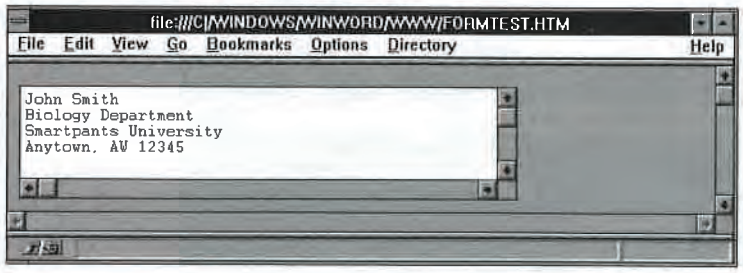


Figure 14-5 A textarea with a default address filled in

Size

By default, a textarea is only 1 line high and 40 characters wide. In most cases, you want to create a custom size for your textarea. Specify the height of your box with the ROWS attribute, and specify the width with the COLS attribute.

For example, to create a huge box that fills much of the screen (20 characters high by 70 characters wide), you would use the markup:

```
Type your message in the following box:<BR>
<TEXTAREA NAME="message" ROWS=20 COLS=70></TEXTAREA>
```

WRAP

Another useful attribute for the TEXTAREA element is WRAP. If you select WRAP, the textarea has no horizontal scroll bar. Whenever the typist gets to the end of the line, the next word automatically wraps around. This HTML+ attribute does not work with some browsers, in which case the TEXTAREA will function as usual.

As an example:

```
<TEXTAREA NAME="message" ROWS=20 COLS=70 WRAP></TEXTAREA>
```

LESSON #7: SELECTION LISTS

The last type of input field you can create is a selection list. A selection list can be an open menu of several items, a pop-up menu, or any other sort of menu list you could hope to design. The look of a selection list depends on the browser. Figure 14-6 shows some sample lists.

You can include as many selection lists as you like. Each selection list is designated by the <SELECT> element and must be closed with the </SELECT> element.

Google/YouTube Exhibit 1005/1105
Page 00394

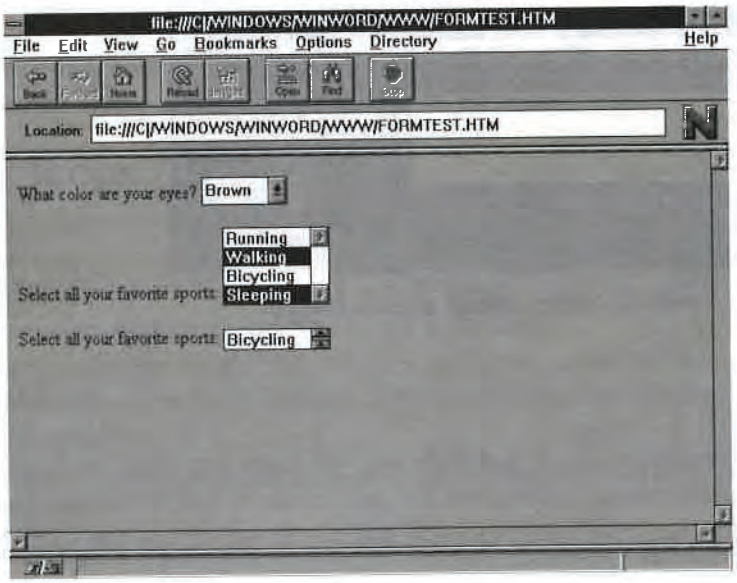


Figure 14-6 A nice selection of selection lists

Your selection list, like any other input type, must have a NAME attribute with a unique identifying name. Each item within a selection list must begin with the <OPTION> element. Although you can leave it out, it usually makes the HTML markup easier to read by closing each option with </OPTION> element.

For example, a basic selection list could look like:

```
What color are your eyes?
<SELECT NAME="color">
  <OPTION>Green</OPTION>
  <OPTION>Brown</OPTION>
  <OPTION>Blue</OPTION>
  <OPTION>Black</OPTION>
</SELECT>
```

SIZE

The size of a selection list can drastically change its look. If the SIZE attribute is omitted, or if the size is set to 1, then the selection list looks like a pop-up menu, as shown at the top of Figure 14-6. Only one item is shown. The user can click on a little arrow to the right of the item to see the full list pop up on the screen. Here's the markup to create a pop up menu:

```
What color are your eyes?
<SELECT NAME="color" SIZE=1>
```


Google/YouTube Exhibit 1005/1105
Page 00395

```
  <OPTION>Green</OPTION>
  <OPTION>Brown</OPTION>
  <OPTION>Blue</OPTION>
  <OPTION>Black</OPTION>
</SELECT>
```

If the SIZE is 2 or more, then the selection list appears as a scrollable menu. For example, if your list has 15 options and a size of 5, then only the first 5 options are shown. To see the remaining options, a user could scroll down through the list using the scroll bar. The size of the list, for obvious reasons, should not be larger than the number of options.

STYLE TIP: You can't place selection lists next to each other or next to more than one line of text. If your SIZE matches the number of list elements, then all items will be visible. This is often a good idea for short lists. If a selection list is very long, however, it takes up many rows and leaves a lot of white space on your Web page screen. You can prevent this by making your lists smaller, and having people scroll through them using the scroll bar.

Multiple Selections

Can users select more than one option from the current list? If your list topic has several possible answers, you should use the MULTIPLE attribute. For example, if you have a list of several hobbies, you may want to allow a user to select several of them:

```
Select all your favorite sports:
<SELECT NAME="sports" SIZE=4 MULTIPLE>
  <OPTION>Running</OPTION>
  <OPTION>Walking</OPTION>
  <OPTION>Bicycling</OPTION>
  <OPTION>Sleeping</OPTION>
</SELECT>
```

STYLE TIP: If you use the MULTIPLE attribute, your selection list cannot be a pop-up menu. It may, however, be scrollable.

Options Attributes

Each <OPTION> element can have its own attributes, including SELECTED and DISABLED.

SELECTED

If you'd like an option to be selected by default, use the SELECTED attribute. If your selection list is MULTIPLE, then you can select several options as defaults.

Google/YouTube Exhibit 1005/1105
Page 00396

For example, in the following markup the "Running" and "Sleeping" options will be selected:

```
Select all your favorite sports:
<SELECT NAME="sports" SIZE=4 MULTIPLE>
  <OPTION SELECTED>Running</OPTION>
  <OPTION>Walking</OPTION>
  <OPTION>Bicycling</OPTION>
  <OPTION SELECTED>Sleeping</OPTION>
</SELECT>
```

DISABLED

As with any other input field, you can disable any option in a selection list by using the DISABLED attribute. Any disabled attribute appears grayed-out and cannot be selected. For example:

```
<OPTION DISABLED>Bicycling</OPTION>
```

STYLE TIP: Many browsers do not understand the DISABLED attribute, so do not rely on disabled options; use these options only for cosmetic reasons, to help suggest which options to avoid.

SHAPE

The latest specification of HTML3 allows you to specify the SHAPE attribute for each option. This would allow you to design the graphical look of the option. Currently, however, there are no browsers that understand this attribute.

LESSON #8: PUTTING IT ALL TOGETHER

To actually create your form, first plan it out on paper. A form can have any text style, headings, or paragraph styles. Don't neglect to use plenty of <BR> line breaks and <P> paragraphs to provide enough spacing between each input field or each caption and the next one. If your form is long, break it up into sections with horizontal lines. Use bulleted or menu lists (perhaps without the <LI> bullet element) for a series of checkboxes or radio buttons. All in all, try to make your form as intuitive as possible to fill out.

STYLE TIP: When you have a list of options, you can present them either as a bunch of checkboxes or as a multiple selection list. In most cases, you should make options that relate to each other into a list and arrange options that are independent from each other as check boxes.

Google/YouTube Exhibit 1005/1105
Page 00397

A Basic Form

The form shown at the beginning of the chapter, in Figure 14-1, was created using the following HTML code:

```
<FORM ACTION="http://www.smartpants.edu/cgi-bin/survey">
   <P>What is your name? <INPUT NAME="name" SIZE="48">
   <P><INPUT NAME="sex"TYPE=radio VALUE="male">Male
   <INPUT NAME="sex" TYPE=radio VALUE="female">Female
   <P>How old are you?
   <SELECT NAME="age">
      <OPTION>Less than 18</OPTION>
      <OPTION>18 to 29</OPTION>
      <OPTION>29 to 39</OPTION>
      <OPTION>More than 40</OPTION>
   </SELECT>
   <P>What type of computer do you use?
   <UL>
       <LI><INPUT NAME="ibm" TYPE=checkbox VALUE="ibm"> IBM
       <LI><INPUT NAME="mac" TYPE=checkbox VALUE="mac"> Macintosh
       <LI><INPUT NAME="other" TYPE=checkbox VALUE="other"> Other
(please specify)<BR>
         <TEXTAREA NAME="others" COLS=48 ROWS=4></TEXTAREA>
   </UL>
    <P><INPUT TYPE=submit> <INPUT TYPE=reset>
</FORM>
```

A Sample Message Form

The easiest and quickest way to start understanding and producing your own forms is to take a look at other forms that contain features that you are interested in using. Once you've looked at the code, you can start cutting, pasting, refining, personalizing...doing whatever you need to make the form do what you want.

The following HTML code describes a page that contains a single form. This form contains most of the major form features, including other common HTML statements. See Figure 14-7 to get an idea of what this form would look like to a client using the X-Windows version of Mosaic. The markup for the entire page is as follows:

```
<HTML>
<HEAD>
 <TITLE>Comment Form</TITLE>
</HEAD>
<BODY>
<H1>Troy's feedback page</H1>
<IMG SRC="/shaggy.a/downing/public-html/face.xbm"><BR>
Let me know what's on your mind....<BR>
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00398

Continued from previous page

```
<HR>
<FORM ACTION="http://found.cs.nyu.edu/cgi-bin/testtbd" METHOD="POST">
Who's calling?<BR>
<PRE>
First Name <INPUT TYPE="text" NAME="first" SIZE=10>
Last<INPUT TYPE="text" NAME="second" size=20>
E-Mail <INPUT TYPE "text" NAME="email" VALUE="yourname@youradd.edu"
SIZE=25>
<INPUT TYPE="checkbox" VALUE="list">Check here to be added to my
mailing list.
</PRE>
<HR>
Select message type:
<SELECT NAME="urgency">
   <OPTION>Urgent!
   <OPTION>Important
   <OPTION>No Hurry
   <OPTION>FYI
</SELECT>
<BR>
   <INPUT TYPE="radio" NAME="Menu" CHECKED VALUE="complain">Complaint
   <INPUT TYPE="radio" NAME="Menu" VALUE="joke">Joke
   <INPUT TYPE="radio" NAME="Menu" VALUE="quote">Quote
   <INPUT TYPE="radio" NAME="Menu" VALUE="greet">Greeting
   <INPUT TYPE="radio" NAME="Menu" VALUE="offer">Offer
   <INPUT TYPE="radio" NAME="Menu" VALUE="suggest">Suggestion
<BR>
<HR>
Tell me about it...<BR>
<TEXTAREA NAME="textbox" ROWS="3" COLS="60">Type your message here...
</TEXTAREA>
<BR>
<INPUT TYPE="submit" VALUE="Submit"><INPUT TYPE="reset" VALIE="Clear
Entries">
</FORM>
<BODY>
</HTML>
```

The CGI script to handle this form is included in the next chapter.

LESSON #9: INCLUDING HIDDEN TEXT

At times, it is important to include variable information without having it appear as part of the form. This information can be a user ID number, a time stamp, or whatever makes sense in each particular case. These cases will be much more apparent after you read the next chapter. To send form data that a user does not explicitly type or select, you can use a hidden field.

NOTE: Users can see a hidden field by viewing the source for the Web page, but the field won't show up in the normal Web browser.

Google/YouTube Exhibit 1005/1105
Page 00399

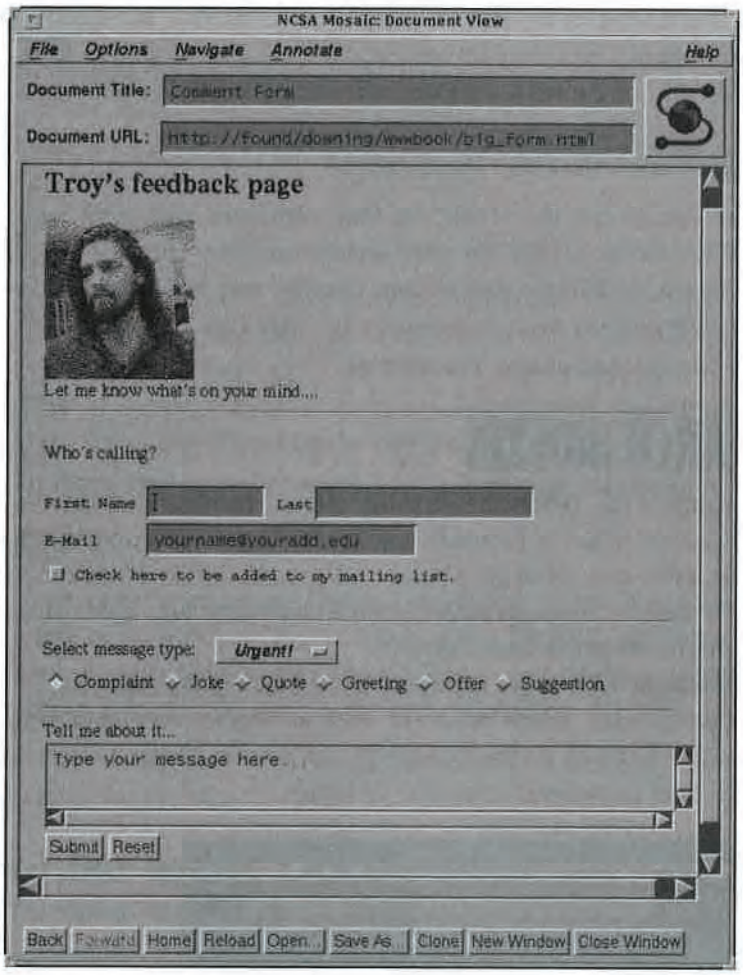


Figure 14-7 A groovy message form

Hidden fields are particularly useful if you want to track a user who is using multiple forms without asking the user to input a name or ID at each level or step of the form. Hidden fields can be created using the INPUT element:

```
<INPUT TYPE=hidden NAME="invisible" VALUE="samplevalue">
```

When a user submits this form, the value pair "invisible=samplevalue" is automatically included. Hidden fields are most useful when you have their values dynamically assigned and included in your forms. Chapter 15, CGI Scripts, will explain how to do this.

Google/YouTube Exhibit 1005/1105
Page 00400

Sometimes it is useful to be able to include static hidden variables as well. One example would be a time or version stamp. If you wanted to keep track of which version of a form a user was accessing your server with, you could include something like:

```
<INPUT TYPE=hidden NAME="version" VALUE="6.1">
```

Now, every time you edited the HTML file that contained this form, you could increment the value "6.1" to the new version number. The CGI script side of your form could evaluate this version number and make a decision based on the version number such as sending a message like

```
"Version number not valid, please reload form."
```

LESSON #10: SEARCH INDEXES

Let's go back in time a bit. Before fill-in forms were created, there was the search index. A search index is basically similar to a form text input field. The user simply types in a word or phrase and presses [ENTER]. This search term can then be used to look up information in an index file. Index files themselves are discussed in the next chapter.

If your collection of Web pages is huge, a good index is essential. Figure 14-8 shows a typical index. Some browsers, such as Mosaic version 1, will display the search index box at the bottom of the screen. Other browsers display the index field anywhere within the Web page.

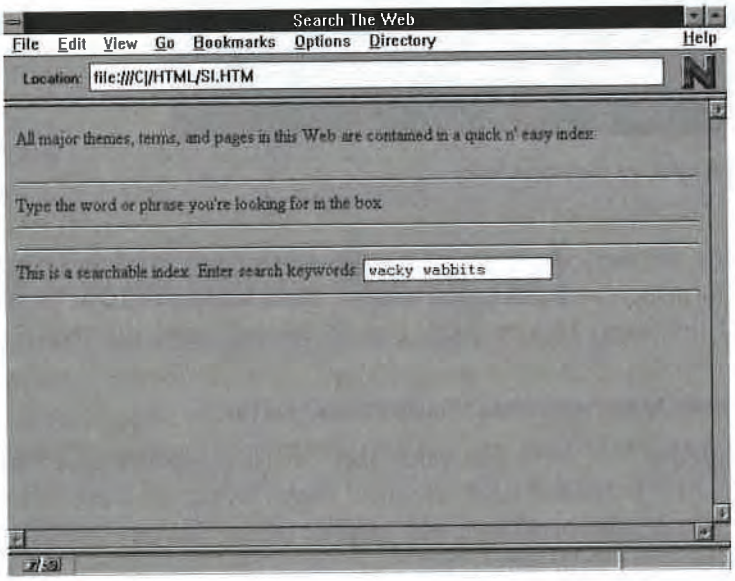


Figure 14-8 A simple search index

Google/YouTube Exhibit 1005/1105
Page 00401

ISINDEX

To prompt the user for a search term, use the <ISINDEX> element.

The index query is usually placed in the <HEAD> section of a document, though the latest specification of HTML+ allows you to place an index request anywhere in the Web page.

HREF

You can use <ISINDEX> by itself, in which case the current Web page must be a search index of some sort; in other words, your Web page is actually a CGI program. You can learn how to create these hybrid cgi-script/Web pages in the next chapter. In most cases, however, your index should point to the script designated by the HREF attribute:

```
<ISINDEX HREF="http://smartpants.edu/cgi-bin/vastindex">
```

PROMPT

By default, the message "This is a searchable index. Enter search keywords:" precedes an index field. You can use the PROMPT attribute to change this label. For example, to ask for a book title, you could use a tag similar to:

```
<ISINDEX PROMPT="Enter a book title here:">
```

Currently, only Netscape understands this attribute, though it has been defined for HTML3.

Book-Like Index Documents

The latest specification of HTML+ allows you to create text-only index documents, similar to the index in a book. An HTML+ index document would list every topic or keyword and include a hyperlink to places where that word appears. These indexes could then be viewed directly, or can be searched. Unfortunately, no current browsers support these indexes.

Whenever you wanted to create an indexed term, you could just use the INDEX attribute, as part of a heading, paragraph, blockquote, or other type of text block:

```
<h3 id="a99" index="Horses/Ponies">Ponies</h3>
```

These indexes could then be used to automatically create an index file with segments like this:

```
Horses
   caring for
   feeding
   history of
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00402

Continued from previous page

```
   ponies
   racehorses
Iguanas
```

Users could then view this index directly, or search it using <ISINDEX>. Once indexes like these become widely accepted, it will be easy to organize vast amounts of text or even sounds, video, and graphics.

LESSON #11: TESTING IT OUT

You can quickly and easily test out your HTML fill-in forms using one of NCSA's public server scripts. If you want to use the POST method, use the following format for your form:

```
<FORM METHOD="POST" ACTION="http://hoohoo.ncsa.uiuc.edu/htbin-
post/post-query">
```

If you want to try out the GET method, use a different server script:

```
<FORM METHOD="GET" ACTION="http://hoohoo.ncsa.uiuc.edu/htbin/query">
```

This gets you a returned Web page that shows exactly what you submitted, and lists the names and values of each of your input fields. A sample page for the message form shown in Figure 14-7 might look like:

```
Query Results
You submitted the following name/value pairs:
• first = David
• second = Fox
• email = dfox@graphics.cs.nyu.edu
• urgency = Important
• Menu = joke
• textbox = Why did the chicken cross the road? Because there was an
earthquake!
```

LESSON #12: WHAT THE DATA LOOKS LIKE

When you submit a form or an index query, every field's name and value is sent to the specified ACTION server or HREF. The data format differs depending on whether you use the GET or POST method.

With search indexes, the browser sends the actual text string the user typed. With fill-in forms, the values that go to the Web server correspond to the input fields. Each input field sends a different type of information:

HTML Text, integer, and password fields: The name of the field, along with whatever the user typed. (If the user typed nothing, then a blank string follows the field name.)

Google/YouTube Exhibit 1005/1105
Page 00403

HTML Checkboxes: The name and value of any checked box. (If the box has no individual value, the browser sends the default value—"on". It sends nothing at all if the box isn't checked.)

HTML Radio buttons: The name of the radio button group, along with the selected value. (It sends nothing if none of the buttons are checked.)

HTML Textareas: The name of the textarea along with any typed text.

HTML Selection lists: The name of the selection list along with the value (text) of the selected option. (If the user selects more than one option, an additional name/value pair is sent for each choice.)

In general, the NAME of each entry in the form is paired with the VALUE of that entry, along with an equals sign (=). Each of these entries is then separated by the ampersand (&). For example, a form with two entries would send data similar to the following:

```
name=value&name=value
```

If the value of a field contains unusual characters such as the ampersand (&) or the equals sign (=), these characters are *escaped*. Escaped characters are represented by a percent sign (%) followed by the hexadecimal (base 16) ASCII number for that character.

In addition to these special characters, spaces embedded in text fields are changed to the plus sign (+). This way, your stream contains no spaces or other strange gaps.

Search Indexes

The value the user types in the search index is known as the *query*. This query is sent tacked on the URL specified with the HREF attribute. For example, you can create an index as follows:

```
<ISINDEX HREF="http://www.smartpants.edu/cgi-bin/index">
```

A user may then type a query, such as "jack daniels." The browser creates the following URL and jumps to it:

```
http://www.smartpants.edu/cgi-bin/index?jack+daniels
```

The GET Method

Much like an index query, The GET method sends all your form's data as part of the URL, which takes the form:

```
action?name=value&name=value&name=value
```


Google/YouTube Exhibit 1005/1105
Page 00404

Date fields, file fields, and URL fields would also be useful things to have. Eventually, fields could be created that allow you to pop in graphics, say something into a microphone, or scribble something using a pen. This would allow you to send as well as retrieve multimedia using the Web. All in all, a Web page could become a full-fledged piece of software in its own right.

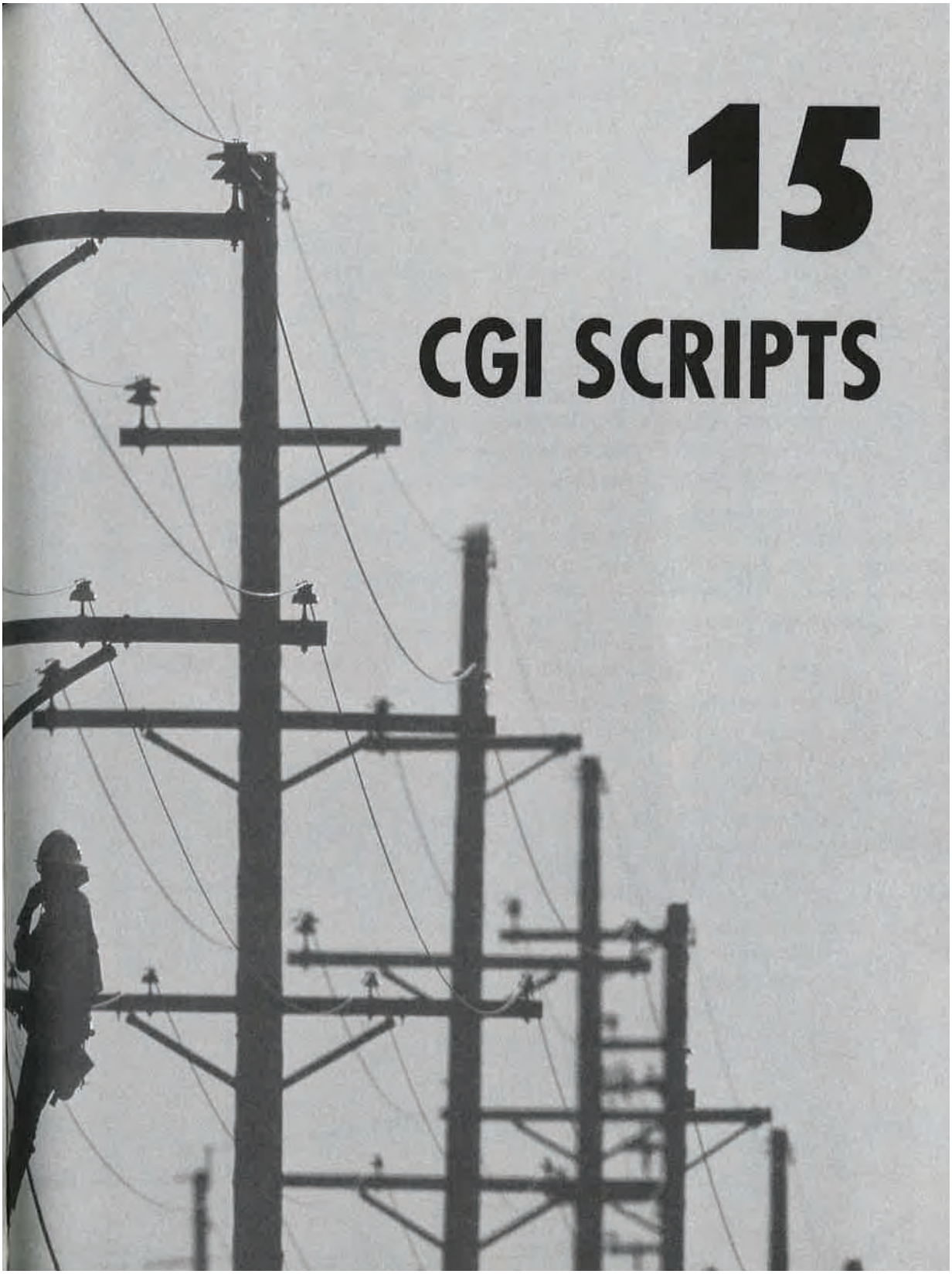
Such developments are underway, and will probably arrive sooner than you can click Submit.

WHAT NOW?

The next essential step in form creation is to write a CGI script or other program to handle your form data and do something useful with it. All sorts of examples, suggestions, and ideas can be found in the next chapter.

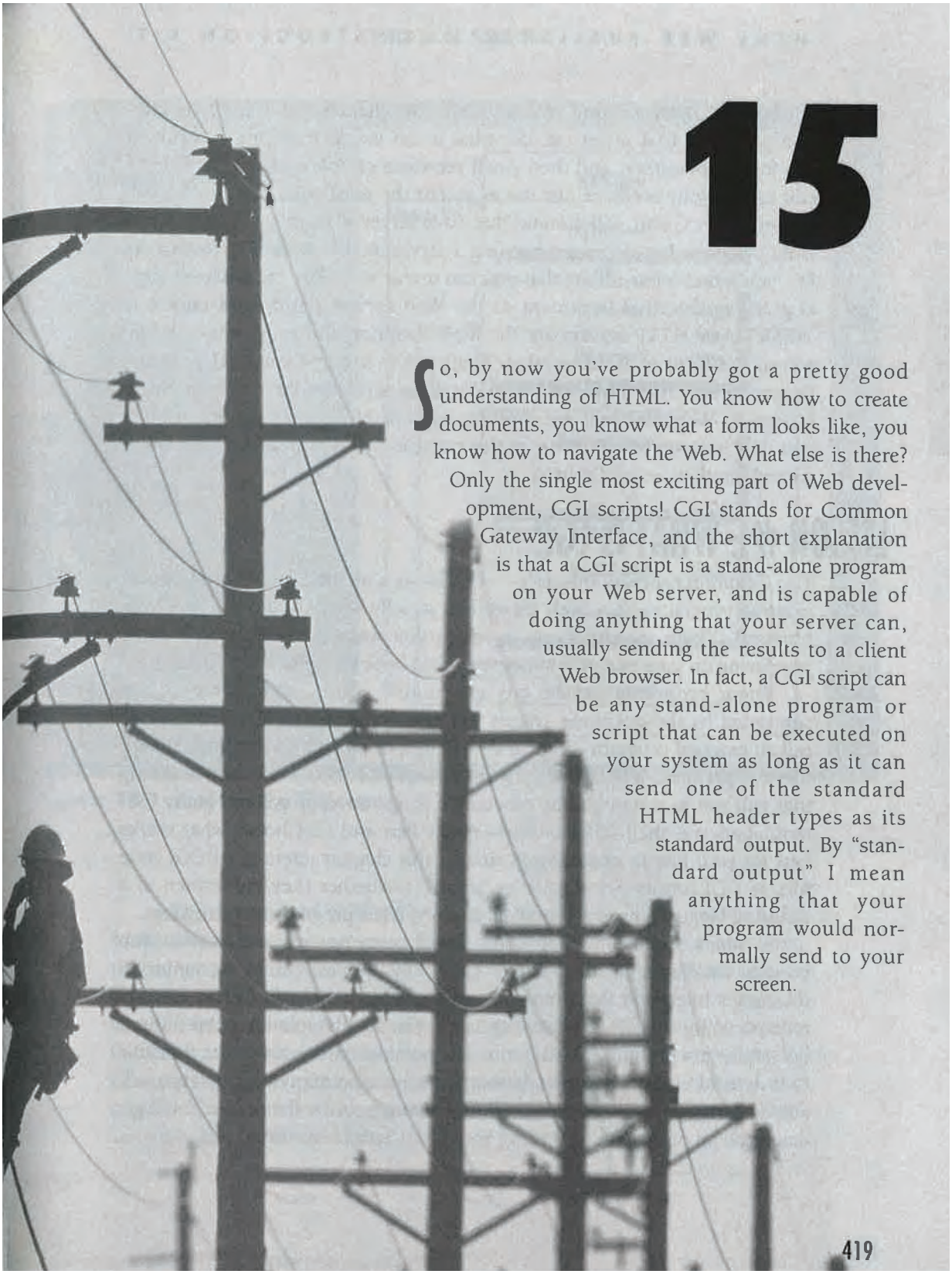
If you haven't yet installed a Web server, then all this information about dealing with forms can seem pretty useless. Read up about Web servers in Chapter 21, Starting Your Own Web Site.

Google/YouTube Exhibit 1005/1105
Page 00405



15

CGI SCRIPTS



15

So, by now you've probably got a pretty good understanding of HTML. You know how to create documents, you know what a form looks like, you know how to navigate the Web. What else is there? Only the single most exciting part of Web development, CGI scripts! CGI stands for Common Gateway Interface, and the short explanation is that a CGI script is a stand-alone program on your Web server, and is capable of doing anything that your server can, usually sending the results to a client Web browser. In fact, a CGI script can be any stand-alone program or script that can be executed on your system as long as it can send one of the standard HTML header types as its standard output. By "standard output" I mean anything that your program would normally send to your screen.

Google/YouTube Exhibit 1005/1105
Page 00407

Sounds pretty exciting, doesn't it? In this chapter, you'll get a description of what a CGI script can do, what it can use as its input, and how it can format its output, and then you'll see some sample scripts that you can edit to suit your needs or just use as is. For the most part, the samples will be written in C and will assume that your server is running on a standard Unix platform, but if you are running a server on a Macintosh or Windows PC, you'll find a few tidbits that you can use as well. The main reason Unix is given preferential treatment as the Web server platform of choice is simple: Unix HTTP servers are the most common and most robust HTTP servers available as of this writing. That's not to say that you can't get satisfactory results from a Macintosh or Windows server; in fact there are many intriguing Web sites that are running on both of these platforms. It's just that the Unix side works better at this point in the history of the Web. Well, enough on that, let's get started.

LESSON #1: WHAT IS CGI?

The Common Gateway Interface—or CGI—is a method that lets you access external programs on a Web server and usually send the results to a Web browser. (There are situations in which you want the script to do some processing on your server, but not send data back to the client.)

These programs can be any executable code, script, or program supported by the operating system that runs your server. The CGI code to call an external program can be a shell script, or a batch file, an AppleScript file, a C program, a PASCAL program, compiled BASIC ... literally anything that will run as a stand-alone executable Script on your system. Many CGI developers use shell scripts; others prefer Perl and C. Choose what works best for you! Just to make things simple, this chapter refers to all CGI code files as CGI scripts—or simply as "scripts"—whether they are written in a scripting language or in a compiled or interpreted programming language.

A server executes a CGI script based on a user request from a Web browser, as diagrammed in Figure 15-1. This request can be as simple as selecting a hyperlink that points to an executable item, or it can be a search request using the <ISINDEX> tag, or it can involve clicking the Submit button from within an HTML form. The parameters the script has available to it depend on how it was accessed. There are also many parameters available to scripts via environment variables that are set by the server. You'll get details on all of this in the Writing Your Own Scripts section.

Google/YouTube Exhibit 1005/1105
Page 00408

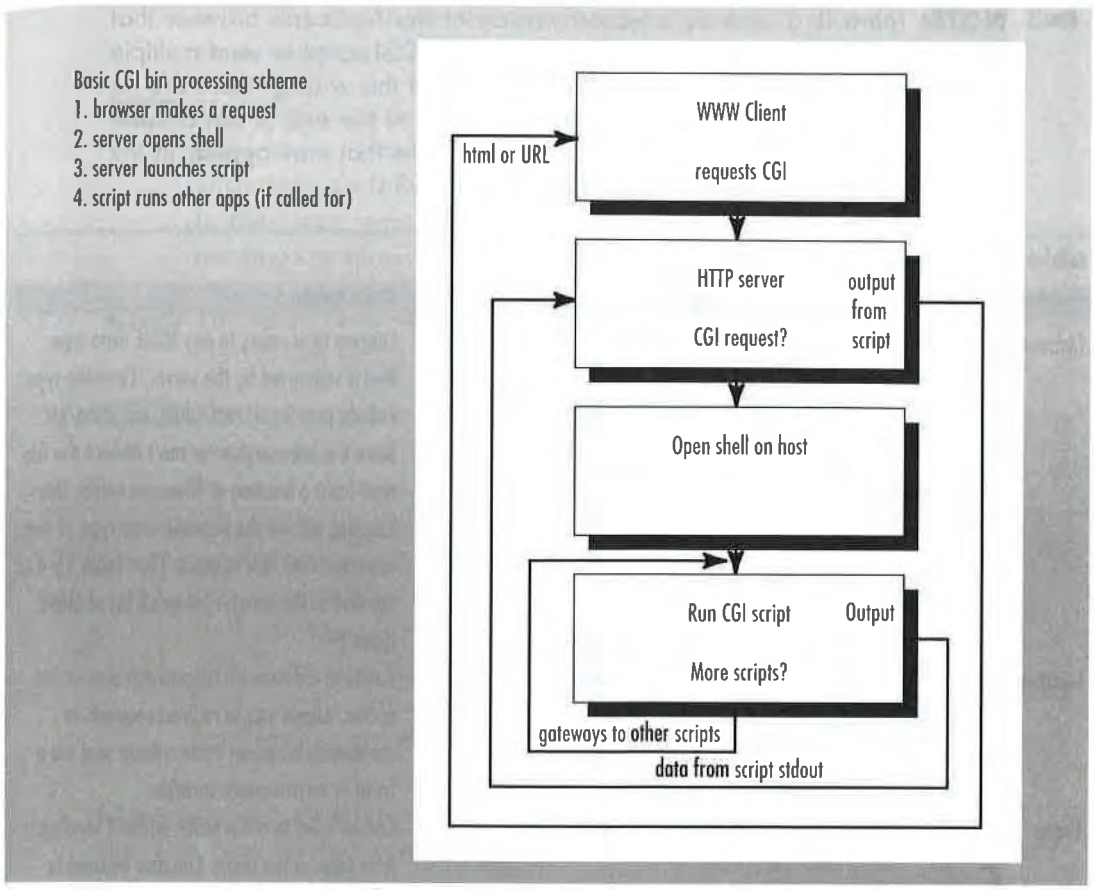


Figure 15-1 Diagram of a CGI session

A CGI script must produce an output header even if no data is to be forwarded to the Web browser. The HTML header must be the first thing that a script sends as output and must be followed by a blank line or carriage return. The header tells the server what kind of data to expect, if any; the server in turn tells the client that invoked the script what to expect. Currently, there are three types of headers. These headers are mutually exclusive—that is, you can't have more than one header for any one request. (See note for exception.) Valid header types are Content-Type, Location, and Status (see Table 15-1).

Google/YouTube Exhibit 1005/1105
Page 00409

NOTE: There is a recently released version of the Netscape browser that supports a multi-part mime type. This will allow a CGI script to send multiple types of data in response to a single request. As of this writing, there are no other browsers that I'm aware of that support this. At the end of this chapter is a brief addendum that describes the new features that may appear in the next version of browsers that support HTML version 3.0.

Table 15-1 Header types

Header Type	Format	Description
Content-Type	Content-Type: xxx/xxx	Content type refers to any MIME data type that is supported by the server. Common types include text/html, text/plain, and data/gif. Since the browser/server can't deduce the file type from a location or filename suffix, this heading will tell the browser what type of data to expect and how to use it. (See Table 15-4 at the end of the chapter for a full list of MIME types.)
Location	Location: /path/doc	Points to a document somewhere else on the server. Allows you to redirect requests to documents based on some criteria sent via a form or environment variable.
Status	Status: nnn XXXX	Can be used to run a script, without sending a new page to the client. Can also be used to send an error message or other information to the client.

What Can I Do with a CGI Script?

A CGI script can do anything allowed on the host system as long as it sends one of the three header types listed in Table 15-1. It can access other programs, open files, read from files, create graphics, dial your modem, call your mother, do database searches, send e-mail, you name it. The only rules are:

HTML The script has to be in a place designated by the server for CGI scripts, or it has to have a special suffix that the server is configured to recognize as a legal CGI script. Most systems store CGI scripts in a directory relative to the root directory of the HTTP server called cgi-bin, which is set up so that only certain trusted users can write

Google/YouTube Exhibit 1005/1105
Page 00410

to it. This avoids the obvious security problems of allowing anonymous remote users to execute anything they want on your system.

HTML The script can take its parameters from the standard input (by standard input, I mean what would normally be typed in at the keyboard), the environment variables, or both. (It is not necessary to take user input at all, the script can simply execute without needing any more information.)

HTML The script must output one of the three standard header types as a normal text string.

HTML The script must be runable by the user that the server is configured to run as. (On a Unix machine every directory, file, and program has a set of permissions attached to it. These permissions specify who can read, change, or execute different files. These permissions are divided into three groups, owner, group, and world. Also, every process must run as some user. There is a special user called "nobody" that is the default user for most Web servers. You must make sure that the user "nobody", or the user that your server is configured to run as, has permission to execute your scripts and read/write to any files that the script may use.)

CGI scripts are used for doing all of the "cool" stuff on the Net. There are sites that have interactive robots you can control with a Web browser, sites that allow you to control cameras and take pictures of remote places, sites that create graphic images on-the-fly, serve maps, open X clients on your machine and send you live video feeds, access huge databases, order submarine sandwiches, and ask questions of the Web's own version of the Magic Eightball, as shown in Figure 15-2. All of this is possible through the use of CGI scripts.

LESSON #2: THE STANDARD CGI SCRIPTS

As you read this, CGI scripts are coming to life all over the world. Some are special purpose, some are useful utilities, some are interesting, and some are just plain silly. Among these scripts, there are a handful that have become standard at most Web sites. If you download NCSA's HTTPD or copy it from the CD-ROM that came with this book, you'll find that it includes two CGI directories, cgi-bin and cgi-src. The cgi-bin directory contains many

Google/YouTube Exhibit 1005/1105
Page 00411

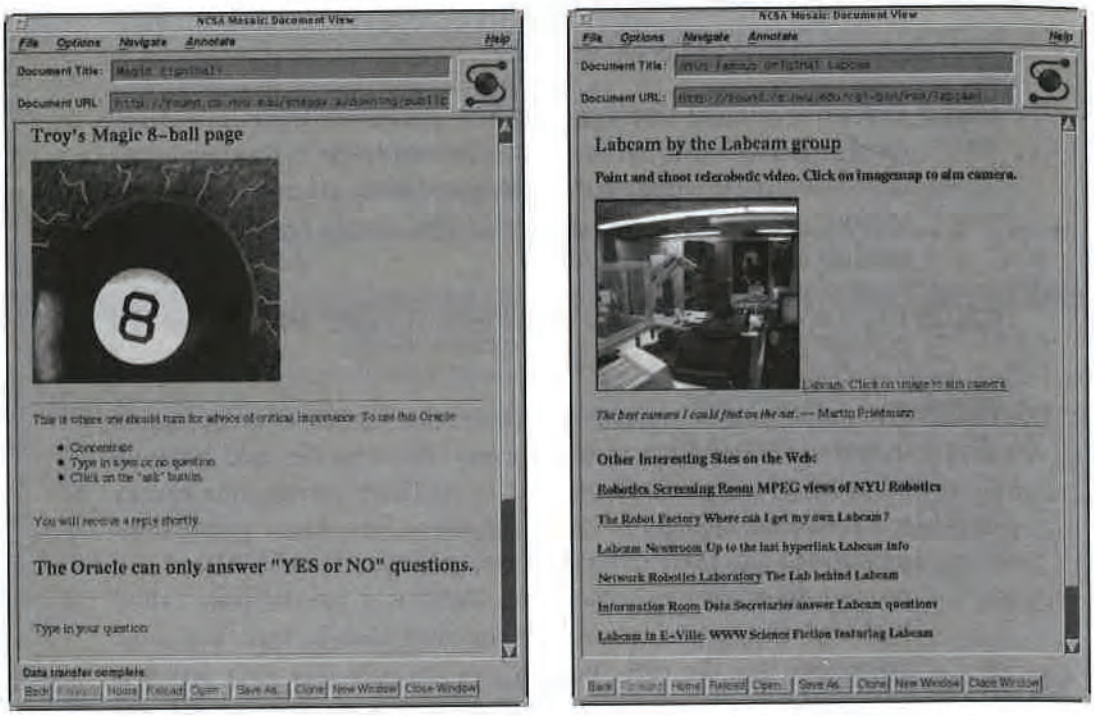


Figure 15-2 Two nifty services, courtesy of CGI scripts

demos and useful CGI scripts, and cgi-src contains the source code for these scripts so you can customize them—or just learn from them.

The CD-ROM scripts include C programs, shell scripts, and Perl scripts that do a few helpful little tasks for you. A few of them are even necessary for your server to have all the utility that you expect of it, like processing image maps. Table 15-2 describes all the scripts in the cgi-bin directory that comes standard with NCSA's HTTPD.

Table 15-2 Standard CGI scripts

Name	Type	Description
archie	shell script	Gateway to an archie server.
calendar	shell script	Gateway to the Unix calendar utility.
date	shell script	Calls the system date and sends it as an HTML doc.
finger	shell script	Gateway to the Unix finger utility.
fortune	shell script	Gateway to the Unix fortune utility.

Google/YouTube Exhibit 1005/1105
Page 00412

Name	Type	Description
imagemap	C program	Handles imagemaps in HTML documents, taking the X and Y coordinates from the user, and forwarding a URL based on a map file created by the map developer.
jj	C program	Processes an order form from a submarine sandwich shop, then opens a pipe to a mailer, and faxes the order out of a fax modem. (A useful example, even if you're not selling sandwiches.)
nph-test-cgi	shell script	Echoes back the names and values of the environment variables. (Good for testing forms, or just figuring out what's going on.)
phf*	C program	Creates a fill-in form interface for a CSO ph database. (Great for looking up names/addresses on ph servers.)
post-query*	C program	Echoes the name/value pairs of a form that uses the POST method.
query*	C program	Echoes the name/value pairs of a form that uses the GET method.
test-cgi*	shell script	Echoes the names and contents of the environment variables.
test-cgi.tcl*	tclsh script	Echoes the names and contents of the environment variables.
uptime*	shell script	Gateway to the Unix uptime command. Will print the time that the system has been running.
wais.pl*	perl script	Provides an <ISINDEX> front end for WAIS searches.

Some of these are C programs, some are TCL scripts, and some are Perl scripts, but they all function in the same way. They get executed by the HTTP server, take their parameters (if any) from standard input or environment variables, and output at least a header when they're done.

As you can see, there are a few utilities here that you may find useful, such as a WAIS or finger gateway, and some that are necessary in some form to allow valuable utilities like imagemap, and still others that are intended simply as learning tools to demonstrate how to write a CGI script in the

Google/YouTube Exhibit 1005/1105
Page 00413

TCL scripting language. They all come standard with the NCSA HTTPD server software, and it's good to know what's there and how it can be used, either as a learning tool or utility. Later on, this chapter will describe many other special-purpose and form-handling scripts.

LESSON #3: HOW TO USE PRE-EXISTING SCRIPTS

OK, I know what a CGI script is, I know what one looks like, I know the names of a few standard scripts, now how do I use them?

The answer to this is simple. You access a CGI script in the same way you access any other URL: create a hyperlink in a document that points to a CGI script, or use the "open URL" option on your browser, or include the URL in the METHOD attribute of a form (as described in Chapter 14). Since most installations require all CGI scripts to be in one protected directory (namely cgi-bin), the following examples all use this convention.

Constructing a CGI URL

A URL that points to a CGI script follows the same conventions as other URLs that point to HTTP servers. It contains a protocol type (HTTP), the name of the server that will execute the script and forward the results, and the name and path of the CGI script to be executed. The simple generic format to create a CGI script URL is:

```
http://machinename/cgi-bin/myprogram
```

In this form, the URL will open an HTTP connection to the "machinename" server, the server will then invoke the "myprogram" script from the standard cgi-bin directory, and will forward the results of the execution of "myprogram" back to the Web browser. There are also ways to include query and path information along with the URL. Query information can be appended to the URL separated by a question mark (?).

```
http://machinename/cgi-bin/myprogram?whoareyou
```

Here the HTTP server sets the environment variable QUERY_STRING to the value "whoareyou" when it executes the "myprogram" script. The script can then access the query data through the environment and make a decision based on the "whoareyou" value that was stored in the QUERY_STRING variable.

To include path information in the URL, simply append the relative path to the URL. For example:

```
http://machinename/cgi-bin/myprogram/people/docs
```


Google/YouTube Exhibit 1005/1105
Page 00414

This will invoke the script and assign the value "people/docs" to the environment variable PATH_INFO. It will also resolve the address from a virtual path to a physical path and store that value into PATH_TRANSLATED. For example, if your server root is set to "/usr/httpd" and "/people/docs" is sent along with the CGI URL, the server will assign PATH_INFO the value "people/docs" and "/usr/httpd/people/docs" is assigned to PATH_TRANSLATED.

The following listing is a sample HTML document that calls a CGI script called test-cgi on a server called myserver.com. If you have a standard HTTPD installation like the one on the CD-ROM that comes with this book, you should be able to replace "myserver.com" with the name of the machine running the server software and use this document.

```
<TITLE>test-cgi</TITLE>
<H1>Test CGI</H1>
<HR>
<A HREF="http://myserver.com/cgi-bin/test_cgi">Click here to run test-
cgi</A><BR>
<HR>
test-cgi should return a virtual HTML document that contains the names
of environment variables and their values on the HTTP server speci-
fied.<BR>
```

Specifying a Script Within a Form

When specifying a script to act on form data submitted from a client, construct the URL in the same way as before, that is:

```
http://machinename/cgi-bin/programname
```

The only difference is where to place it within the HTML document. Specifically, include the script URL in the ACTION attribute of the <FORM> tag.

```
<FORM ACTION="http://machinename/cgi-bin/programname">
```

Later in this chapter, the section headed Handling Form Data will give you the details on creating a CGI script that will process the HTML form.

Specifying a Script from the "Open URL" Interface

Executing a script directly from your Web browser is as simple as selecting the Open URL or Open Location option in your browser and entering the URL for the script. The URL can contain any of the standard URL conventions—see Chapter 1 if you need a review. In particular, you can include

Google/YouTube Exhibit 1005/1105
Page 00415

optional port numbers if you need them, and you have to escape any special characters that may be required to specify the path or filename of the CGI script.

LESSON #4: HOW TO USE A SCRIPT TO ACCESS OTHER APPLICATIONS

Why is it called the Common "Gateway" Interface? Well, the answer is simple: the Common Gateway Interface was originally intended as a "gateway" between WWW clients and other programs that could be run remotely on your server. Many CGI scripts, especially those that access databases, simply execute another application on the server and redirect its output with whatever formatting changes are required to the HTTP server and then to the client that requested the script.

As a simple example, let's take a quick look at the finger script that was mentioned in the standard script table (Table 15-2). Finger is a standard Unix utility that allows you to locate users and/or machines and retrieve information about them. Don't worry if you don't understand the entire script, we'll cover that in the next section.

The finger gateway is written in the Unix shell scripting language, so first we give it a shell script header, and define a constant that points to the actual finger program in the Unix filesystem.

```
#! /bin/sh

FINGER=/usr/ucb/finger
```

Notice that the FINGER constant is assigned the entire path of the program that it will be running. Next, we send the server a standard header. We'll use the Content-Type header and specify the "text/html" MIME type. (For a complete list of MIME types, see Table 15-4 at the end of the chapter.) This will inform the client that we plan on sending it straight ASCII text, and that the text should be interpreted as HTML code. It is important to specify a header of some type, as most servers and browsers will return an error message if they don't get a header. We send the header to the server simply by writing it to the standard output. (In the examples of this chapter, standard output is the same as printing directly to the screen or console. The HTTPD server intercepts and redirects this output as necessary.) An easy way to do this in a shell script is with the "echo" command. Note the blank

Google/YouTube Exhibit 1005/1105
Page 00416

echo line after the header. This is necessary with most servers and should always be included.

```
echo Content-type: text/html
echo
```

Now we are ready to start sending the output from the finger script to the server. Just to make it look a little nicer, we add a <TITLE> tag and a short description of the output as follows.

```
echo <TITLE>Finger Gateway</TITLE>
echo <H1>Finger Gateway</H1>
echo This will finger our HTTP server
echo
```

Now, we execute finger, with a <PRE> tag added just before and a </PRE> right after to make it look a little nicer to the user:

```
echo <PRE>
$FINGER
echo</PRE>
```

Since finger will automatically send its results to the standard output, this text goes to the server and then to the browser as part of an HTML document. This simplified finger gateway looks like this:

```
#! /bin/sh
FINGER = /usr/ucb/finger
echo Content-type: text/html
echo
echo <TITLE>Finger Gateway</TITLE>
echo <H1>Finger Gateway</H1>
echo This will finger our HTTP server
echo
<PRE>
$FINGER
</PRE>
```

Since finger usually works best with parameters, such as user@machine, it's nice to be able to pass along a parameter supplied by the user. The full finger gateway uses the <ISINDEX> tag to get a username and machine-name from the user, and passes these along to the Unix finger utility. A listing of the complete finger gateway is listed below. Some parts may be unfamiliar, but these will be discussed in the Writing Your Own Scripts section.

```
#! /bin/sh
#This script comes standard with NCSA's HTTPD
FINGER=/usr/ucb/finger
echo Content-type: text/html
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00417

Continued from previous page

```
echo
if [ -x $FINGER ]; then
        if [ $# = 0 ]; then
                cat << EOM
<TITLE>Finger Gateway</TITLE>
<H1>Finger Gateway</H1>
<ISINDEX>
This is a gateway to "finger". Type a user@host combination in your
browser's search dialog.<P>
EOM
        else
                echo \<PRE\>
                $FINGER "$*"
        fi
else
        echo Cannot find finger on this system.
fi
```

WARNING! Remember to make your shell scripts executable. In Unix this means that you must type:

```
chmod o+x scriptname
```

for any new shell script you create.

It should start becoming clear how simple and powerful a gateway script can be. A script can point to any executable file on your server and execute it. All data sent to the standard output–either your script or the file(s) it executes–will be forwarded to the client and interpreted as the MIME type specified in the Content-Type header. This is a lot of power and should be used with caution. You don't want users imposing potentially (or deliberately) destructive scripts on your server, so it is usually a good idea, on a shared system, to allow only a few trusted users to create CGI scripts.

There are a few mechanisms in place to help protect you. One is the ability to require CGI scripts to be in a specific directory, and the other is the ability to require CGI scripts to have a specific suffix. Either method prevents anonymous users from being able to write URLs that point to your server and run whatever they want (Say, rm -r * for example. Not a pleasant thought.) Don't worry! The risk potential is there, but if you install your server with some thought, you should be able to avoid such mishaps. If you are concerned about security, read the security section in Chapter 21.

LESSON #5: WRITING YOUR OWN SCRIPTS

OK, enough talk, let's see some action. We're going to try the learning-by-example method here, so let's just get a few things out of the way first. In order to run CGI scripts, make sure the following infrastructure is in place:

Google/YouTube Exhibit 1005/1105
Page 00418

HTML You have an HTTP server installed at your site.

HTML The HTTP server has been configured to allow CGI scripts.

HTML You, or someone you know, has write permission in the cgi-bin directory on this server, unless the server has been configured to allow CGI scripts elsewhere.

That in mind, writing a CGI script is a six-step process:

1. Write the script and compile it if necessary. (Obviously, you don't compile a shell script.)
2. Have the script moved into the cgi-bin directory (or equivalent).
3. Make sure the script is executable. (The Unix command is *chmod o+x scriptname*.)
4. Write a reference URL or form to access the script.
5. Debug the script.
6. Publish the script. (Tell your audience about it, or create links to it.)

Writing a Simple Script

Let's start with a simple script: an interactive <ISINDEX> form that will ask the user to input his or her name, and then echo back a short greeting to the user's browser. This example assumes that you are using the NCSA HTTPD server software from the CD-ROM that came with the book.

We will use the Unix shell scripting language to write it. To start, using your favorite text editor, create the following file:

```
echoname.sh
```

Since this is a shell script, we will start it with a standard shell script header. The first line is:

```
#! /bin/sh
```

Now, to avoid having problems interpreting the data, or getting error messages for not being specific, we add the HTML header. In this case, we are returning text that we want interpreted as HTML code. The header for this type of data is just the MIME type for HTML code. Add the following to your file to print the header information:

```
echo Content-Type: text/html
echo
```


Google/YouTube Exhibit 1005/1105
Page 00419

The extra "echo" is necessary, as most installations require a blank line after the header. Next, we want to create the HTML code that is sent to the browser. We will share some code, and the rest will be unique depending on whether there was user input or not. First we create the common HTML. Notice that in a shell script, the greater-than and less-than brackets (< >) are reserved symbols and must be escaped with backslashes (\). So if you want to print a less-than symbol using echo, you would use "echo \<"—"echo <" won't work by itself. You can avoid messing with backslashes by enclosing the entire string in double-quotes. With that in mind, add the following lines to your file:

```
echo "<TITLE>Echoname example</TITLE>"
echo "<ISINDEX>"
echo "<H1>CGI script example</H1>"
echo Any name typed into the query window will be echoed to the
screen.
echo "<HR>"
```

Now we want to create a fork—one side allows the user to input a name, and the other displays a message if the user types in a name. In this example, since we are using the <ISINDEX> tag, we can assume that the command line parameter count is greater than zero if the user entered a value, and zero if not. The following lines will check for a value on the command line and print a prompt message if there were no parameters.

```
if [ $# = 0] ; then
        echo Please enter your name in the query window.\<BR\>
else
        echo Hello $*, Welcome to our server.
fi
```

Now, if this script is called without parameters, it will print the message "Please enter your name in the query window." When called with parameters, it will print the message "Hello [parameters], Welcome to our server." The entire script looks like this:

```
#! /bin/sh
echo Content-Type: text/html
echo
echo "<TITLE>Echoname example</TITLE>"
echo "<ISINDEX>"
echo "<H1>CGI script example</H1>"
echo Any name typed into the query window will be echoed to the
screen.
echo "<HR>"
if [ $# = 0 ]; then
        echo Please enter your name in the query window.\<BR\>
else
        echo Hello $*, Welcome to our server.\<BR\>
fi
```


Google/YouTube Exhibit 1005/1105
Page 00420

NOTE: The <ISINDEX> tag uses the GET method to pass data from the browser to the calling script. This means that the encoded input data is stored in the QUERY_STRING environment variable. But the <ISINDEX> query will also list the unencoded values on the command line of the calling script.

Before we can test the script, we need to make sure it is executable. At the Unix command prompt, type:

```
chmod +x echoname.sh
```

This will make the shell script executable, a necessity if you want to be able to run this script. Now to test it, type:

```
echoname.sh
```

This should produce the results:

```
Content-Type: text/html

<TITLE>Echoname example</TITLE>
<ISINDEX>
<H1>CGI script example</H1>
Any name typed into the query window will be echoed to the screen.
<HR>
Please enter your name in the query window. <BR>
```

Now test the script with a command line argument. Try the following:

```
echoname.sh Troy
```

The result should be:

```
Content-Type: text/html

<TITLE>Echoname example</TITLE>
<ISINDEX>
<H1>CGI script example</H1>
Any name typed into the query window will be echoed to the screen.
<HR>
Hello Troy, Welcome to our server.
```

We are now ready to place the file into the correct directory and try it out with a Web browser. Copy the file into the appropriate directory for CGI scripts on your server. If you're using NCSA HTTPD from the CD-ROM, this is the cgi-bin directory. In any case, the command will be something like:

```
cp echoname.sh /usr/httpd/cgi-bin
```

Now, let's try it out with a browser. Take your favorite Web browser and open the following URL, substituting the name of your server and the exact path to your CGI directory as necessary:

```
http://yourserver/cgi-bin/echoname.sh
```


Google/YouTube Exhibit 1005/1105
Page 00421

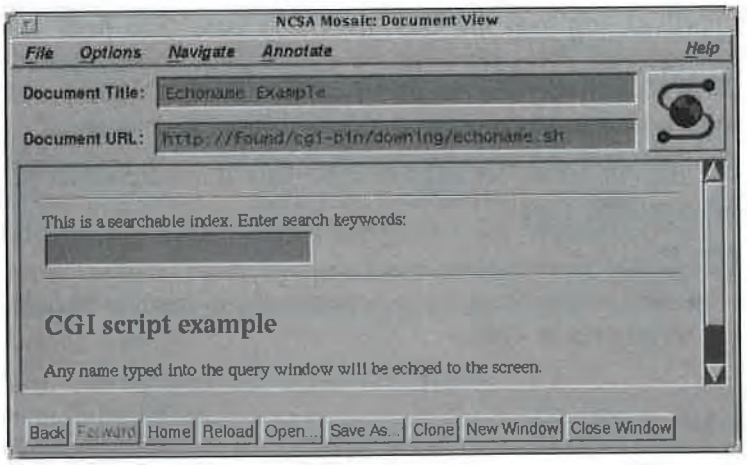


Figure 15-3 echoname.sh with no arguments

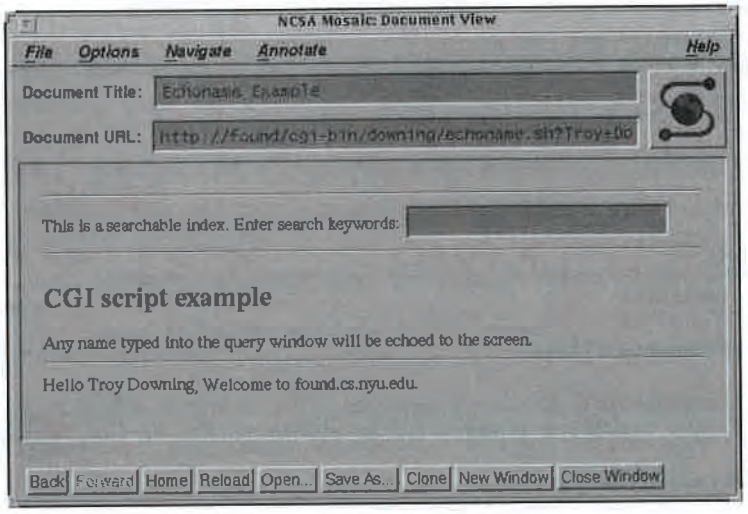


Figure 15-4 echoname.sh with "Troy Downing" as the argument

Opening this URL should cause your HTTP server to execute the script "echoname.sh" and send back an HTML page with a query window as in Figure 15-3.

Now try typing a name into the query window. The resulting screen should look something like Figure 15-4.

Congratulations! You've just created your first CGI script. Doesn't do a whole lot, but it shows how simple script writing can be. This script could have been created just as easily in another scripting language or a compiled language such as C or PASCAL. The next example will show how to use

Google/YouTube Exhibit 1005/1105
Page 00422

environment variables to get information about the user and the user's environment.

Using Environment Variables in Scripts

Whenever a server launches a CGI script, a number of environment variables are set with information about the data being sent, the client software, the client machine, even the username in some authentication schemes. See Table 15-3 for a list of environment variables set on the NCSA HTTPD server. As a simple exercise, we are going to add a few lines of code to the previous echoname.sh script to make use of some environment variables. The only lines we are going to change are the few at the end that print the "hello" message. The two variables we will use to demonstrate this are SERVER_NAME and REMOTE_HOST. SERVER_NAME is set to the name of the machine that is running the HTTP server and REMOTE_HOST is the name of the machine that is making the HTTP request. Let's make the following changes to echoname.sh:

```
#!/bin/sh
echo Content-Type: text/html
echo
echo "<TITLE>Echoname example</TITLE>"
echo "<ISINDEX>"
echo "<H1>CGI script example</H1>"
echo Any name typed into the query window will be echoed to the
screen.
echo "<HR>"
if [ $# = 0 ]; then
        echo "Please enter your name in the query window.<BR>"
else
        echo Hello $* from $REMOTE_HOST, Welcome to \
        $SERVER_NAME.\<BR\>
fi
```

Notice the addition to the last "echo" line. We've added the two environment variables to our greeting. Now execution of this file should result in a reply string that looks something like:

```
Hello Troy from play.cs.nyu.edu. Welcome to www.nyu.edu.
```

You may or may not want to use the environment variables in this way. Environment variables are particularly useful when processing forms. In many cases, you need them for retrieving data from forms using the GET method, and for determining the length of the data block when using the POST method. This will all be explained in greater detail in the section on form handling coming up next. Before going on to forms, let's take a look at a few more scripts.

Google/YouTube Exhibit 1005/1105
Page 00423

Table 15-3 Environment Variables

Variable Name	Description
SERVER_SOFTWARE	The name and version number of the server software that is serving the request, and running the CGI script. Format: name/version.
SERVER_NAME	The server's hostname, alias, or IP address depending on the particular installation.
GATEWAY_INTERFACE	Revision number of the gateway interface. Format: CGI/revision #.
SERVER_PROTOCOL	The protocol name and revision of the protocol that the request came in with. Format: protocol/revision.
SERVER_PORT	The port number that the server is accepting requests through. (Usually port 80.)
REQUEST_METHOD	The method of the request. Normally POST or GET.
PATH_INFO	The path information that came along with the request. Normally, this information was appended to the end of the URL that called the CGI script.
PATH_TRANSLATED	The physical mapping that is derived from the virtual path supplied in PATH_INFO.
SCRIPT_NAME	The path and file name of the script.
QUERY_STRING	The value of a query URL or a form that was sent using the GET method is stored here. The QUERY_STRING is url-encoded, unless the query was invoked with the <ISINDEX> tag, then the "name" of the field is omitted and only the value is assigned to QUERY_STRING variable. In <ISINDEX> calls, the unencoded value will also be passed along to the script as command line parameters.
REMOTE_HOST	The host name of the machine making the request. Either the DNS name or alias.
REMOTE_ADDR	The IP address of the REMOTE_HOST.
AUTH_TYPE	The authentication method used to validate users for protected scripts.
REMOTE_USER	The user name making the request. This value is only set if user authentication has been used.
REMOTE_IDENT	The user ID for a remote user in some authentication schemes.
CONTENT_TYPE	The MIME type of the data being served.
CONTENT_LENGTH	The number of bytes of content being sent by the client.
HTTP_ACCEPT	The MIME types that the client will accept. Format type/type, type/type,...
HTTP_USER_AGENT	The browser that the client is using.

As an exercise and a utility to see what your environment variables are being set to, we will write a short shell script that simply returns the values of all the main environment variables.

```
#!/bin/sh
#simple script to return the values of environment variables.
echo Content-Type: text/html
echo
#simple header info
echo "<TITLE>env_vars.sh</TITLE>"
echo "<H1>env_vars.sh</H1>"
```


Google/YouTube Exhibit 1005/1105
Page 00424

```
echo "Below are the values of environment variables that were set"
echo "when this script was launched.<b><HR><LISTING>"
#were there any command-line arguments?
echo number of args: $#
echo value of args: $*
echo
#now the variables
echo SERVER_SOFTWARE:          $SERVER_SOFTWARE
echo SERVER_NAME:              $SERVER_NAME
echo GATEWAY_INTERFACE:        $GATEWAY_INTERFACE
echo SERVER_PROTOCOL:          $SERVER_PROTOCOL
echo SERVER_PORT:              $SERVER_PORT
echo REQUEST_METHOD:           $REQUEST_METHOD
echo PATH_INFO:                $PATH_INFO
echo PATH_TRANSLATED:          $PATH_TRANSLATED
echo SCRIPT_NAME:              $SCRIPT_NAME
echo QUERY_STRING:             $QUERY_STRING
echo REMOTE_HOST:              $REMOTE_HOST
echo REMOTE_ADDR:              $REMOTE_ADDR
echo AUTH_TYPE:                $AUTH_TYPE
echo REMOTE_USER:              $REMOTE_USER
echo REMOTE_IDENT:             $REMOTE_IDENT
echo CONTENT_TYPE:             $CONTENT_TYPE
echo CONTENT_LENGTH:           $CONTENT_LENGTH
echo HTTP_ACCEPT:              $HTTP_ACCEPT
echo HTTP_USER_AGENT:          $HTTP_USER_AGENT
```

Be sure to make this script executable and to put it into the correct directory for CGI scripts. If you include this script as the action for a form, or just call the script directly from your favorite browser, it will list the contents of the environment variables that we listed in the script. Typically, the results will look something like Figure 15-5.

Location and Status Headers

The Content-Type header we've been discussing tells the browser to expect a stream of data of a certain type, but sometimes you don't want to create a data stream at all. If you want your script to simply redirect clients to a different location based on the machine they are connecting from or the browser they are using, use the Location header. You can also use a Location header to point the browser to a different file somewhere on the server. The format is simple:

```
Location: path/newfile
```

Like the Content-Type header, the Location header requires a blank line after it. To use this header in a shell script, it would look something like:

```
echo Location: ../downing/funstuff.html
echo
```


Google/YouTube Exhibit 1005/1105
Page 00425

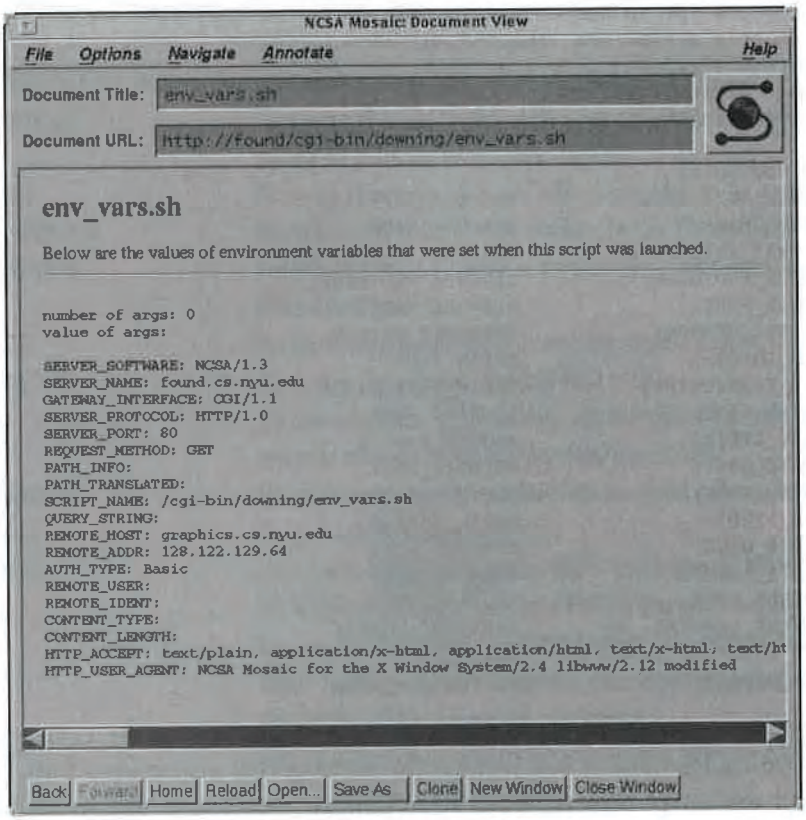


Figure 15-5 env_vars.sh results

NOTE: You cannot mix header types. Every header must be either Content-Type, or Location, or Status.

The following script will redirect a request based on the browser making it. A Netscape browser will get a file formatted for Netscape, other browsers will get a default page.

```
#!/bin/sh
#This will send the location of a file based on the
#client browser
FILENAME="default.html"
#default.html is the standard HTML file we want to serve
#if the user is using Netscape, we will redirect to
#nsversion.html
if [ "$HTTP_USER_AGENT" = "Mozilla"]; then
```


Google/YouTube Exhibit 1005/1105
Page 00426

```
        FILENAME="nsversion.html"
fi
        echo Location: ../htdocs/$FILENAME
        echo
```

If you want to run a CGI script without having any change appear on the user's browser, use the Status header. If the script returns a Status header with the status number set to 204 and the string "No Response" attached, the calling browser will simply stay on the page that the request was made from. In other words, the browser does nothing, even though the server ran a remote script based on the browser's request. We can take care of some task, say add a line to a database, without changing the user's current page. The following script will add the machine name that the request came from, and the name of the browser used, to a database in the "logs" directory relative to the cgi-bin directory.

```
#! /bin/sh
#This will add the machine name and browser
#name of a client to a database
LOGPATH="../logs/browser.dat"
echo $REMOTE_HOST $HTTP_USER_AGENT >> $LOGPATH
#now send the status to the browser
echo Status: 204 No Response
echo
```

Assuming that you have created a file called /logs/browser.dat, this script will add the remote host name and the browser name to this file and terminate, sending a status code back to the client. The client will stay on the page where the call came from. There are a number of status codes that are sent from a server to a browser. Most of them aren't very useful in cgi scripts but are used to tell the browser that a file was not found or that the user doesn't have permission to access a certain file. I'm sure you've all seen the "404 Not Found" error message—this was status number 404.

Security with CGI Scripts

BEWARE! Watch out for characters that have special meanings to the shell, such as %.<,>... A client can enter these characters into input fields and sometimes compromise your system if you don't handle them carefully. Any user-supplied data that is used as a command line argument can take advantage of this problem. An easy, but not foolproof way to handle some of these problems is to include the command line parameters in double quotes so that any special characters will be treated as literals rather than shell directives. See Chapter 21 for more information on server security.

Google/YouTube Exhibit 1005/1105
Page 00427

For example, if your ACTION server is *http://www.smartpants.edu/cgi-bin/survey* and your form asks for a name and phone number, the URL might look like this:

```
http://www.smartpants.edu/cgibin/survey?lname=smith&fname=john&phone=1
23-555-1234
```

Since different operating systems have different limits for the size of the URL, the URL is often truncated if you send a lot of data.

The Web server on the receiving side assigns the string passed by the form to an environmental variable called QUERY_STRING. The CGI script processing the form can then access the data by reading the contents of QUERY_STRING and parsing it into its component parts. See the next chapter for details.

POST

The POST method sends a special data stream to the server (specified by the ACTION attribute). The POST stream looks similar to the GET query. A typical POST may take a form similar to the following:

```
lname=smith&fname=john&phone=123-555-1234
```

The POST method allows you to send an arbitrary amount of data. There is no need to worry about it being too long and getting cut off. The string goes to the receiving URL as a standard *input stream* to the specified CGI script. The length of the string is stored in an environment variable called CONTENT_LENGTH, so the CGI script can access the names and values by reading the number of characters stored in CONTENT_LENGTH from the standard input stream. The CGI program can then break up the string into its component parts and process the data. The next chapter gets into all this nitty-gritty.

THE FUTURE OF FORMS

Right now, forms are somewhat limited. It would be nice to have calculated fields, so you could turn a Web page into an instant spreadsheet. The values of some fields should also be able to affect the defaults or status of others. If you wanted to have the server control these calculated fields, forms would take way too long to process. As a result, people are developing a special *application program interface* (*API*) to allow live fill-in form processing to work hand-in-hand with all Web browsers.

Google/YouTube Exhibit 1005/1105
Page 00428

Most HTTP servers and clients have certain security features built in, but you may occasionally want to try protecting a document by having a CGI script ask the user for a password of some sort. The following script will give very rudimentary security to a script; it's listed here as an example of how you might implement such a scheme, even though it's not necessarily a completely secure solution. It will print a message prompting the user to input a password. Since this is a single field of input, we will use the <ISINDEX> tag.

NOTE: If you wanted to add a more secure password field in a form, it would make sense to use <FORM> tags instead of the <ISINDEX> tag and use the <INPUT TYPE="password"> tag to prevent characters from being echoed to the screen.

This script will return a "failure" page if it receives an incorrect password. If it gets the correct password, it will redirect the browser to another location.

```
#!/bin/sh
#simple password script
PASSWORD=Schmoo
PROTECTEDFILE=/usr/me/securefile.html
if [ $# = 0 ]
then
        cat << EOM
                Content-Type: text/html
                <TITLE>Password script</TITLE>
                <ISINDEX>
                <H1>This page is protected. Enter password</H1>
EOM
elif [ "$#" = "$PASSWORD" ]
then
        echo Location: $PROTECTEDFILE
        echo
else
        cat << EOM2
                Content-Type: text/html

                <TITLE> FAIL!</TITLE>
                <ISINDEX>
                <H1>Password failed! Try again.</H1>
EOM2
fi
```

Remember—passwd.sh is meant as a demonstration to base other schemes on; it's far from the most secure way to protect a page or server. If you are interested in security, read the security section in Chapter 21.

Google/YouTube Exhibit 1005/1105
Page 00429

LESSON #6: HANDLING FORM DATA

And now, (drum roll please), the moment you have all been waiting for... Form Handling! In Chapter 14, we learned all about one half of the form scheme: how to write the HTML code that describes a form interface and how the form sends its data. Unfortunately, you can't do much with a form without having some sort of program that can accept the data that is passed by a form and do something with it. To clear the air about using form data, there is some good news and some bad news. The bad news: the form data is sent in an encoded data block that can be a pain to decode into its component parts. The good news, this is such a common task among CGI scripters that people have already written a number of form-decoding utilities. A useful little collection of C functions comes with NCSA's HTTPD; you can just plug these functions into your C programs, and voilà! the task is done. For those of you who are not C programmers, there are also plenty of utilities that can be used with shell scripts, TCL scripts, and most of the common scripting and programming languages.

What Does Form Data Look Like?

There are two methods that a form can use to pass data to a script, GET and POST. As a quick guideline, use POST whenever possible, and use GET only for indices and single-parameter forms.

When a form sends its data using the GET method, the data is encoded and stored in the environment variable QUERY_STRING. With the POST method, data is sent along through the standard in stream of the script. (By standard in, I mean what would normally be typed in from the keyboard. The HTTPD server redirects that data to the script as if it were being typed in.) In either case, the string is URL encoded. All variables and their values are paired together with equals (=) signs, then all of the name/value pairs are concatenated and separated with ampersands (&). The spaces are replaced with plus (+) signs, and the special characters are escaped. (Backslashes don't work here as they did in shell scripts); in this context, "escaped" means the character is represented by a percent sign (%) followed by the hexadecimal ASCII representation for that character.)

NOTE: Some definitions may be useful here. *American Standard Code for Information Interchange (ASCII)* is a code assigning unique numbers to the standard printable and control characters. This code can be read by virtually

Google/YouTube Exhibit 1005/1105
Page 00430

any computer in operation today. *Hexadecimal,* or base 16, is a numbering system using 16 as the base instead of 10. Numbers from 10 to 15 are represented by the first five letters of the alphabet.)

Encoding example: If a form had the following text input fields in its description: <INPUT TYPE="text" NAME="VAR1"> and <INPUT TYPE="text" NAME="VAR2">, and the strings typed into these text fields were "Troy Downing" and "Boo{TAB}Radley", the resulting encoded string would look like:

```
VAR1=Troy+Downing&VAR2=Boo%09Radley
```

To interpret the contents of the string, you would want to parse it into name/value pairs, replace the "+" with a space, and replace %09 with the {Tab} character. Simple enough? The following code has a number of C routines to do just that. It's based on the NCSA distribution version, but has a few comments added to the code and a header file to go with it.

```
/* util.c standard utility functions for parsing encoded
  strings, and other related CGI tasks. */
#include <stdio.h>
#define LF 10
#define CR 13
/* getword separates two words separated by the "stop"
  character */
void getword(char *word, char *line, char stop) {
   int x = 0,y;

   for(x=0;((line[x]) && (line[x] != stop));x++)
       word[x] = line[x];

   word[x] = '\0';
   if(line[x]) ++x;
   y=0;

   while(line[y++] = line[x++]);
}
char *makeword(char *line, char stop) {
   int x = 0,y;
   char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));

   for(x=0;((line[x]) && (line[x] != stop));x++)
       word[x] = line[x];

   word[x] = '\0';
   if(line[x]) ++x;
   y=0;

   while(line[y++] = line[x++]);
   return word;
```


Google/YouTube Exhibit 1005/1105
Page 00431

```
}
char *fmakeword(FILE *f, char stop, int *cl) {
   int wsize;
   char *word;
   int ll;

   wsize = 102400;
   ll=0;
   word = (char *) malloc(sizeof(char) * (wsize + 1));

   while(1) {
       word[ll] = (char)fgetc(f);
       if(ll==wsize) {
           word[ll+1] = '\0';
           wsize+=102400;
           word = (char *)realloc(word,sizeof(char)*(wsize+1));
       }
       --(*cl);
       if((word[ll] == stop) || (feof(f)) || (!(*cl))) {
           if(word[ll] != stop) ll++;
           word[ll] = '\0';
           return word;
       }
       ++ll;
   }
}
char x2c(char *what) {
   register char digit;

   digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] -
'0'));
   digit *= 16;
   digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] -
'0'));
   return(digit);
}
void unescape_url(char *url) {
   register int x,y;

   for(x=0,y=0;url[y];++x,++y) {
       if((url[x] = url[y]) == '%') {
           url[x] = x2c(&url[y+1]);
           y+=2;
       }
   }
   url[x] = '\0';
}
/* plustospace turns the '+' characters into spaces
  in an encoded string */
void plustospace(char *str) {
   register int x;

   for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00432

Continued from previous page

```
}
/* plusto_ is similar to plustospace, but replaces the '+'
  with an underscore character */
void plusto_(char *str) {
   register int x;

   for(x=0;str[x];x++) if(str[x] == '+') str[x] = '_';
}

int rind(char *s, char c) {
   register int x;
   for(x=strlen(s) - 1;x != -1; x--)
       if(s[x] == c) return x;
   return -1;
}
int getline(char *s, int n, FILE *f) {
   register int i=0;

   while(1) {
       s[i] = (char)fgetc(f);

       if(s[i] == CR)
           s[i] = fgetc(f);

       if((s[i] == 0x4) || (s[i] == LF) || (i == (n-1))) {
           s[i] = '\0';
           return (feof(f) ? 1 : 0);
       }
       ++i;
   }
}
void send_fd(FILE *f, FILE *fd)
{
   int num_chars=0;
   char c;

   while (1) {
       c = fgetc(f);
       if(feof(f))
           return;
       fputc(c,fd);
   }
}
/* util.h simple header file with the prototypes used in util.c. */
#ifndef __util.h
#define __util.h
extern void getword(char *word, char *line, char stop);
extern char *makeword(char *line, char stop);
extern char *fmakeword(FILE *f, char stop, int *cl);
extern char x2c(char *what);
extern void unescape_url(char *url);
extern void plustospace(char *str);
extern void plusto_(char *str);
```


Google/YouTube Exhibit 1005/1105
Page 00433

```
extern int rind(char *s, char c);
extern int getline(char *s, int n, FILE *f);
extern void send_fd(FILE *f, FILE *fd);
#endif
```

Right now, you're probably thinking, "Wow, what a mess! What am I supposed to do with all of that stuff?" Well, don't worry, you don't have to look at the util.c code again. All we will do is compile it into an object file once, then link it to whatever programs we write that use its functions. So, let's get that out of the way. You should create a directory somewhere to contain the source code for your CGI scripts. Move the util.c and util.h files into this directory. Now, using your favorite C compiler, compile the util.c file into an object file called util.o. If you use the gcc compiler, all you have to type to create this object file is

```
gcc -c util.c -o util.o
```

This would compile the code and save it in the same directory as util.c. OK, that's taken care of. Now, the best way to show how to use these functions is to demonstrate them in a CGI script example. The following program will create a Magic Eightball game on your server. For those of you who have never heard of the Magic Eightball, it is a black plastic ball full of blue liquid, with an icosahedron floating inside. The bottom of the ball has a window, in which one of the sides of the icosahedron is visible. There are 20 different answers to "Yes" or "No" questions written on the sides of the icosahedron. The user asks the Eightball a question, shakes the ball, and then turns it upside down to read its response.

In this version of the Eightball, the user types a question into a form on their Web browser. A click of the Submit button encodes the question and sends it to a server. The server randomly picks one of the 20 replies that would normally be written on the icosahedron, and returns an HTML document with the answer. This CGI script also adds the question and answer to a log file so that users can select the "log" page and look at all of the extremely useful advice that the Eightball has given.

Forms generally have two parts, an HTML part and a CGI script part. This example lists the CGI script first since that's what we're working on here, then it gives a few notes on how to compile the script, and finally lists the corresponding HTML form description.

```
#include <stdio.h>
#include "util.h"
#define CHOICES         20
#define LOG             "/usr/httpd/logs/eightball_log.html"
#define LF              10
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00434

Continued from previous page

```
int main(void)
{

        int cont_len;   /* number of bytes to read from stdin*/
        unsigned char rn; /* random number */
        FILE *questions; /* pointer to the log file */
        char *entry;     /* value of the data string */
          /* 20 answers of the icosahedron */
        static char *message[]={
          "Yes",
          "No",
          "Maybe",
          "My reply is YES",
          "Reply Hazy try again",
          "Concentrate and ask again",
          "Definitely",
          "Signs point to YES",
          "Ask again later",
          "Without a doubt",
          "It is certain",
          "Outlook not so good",
          "My reply is NO!",
          "Don't count on it",
          "Outlook good",
          "Most likely",
          "VERY doubtful",
          "My sources say no",
          "You may rely on it",
          "It is decidedly so"

        };
          /* getstring length */
        cont_len = atoi(getenv("CONTENT_LENGTH"));
          /* store data string in entry */
          entry = fmakeword(stdin, '&', &cont_len);
          /* replace +s with spaces */
        plustospace(entry);
        /* unescape any special character
        unescape_url(entry);
          /* store value of name/value pair in entry */
        makeword(entry,'=');
          /* print standard MIME header */
        printf("Content-type: text/html%c%c",LF,LF);

        if (!(questions=fopen(LOG,"a")))
                printf(stderr,"unable to open log file");
          /* seed the random generator */
        srand(getpid());

        rn = rand()%20; /* take a random number between 0..19*/
          /* add the question and answer to the log */
        fprintf(questions, "<pre>%s </pre>",entry);
```


Google/YouTube Exhibit 1005/1105
Page 00435

```
        fprintf(questions, "<B> %s</B><hr>\n",message[rn]);

        fclose(questions);
           /* return the answer page to the user */
        printf("<title>Magic 8-ball reply</title>%c",LF);
        printf("The Magic Eight Ball has thought long");
        printf(" and hard about your question and has come");
        printf(" to the following answer:<p>%c<hr>%c",LF,LF);
        printf("<H1>%s</H1>%c<hr>",message[rn],LF);
        printf("<a
href=\"http://found.cs.nyu.edu/shaggy.a/downing/public-html/e
ightball.html\">Return to 8-ball</a><p>");
        printf("<a
href=\"http://found.cs.nyu.edu/shaggy.a/downing/public-html/i
ndex.html\">Troy's page</a><p><hr>%c",LF);
        printf("If you find public/anonymous postings in a log format
%c",LF);
        printf("offensive, or embarrassing, ");
        printf("then please do not read the 8-ball log <p>%c",LF);
        printf("<a href=\"logs/eightball_log.html\"> Read 8-ball
log</a><p>");
        printf("%c<hr>",LF);
}
```

Note that you will have to change the path names and filenames to match the ones your system uses. For example, if you put the log file in a different location, you will have to change the HREF that points to it in the bottom of the file.

To install this file, compile it and link it with the util.o object file that we created earlier in this section. If you're following the example exactly, type:

```
gcc eightball.c util.o -o /usr/httpd/cgi-bin/eightball
```

Note the path to the cgi-bin directory. This eliminates the step of moving the compiled program into the correct directory. If you compile your version locally, remember to either move it into the correct directory, or ask your system administrator to move it for you if you don't have write privileges to make changes there.

Now, let's take a look at the form that calls this script. Remember that you will have to change the machine name and path to match your own installation.

```
<TITLE>Magic Eightball</TITLE>
<H1>Troy's Magic 8-ball page</H1>
<IMG SRC="/shaggy.a/downing/public-html/8-ball.gif"><P>
<HR>
This is where one should turn for advice of critical
importance. To use this Oracle:
<UL>
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00436

Continued from previous page

```
<LI>Concentrate
<LI>Type in a yes or no question
<LI>Click on the "ask" button.
</UL><BR>
You will receive a reply shortly.<BR>
<HR>
<H1> The Oracle can only answer "YES or NO" questions.</H1>
<HR>
<FORM ACTION="http://found.cs.nyu.edu/cgi-bin/downing/eightball"
METHOD="POST">
Type in your question:<P>
<HR>
<TEXTAREA NAME="question" ROWS=2 COLS=60></TEXTAREA>
<HR>
<INPUT TYPE="submit" VALUE="Ask 8-ball"><INPUT TYPE="reset"
VALUE="Clear Entry">
</FORM>
<HR>
<H5>I can't be held responsible for bad advice given
by this oracle</H5>
<H4>Accept no imitations! This is the Original WWW 8Ball and not a
cheap imitation!</H4>
<A href=/logs/eightball_log.html>Read Log</A>
```

To make this work, you must make sure that all of the hrefs are pointing to the path/names of the files you created on your system. The paths and filenames listed here are only examples. To see the Eightball in action in its original home, point your Web browser at:

```
http://found.cs.nyu.edu/downing/eightball.html
```

The Eightball example only deals with a single variable that contains the question string. What happens when we need to deal with several name/value pairs? Well, the following example works with the basic "feedback" page form that was described in Chapter 14, which allows a user to submit comments via their browser. This version adds the comments to a database, but this could easily be changed to a mailer that would mail the results to the form's owner.

The technique used for dealing with multiple name/value fields in this example is not necessarily the best. It assumes that the name/value pairs will arrive in a specific order, trusting that the form will control the order and eliminate the need to bother checking the names of the variables. To take a simple example, if a form has the following code bits in it:

```
<INPUT TYPE="text" NAME="Foo">
<INPUT TYPE="text" NAME="bar>
```

The CGI script can assume that the encoded string will be something like:

```
Foo=foovalue&bar=barvalue
```


Google/YouTube Exhibit 1005/1105
Page 00437

If you separate the values and index them in an array of strings, you can assume that string 0 will contain the value of Foo and string 1 will contain the value of bar. This method falls apart if the input fields are reordered in the form description, but let's assume, for now, that they are not.

Here is the description of the database handler that goes with the form description in Chapter 14. It will take the form data, return an HTML page to the browser confirming receipt of the data, write the data to a database, and send e-mail informing the author that new data has been added.

```
#include <stdio.h>
#include <stdlib.h>
#include "util.h"
#define MAILER  "/usr/ucb/mail"
#define LOGFILE "/usr/pub/feedback.log"
#define LF      10
#define MAX_ENT 1000                /*  max number of variables
                                      that can be used */
#define MY_EMAIL     "downing@nyu.edu"
/* header index. used to index the struct array containing the
  variables passed from mosaic */
#define FIRST   0
#define SECOND  1
#define EMAIL   2
#define LIST    3
#define URGENCY 4
#define MENU    5
#define TEXTBOX 6
typedef struct {
               char *name;
               char *val;
               } entry;

void main(void)
{

       entry entries[MAX_ENT];
       int cont_len,index,marker;
       char address[256];
       FILE *mdata, *log;

       index=marker=0;

       cont_len = atoi(getenv("CONTENT_LENGTH"));

    /*separate values and put into entries structure */
   for(index=0;cont_len&& (!feof(stdin));index++)
   {
       marker=index;
       entries[index].val = fmakeword(stdin,'&',&cont_len);

       if ( index == BODY)
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00438

Continued from previous page

```
                plustospace(entries[index].val);
        else
                plusto_(entries[index].val);

        unescape_url(entries[index].val);
        entries[index].name = makeword(entries[index].val,'=');
    }

        printf("Content-type: text/html%c%c",LF,LF);

        /* put together e-mail address */

        sprintf(address, "%s -s \"%s\" %s",
                    MAILER,entries[LIST].val,MY_EMAIL);

        if (!(mdata=popen(address,"w"))){
                printf("<h1>Unable to open mail pipe</h1>%c",LF);
                exit(-1); }
           fprintf(mdata,"New message from %s %s arrived.",
                entries[FIRST].val,entries[SECOND].val);
/* return html code to browser */
        printf("<h1>Message sent!</h1> %c",LF);
        printf("content follows:<p><hr>%c",LF);

    log=fopen(LOGFILE, "a");
        fprintf(log,"----------\n");

    for(index=0; index <= marker; index++)
        {
        printf("%s:  %s<p>",entries[index].name,
               entries[index].val);
        fprintf(log,"%s:  %s\n",entries[index].name,
               entries[index].val);
        }

    fprintf(log,"----------end--\n");

        pclose(mdata);
        fclose(log);
        exit(0);
}
```

Here's an exercise. To get an idea of what your encoded strings look like to your CGI scripts, write the following shell script:

```
#include <stdio.h>
#include <stdlib.h>
#define LF 10
void main(int argc, char *argv[])
{
```


Google/YouTube Exhibit 1005/1105
Page 00439

```
        int cont_len,index;
        char c;

        cont_len = atoi(getenv("CONTENT_LENGTH"));

        printf("Content-type: text/html%c%c",LF,LF);
        printf("<listing>\n");
    for(index=0;index<cont_len;index++)
    {
        c = getchar();
        printf("%c",c);
    }
}
```

Compile this program and place it in the cgi-bin directory. Now, using one of the forms you've created, or one of the forms described in this book, change the <ACTION> tag to point at this program. (You must use the POST method for this to work.) Now, when you submit a form that points to this script, it should return the entire encoded string that your form sent it. This can be a useful tool if your decoding algorithms aren't doing what you expect.

LESSON #7: SAMPLE SCRIPTS FOR UNIX, WINDOWS, AND MACINTOSH SERVERS

This final section offers a number of educational, useful, and/or interesting CGI scripts, with something for whatever platform you're likely to be using. Most of them come from public archive sites around the Internet community, and their authors deserve great thanks for making them publicly available.

Unix

Unix is generally the CGI programmer's operating system of choice. Most of the more intriguing scripts seem to have been written for Unix servers. This doesn't mean that they can't be revised to run on other servers; in fact, many of them can be modified to run on other platforms with very little work. It may take some doing to convert a shell script to an AppleScript program, but the C programs should port quite nicely. Note that any C programs that are listed below that contain "#include "util.h"" must be linked with the util.c functions listed earlier in this chapter. Well, enjoy!

Google/YouTube Exhibit 1005/1105
Page 00440

fax_mailer.c

```
/*      This works with a specific fax-modem terminal that the author
runs locally. The fax modem takes e-mail as its input. This may need
to be modified to work on your system.

        HTML Fax Utility. Should be run as a cgi file under
        HTTPD. Takes the variable string supplied from the
        HTML "Form" submittal and parses it into an e-mail
        address.... The order that the variables appear in the
        html form is important. They must appear in the form
        in the same order that they are listed in the defines
        below, that is, AT..FROM. (Not the best approach for the
        job; it'll get fixed at some point.) This will produce an
        e-mail message of the form:

        /FN=995-4122/AT=Troy_Downing/O=NYU/@text-fax.nyu.edu

        text-fax.nyu will parse the "to" line and create a fax
        cover page and attempt to fax it to the number listed
        after /FN=. Currently, all spaces are converted to
        underscores in the address but not in the body.

        The fax is mailed from nobody@yourserver.com and this
        will appear on the header as the sender. So, It's
        important to include the FROM string so that the
        recipient will know who it came from....

*/

#include <stdio.h>
#include <stdlib.h>
#include "util.h"
#define MAILER  "/usr/ucb/mail"
#define LOGFILE "/usr/logs/fax.log"
#define LF      10
#define MAX_ENT 1000                /*  max number of variables
                                        that can be sent from form*/
#define FAX     "@text-fax.myserver.com"

/* header index. used to index the struct array containing the
  variables passed from mosaic */
#define AT      0
#define SUBJECT 1
#define FN      2
#define O       3
#define OU      4
#define BODY    5
#define FROM    6
typedef struct {
                char *name;
                char *val;
```


Google/YouTube Exhibit 1005/1105
Page 00441

```
                } entry;

void main(int argc, char *argv[])
{

        entry entries[MAX_ENT];
        int cont_len,index,marker;
        char address[256];
        FILE *mdata, *log;

        index=marker=0;

        cont_len = atoi(getenv("CONTENT_LENGTH"));

    for(index=0;cont_len&& (!feof(stdin));index++)
    {
        marker=index;
        entries[index].val = fmakeword(stdin,'&',&cont_len);

        if ( index == BODY)
                plustospace(entries[index].val);
        else
                plusto_(entries[index].val);

        unescape_url(entries[index].val);
        entries[index].name = makeword(entries[index].val,'=');
    }

        printf("Content-type: text/html%c%c",LF,LF);

        /* put together e-mail address */

        sprintf(address, "%s -s %s /FN=%s/AT=%s/O=%s/OU=%s/%s",
                MAILER, entries[SUBJECT].val, entries[FN].val,
                entries[AT].val, entries[O].val, entries[OU].val,FAX);

        if (!(mdata=popen(address,"w"))){
                printf("<h1>Unable to open mail pipe</h1>%c",LF);
                exit(-1); }

        fprintf(mdata,"%s %c",entries[BODY].val,LF);
        fprintf(mdata,"%c%cMessage Sender:
%s",LF,LF,entries[FROM].val);

        printf("<h1>Mail sent!</h1> %c",LF);
        printf("content follows:<p><hr>%c",LF);

    log=fopen(LOGFILE, "a");
        fprintf(log,"----------\n");

    for(index=0; index <= marker; index++)
        {
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00442

Continued from previous page

```
        printf("%s:  %s<p>",entries[index].name,
               entries[index].val);
        fprintf(log,"%s:  %s\n",entries[index].name,
               entries[index].val);
        }

    fprintf(log,"----------end--\n");

        pclose(mdata);
        fclose(log);
        exit(0);
}
```

mailer.c

This next script will send e-mail submitted via a form. It can be useful if users can't use mailto: URLs with their browsers.

```
#include <stdio.h>
#include <stdlib.h>
#include "util.h"
#define MAILER          "/usr/ucb/mail"
#define USER            "you@youraddress.edu"
#define SUBJECT         "Webpage_Reply"
#define LF              10
#define MAX_ENT         10000       /*  max number of variables
                                        that can be sent from form*/

#define FROM            0
#define EMAIL           1
#define USUBJECT        2
#define BODY            3
typedef struct {
               char *name;
               char *val;
               } entry;

void main(int argc, char *argv[])
{

        entry entries[MAX_ENT];
        int cont_len,index,marker;
        char address[256];
        FILE *mdata, *log;

        index=marker=0;

        cont_len = atoi(getenv("CONTENT_LENGTH"));

    for(index=0;cont_len&& (!feof(stdin));index++)
    {
        marker=index;
        entries[index].val = fmakeword(stdin,'&',&cont_len);
```


Google/YouTube Exhibit 1005/1105
Page 00443

```
                plustospace(entries[index].val);

        unescape_url(entries[index].val);
        entries[index].name = makeword(entries[index].val,'=');
    }

        /* put together e-mail address */

        sprintf(address, "%s -s %s %s",
                MAILER, SUBJECT, USER);

        /* open pipe to mailer */

        if (!(mdata=popen(address,"w"))){
                printf("<h1>Unable to open mail pipe</h1>%c",LF);
                exit(-1); }

        /* send data to mailer */

        fprintf(mdata,"Subject: %s%c%c",entries[USUBJECT].val,LF,LF);
        fprintf(mdata,"%s %c",entries[BODY].val,LF);
        fprintf(mdata,"%c%cMessage Sender:
%s",LF,LF,entries[FROM].val);
        fprintf(mdata,"%c%cEmail: %s",LF,LF,entries[EMAIL].val);

        /* put together HTML confirmation for mosaic user */

        printf("Content-type: text/html%c%c",LF,LF);

        printf("<h1>Mail sent!</h1> %c",LF);
        printf("<h2>If you included an e-mail address you should
receive
                a reply shortly.</h2>");
        printf("content follows:<p><hr>%c",LF);

    for(index=0; index <= marker; index++)
        printf("%s:  %s<p>",entries[index].name,
               entries[index].val);
    printf("<p><a href=\"http://found.cs.nyu.edu/\">RETURN</a><p>");

        pclose(mdata);
        exit(0);
}
```

mailer.html

```
<HEAD>
<TITLE>Mailer form</TITLE>
<H1> Mailer </H1>
<br>
<hr>
<i>Please send us your problems/questions/comments and we'll get back
to you as soon as we can</I>
```

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00444

Continued from previous page

```
<hr>
<form action="http://yourserver.com/cgi-bin/mailer" METHOD="POST">
<pre>
<B>NAME:   </B><input type="text" name="sender" size=30><p>
<B>EMAIL:  </B><input type="text" name="Email" size=30><p>
<B>SUBJECT:</B><input type="text" name="subject" size=30><p>
</pre>
<hr>
<h2>Message...</h2>
<textarea name="body" rows = 15 cols = 55></textarea><br>
<input type="submit" value="Submit Message"><br>
</form>
<hr>
<br>
If your mosaic browser does not support forms, send comments to:<p>
<B>your @email.address</B>.
<P>
```

names.c

```
/* This program parses a form that contains 3 input fields: a name
suggestion for my expected new baby, a sex which could be
male/female/either, and a comment. The suggestions are added to a data-
base that I'm keeping until I actually have to name the kid. */
#include <stdio.h>
#include <stdlib.h>
#include "util.h"
#define LF              10
#define MAX_ENT         100
#define NAME            0
#define SEX             1
#define COMMENT         2
#define NAMES           "/usr/pub/names"
typedef struct {
               char *name;
               char *val;
               } entry;

void main(int argc, char *argv[])
{

        entry entries[MAX_ENT];
        int cont_len,index,marker;
        char address[256];
        FILE *names;

        index=marker=0;

        cont_len = atoi(getenv("CONTENT_LENGTH"));
    for(index=0;cont_len&& (!feof(stdin));index++)
    {
        marker=index;
```


Google/YouTube Exhibit 1005/1105
Page 00445

```
        entries[index].val = fmakeword(stdin,'&',&cont_len);
                plustospace(entries[index].val);
        unescape_url(entries[index].val);
        entries[index].name = makeword(entries[index].val,'=');
    }
        printf("Content-type: text/html%c%c",LF,LF);
        names = fopen(NAMES, "a");

fprintf(names,"%s\t\t%s\t%s\n",entries[NAME].val,entries[SEX].val,
                entries[COMMENT].val);
        fclose(names);
        system("/usr/bin/sort /shaggy.a/downing/pub/names >
/shaggy.a/downing/pu
b/tmp");
        system("/usr/bin/mv /shaggy.a/downing/pub/tmp
/shaggy.a/downing/pub/names");
        printf("<h1>Thanks for your suggestion!</h1> %c",LF);
        printf("<h2>I will take the name \"%s\" into serious
consideration!<p>"entries[NAME].val);
        printf("<a
href=\"http://found.cs.nyu.edu/shaggy.a/downing/pub/names\">"
);
        printf("List of names</a><p>");
        printf("<a
href=\"http://found.cs.nyu.edu/downing\">return</a><p>");
        exit(0);
}
```

names.form

```
<form action="http://myserver.com/cgi-bin/names" METHOD="POST">
Please help suggest a name. Enter a name, comment if you want, and
sex.<BR>
name:<input type="text" NAME="name" size=15>
<select name="sex">
 <option>Male
 <option>Female
 <option>Either
</select>
comment:<input type="text" name="comment" size=30>
<input type="submit" value="submit name">
<a href="/usr/pub/names">List of names so far...</a>
</form>
```

Neat Tricks for HTML 3 Browsers

Recently, Netscape Communications released version 1.1 of their Web Browser. This was the first commercial release of an HTML 3 compliant browser that I am aware of. There are a couple of neat things this browser allows CGI programmers to do that I'm sure will be supported by the other

Google/YouTube Exhibit 1005/1105
Page 00446

major Web browsers shortly. Namely, the server push and the client pull. What the server push allows you to do is send a series of objects to the client rather than a single one. Normally, a CGI script sends a single type of data, say a text/html document or a GIF image and once this object has been passed on to the client, the connection is broken. With a server push, the connection is left open while the server sends a series of objects and is not closed until the script terminates. This is particularly useful using the multipart/x-mixed-replace MIME type with graphic images. This will allow you to send a graphic image to the client browser and then immediately replace it with another. By stringing a series of images like this together, you can create a sort of inline animation on the client's browser without using an external "helper application."

The client pull is similar to the server push but rather than having the server send another object, the client requests a series of objects after a specified interval. This comes in handy if you want the client to keep checking a document or script for changes automatically and for orchestrating "guided tours" without requiring user intervention. The client pull is implemented as an HTML tag that is interpreted by the client's browser. An example of a server push and a client pull follow.

Slide Show Animation— The following is the code for an inline animation script. It will send a series of GIF files to a browser, replacing each image with a subsequent one. It can easily be modified to use JPEG or XBM images as well. To work as intended, it should be called with an HTML document similar to the one that follows.

movie.c

```
/*      Multipart-mixed cgi demo.

        Troy Downing 1995

        This was written to demonstrate the multipart/mixed-replace
        capabilities that are now available with HTML 3.0 compliant
        browsers.

        This will send a series of images to a browser. Each image
        will replace the previous image giving the illusion of an
        animation. This will work best on fast networks with small
        images of the same size/resolution.

        Just to make things simple, I've named the images 1..8.gif;
        this could be easily modified to deal with other
        filenames/types.
*/
```


Google/YouTube Exhibit 1005/1105
Page 00447

```
#include <stdio.h>


#define BOUNDARY       "--ThisRandomString\n"      /*marks beginning of
file*/
#define ENDING         "--ThisRandomString--\n"   /*marks end of file*/
#define HEADER "Content-type: multipart/x-mixed-
replace;boundary=ThisRandomString"
#define TYPE           "Content-type: image/gif"  /*mime type of file*/
#define IMAGETYPE      "gif"                /*filename suffix*/
#define IMAGEDIR       "/usr/downing/gifs/"          /*contains  the
images*/
#define NAMELEN        256
#define REPEAT         8                     /*number of images to
send*/
#define BUF_SIZE       1024                 /*number of bytes to read
at once*/


void main(void)
{
        FILE   *f_spew;        /*points to image files*/
        char   file[NAMELEN];  /*holds image file name for use with
fopen()*/
        char   buffer[BUF_SIZE]; /*buffer for reading file*/
        int counter,count,tries;

        printf("%s\n\n",HEADER);       /*print the multipart header*/

        counter=REPEAT+1;
        while(counter--)               /*cycle through images*/
        {
           printf("%s",BOUNDARY);      /*print beginning boundary*/
           printf("%s\n\n",TYPE);      /*print mime type for image*/

             sprintf(file,"%s%d.%s",IMAGEDIR,counter,IMAGETYPE);
/*construct filename*/
           while((f_spew = fopen(file,"r"))==NULL) /*open file*/
           {
               if(tries--<0) break;
           }
           while (!feof(f_spew))       /*send data while not EOF*/
           {
              count = fread(buffer, 1, BUF_SIZE, f_spew);
              fwrite(buffer, 1, count, stdout);
           }

           fclose(f_spew);

           printf("%s",ENDING);        /*print ending boundary*/
       }

}
```


Google/YouTube Exhibit 1005/1105
Page 00448

Now, here is the HTML page that calls this program. I'm assuming that the code was compiled as "movie.cgi" and placed in the cgi-bin directory.

```
<HTML>
<HEAD>
<TITLE>Movie Test</TITLE>
</HEAD>
<body>
The following box should show a series of images. Click the
"reload" button to restart it.<BR>
<CENTER>
<TABLE border = 10><td><img src = "http://yourserver.com/cgi-bin/
movie.cgi"></td>
</TABLE>
</CENTER>
</BODY>
```

The following is an example of a client pull. The client pull is specified with an HTML <META> tag. For more information on the HTML 3 specs, see Chapter 10. This page will automatically reload itself every 30 seconds.

```
<HTML>
<HEAD>
<TITLE> Client Pull demo </TITLE>
<META http-equiv="Refresh" content=30>
</HEAD>
<BODY>
The following is a picture of our lab. It will
refresh every 30 seconds.<BR>
<IMG SRC="HTTP://ourserver.com/cgi-bin/take_pic.cgi">
</BODY>
```

DOS/Windows Scripts

There are a number of DOS and Windows scripts available on the Internet archives. Many are written as batch files, C programs, compiled BASIC programs, or PASCAL programs. Since most of the C programs written for Unix can easily be modified to work on a DOS system, this section will concentrate on the batch files.

A few notes about file location: The standard directory for CGI scripts on a Unix server is in the cgi-bin directory relative to the server root directory. As described in Chapter 21, the HTTPD server is installed relative to a directory specified in the configuration files for a particular installation. This directory is usually something like C:\HTTPD and all of the document, configuration, and CGI files are in directories relative to this. Any reference to "server root" is referring to this directory. For example: Given the above installation, to say that the cgi-bin directory is relative to the server root

Google/YouTube Exhibit 1005/1105
Page 00449

directory is the same as saying C:\HTTPD\CGI-BIN. Likewise, the document root directory is the directory that you have configured your server to look for HTML documents in. Normally this would be something like C:\HTTPD\HTDOCS. The HTTPD server for Windows is very similar to the Unix version. Relative to server root, there is normally a cgi-win and a cgi-dos directory. You should place your DOS and Windows CGI scripts into one of these corresponding directories.

args.bat

```
rem
rem ************
rem * ARGS.BAT *
rem ************
rem
rem Script used in args.htm to illustrate argument transfer
rem
rem Bob Denny <rdenny@netcom.com>
rem 30-Apr-94
rem
rem Echo is OFF at script entry
rem
set of=%output_file%
echo Content-type:text/plain > %of%
echo. >> %of%
echo CGI/1.0 test script report: >> %of%
echo. >> %of%
echo argc = %#  argv: >> %of%
if NOT %#==0  echo %1 %2 %3 %4 %5 %6 %7 %8 >> %of%
if %#==0  echo {empty} >> %of%
echo. >> %of%
echo environment variables: >> %of%
echo REQUEST_METHOD=%REQUEST_METHOD% >> %of%
echo SCRIPT_NAME=%SCRIPT_NAME% >> %of%
echo QUERY_STRING=%QUERY_STRING% >> %of%
echo PATH_INFO=%PATH_INFO% >> %of%
echo PATH_TRANSLATED=%PATH_TRANSLATED% >> %of%
echo. >> %of%
if NOT %REQUEST_METHOD%==POST goto done
echo CONTENT_TYPE=%CONTENT_TYPE% >> %of%
echo CONTENT_FILE=%CONTENT_FILE% >> %of%
echo CONTENT_LENGTH=%CONTENT_LENGTH >> %of%
echo ---- begin content ---- >> %of%
type %CONTENT_FILE% >> %of%
echo. >> %of%
echo ----- end content ----- >> %of%
echo. >> %of%
:done
echo -- end of report -- >> %of%
```


Google/YouTube Exhibit 1005/1105
Page 00450

demoindx.bat

```
rem
rem ****************
rem * DEMOINDX.BAT *
rem ****************
rem
rem Offers an ISINDEX document if no query arguments,
rem else reports on the "results" of the query.
rem
rem Bob Denny <rdenny@netcom.com>
rem 28-Apr-94
rem
set of=%output_file%
if NOT %#==0 goto shoquery
rem
rem No query, signal server to do redirect to ISINDEX demo doc.
rem
echo Location: /demo/isindex.htm > %of%
echo. >> %of%
goto done
rem
rem There were query arguments. Generate plain text report
(COMMAND.COM: BAH!)
rem
:shoquery
echo Content-type:text/plain > %of%
echo. >> %of%
echo Here is what the server would have fed to the back-end program:
>> %of%
echo. >> %of%
echo Number of query arguments = %# >> %of%
echo. >> %of%
echo Arguments: >> %of%
echo %1 %2 %3 %4 %5 %6 %7 %8 %9 >> %of%
:done
echo -- end of report -- >> %of%
```

Macintosh Scripts

Most Macintosh scripts are written as AppleScript programs. Here is an AppleScript CGI script and an HTML file to give you an idea of how Macintosh CGI scripting works. Dennis Wilkinson wrote the code you see here as an example of how to get MacHTTP to deal with data from forms supported by clients like XMosaic 2.0.

query.applescript

```
tell window 1 of application "Scriptable Text Editor"
        set contents to http_search_args
```


Google/YouTube Exhibit 1005/1105
Page 00451

```
        return "<title>Server Query Response</title><h1>Hi!</h1>We get
        the picture. Thanks for the feedback.<P><address>Here</Address>"
end tell
```

query.html

```
<title>Feedback Form</title>
<h2>Submit your feedback</h2>
<form method=GET action="http:/form.script">
Name: <input name="username"><p>
E-Mail: <input name="usermail"><p>
<select name="feedback">
<option selected>I Love It!
<option>I'm Lost!
<option>I Hate It!
</select><p>
<input type=submit value="Submit your feedback now"><p>
<input type=reset value="Reset this form"><p>
```

Table 15-4 MIME Types

</form>MIME Type	File Extension(s)
application/activemessage	
application/andrew-inset	
application/applefile	
application/atomicmail	
application/dca-rft	
application/dec-dx	
application/mac-binhex40	
application/macwriteii	
application/msword	
application/news-message-id	
application/news-transmission	
application/octet-stream	bin
application/oda	oda
application/pdf	pdf
application/postscript	ai eps ps
application/remote-printing	
application/rtf	rtf
application/slate	
application/x-mif	mif
application/wita	

Continued on next page

Google/YouTube Exhibit 1005/1105
Page 00452

Continued from previous page

</form>MIME Type	File Extension(s)
application/wordperfect5.1	
application/x-csh	csh
application/x-dvi	dv
application/x-hdf	hdf
application/x-latex	latex
application/x-netcdf	nc cdf
application/x-sh	sh
application/x-tcl	tcl
application/x-tex	tex
application/x-texinfo	texinfo texi
application/x-troff	t tr roff
application/x-troff-man	man
application/x-troff-me	me
application/x-troff-ms	ms
application/x-wais-source	src
application/zip	zip
application/x-bcpio	bcpio
application/x-cpio	cpio
application/x-gtar	gtar
application/x-shar	shar
application/x-sv4cpio	sv4cpio
application/x-sv4crc	sv4crc
application/x-tar	tar
application/x-ustar	ustar
audio/basic	au snd
audio/x-aiff	aif aiff aifc
audio/x-wav	wav
image/gif	gif
image/ief	ief
image/jpeg	jpeg jpg jpe
image/tiff	tiff tif
image/x-cmu-raster	ras
image/x-portable-anymap	pnm
image/x-portable-bitmap	pbm
image/x-portable-graymap	pgm
image/x-portable-pixmap	ppm

Google/YouTube Exhibit 1005/1105
Page 00453

</form>MIME Type	File Extension(s)
image/x-rgb	rgb
image/x-xbitmap	xbm
image/x-xpixmap	xpm
image/x-xwindowdump	xwd
message/external-body	
message/news	
message/partial	
message/rfc822	
multipart/alternative	
multipart/appledouble	
multipart/digest	
multipart/mixed	
multipart/parallel	
text/html	html
text/plain	txt
text/richtext	rtx
text/tab-separated-values	tsv
text/x-setext	etx
video/mpeg	mpeg mpg mpe
video/quicktime	qt mov
video/x-msvideo	avi
video/x-sgi-movie	movie

WHAT NOW?

There are popular services that you can use in your Web publishing. The next chapter looks at some of the major non-HTTP services available through WWW browsers.

Google/YouTube Exhibit 1005/1105
Page 00454

THE WAITE GROUP®

HTML WEB PUBLISHER'S CONSTRUCTION KIT

PUBLISHING YOUR OWN HTML PAGES ON THE INTERNET

HTML Web Publisher's Construction Kit provides everything the aspiring Web publisher will need to set up shop in the world's cutting-edge emporium of ideas, sound, and imagery.

For the novice, the book covers every detail, from obtaining the optimal Internet account to installing the converters, viewers, protocol transfer utilities, and communication sockets necessary for full Web capabilities. Accessible chapters walk the reader through the premium Web Browsers—SlipKnot, NetManage Chameleon—included on the enclosed CD-ROM.

HTML Web Publisher's Construction Kit explains in straightforward terms popular methods and options for creating Web pages. The book contains actual Web documents that are dissected, illustrating precisely how each aspect was created. Finally, it shows you how to upload and post your finished Web publications, providing a roster of free and commercial Web servers. *HTML Web Publisher's Construction Kit* is your road map to the exciting terrain of World Wide Web publishing.

- ✔ **Connect to the World Wide Web.** Access a world of information with a click of the mouse.
- ✔ **Browse the WWW.** Cruise easily with Lynx, SlipKnot, NCSA Mosaic, WinWeb, and tools Trumpet Winsock and The Internet Adapter.
- ✔ **Author WWW Pages.** Check out HTML specification 3.0, as well as authoring tools such as HTML Assistant and HoTMetal.
- ✔ **Design Top Documents.** Learn layout basics, CGI scripts, and Gopher, Usenet, FTP, e-mail, and WAIS display.
- ✔ **Publish Without Perishing.** Upload and post completed Web pages, locally and via FTP.
- ✔ **Configure Killer Web Sites.** Construct impressive Web sites and servers under Windows, Unix, and Mac. It's easy!

Platform: PC, Mac, Unix
Operating System: Windows, MacOS, Unix
Requires: 386 DX or better; VGA card and display; 4 MB RAM; hard drive; CD-ROM drive; access to the Internet

$36.95 USA
$53.95 Canada

SHELVING: INTERNET

ISBN 1-57169-018-2

53695>

9 781571 690180

WAITE GROUP PRESS™
200 Tamal Plaza
Corte Madera, CA 94925

LIBRARY OF CONGRESS
0 051 394 291 6

MRCRR

FOX

HTML WEB PUBLISHER'S CONSTRUCTION KIT

QA 76
.76
.H94 F69
1995