

HTML

MANUAL OF STYLE



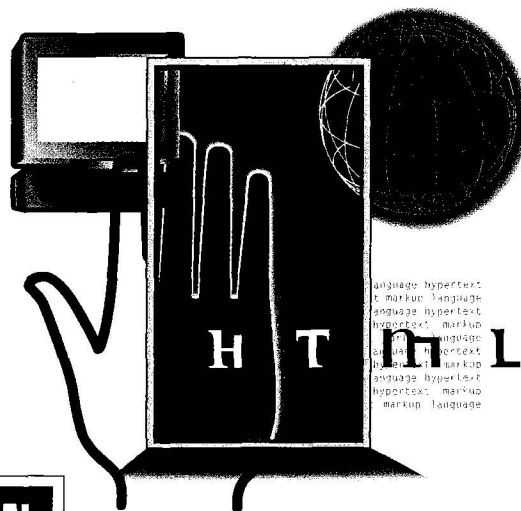
Clear, concise reference for **hypertext
markup language (HTML)**



- ◆ For anyone who wants to make information available to users of **Mosaic** or other **World Wide Web** browser programs

★ ★ ★ ★ ★

- ◆ Features annotated illustrations of **HTML** techniques



LARRY ARONSON

Copy Editor	Kelly Green
Technical Reviewer	Clay Shirky
Project Coordinator	Ami Knox
Proofreaders	Ami Knox and Barbara Dahl
Cover Illustration	Regan Honda
Cover Design	Regan Honda
Book Design	Gary Suen
Screen Graphics Editor	Pipi Diamond
Word Processing	Howard Blechman
Page Layout	Tony Jonick
Indexer	Carol Burbo

Ziff-Davis Press books are produced on a Macintosh computer system with the following applications: FrameMaker®, Microsoft® Word, QuarkXPress®, Adobe Illustrator®, Adobe Photoshop®, Adobe Streamline™, MacLink® Plus, Aldus® FreeHand™, Collage Plus™.

If you have comments or questions or would like to receive a free catalog, call or write:

Ziff-Davis Press
5903 Christie Avenue
Emeryville, CA 94608
1-800-688-0448

Copyright © 1994 by Ziff-Davis Press. All rights reserved.

Ziff-Davis Press and ZD Press are trademarks of Ziff Communications Company.

All other product names and services identified throughout this book are trademarks or registered trademarks of their respective companies. They are used throughout this book in editorial fashion only and for the benefit of such companies. No such uses, or the use of any trade name, is intended to convey endorsement or other affiliation with the book.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording, or otherwise, without the prior written permission of Ziff-Davis Press, except as permitted by the Copyright Act of 1976, and except that program listings may be entered, stored, and executed in a computer system.

THE INFORMATION AND MATERIAL CONTAINED IN THIS BOOK ARE PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY CONCERNING THE ACCURACY, ADEQUACY, OR COMPLETENESS OF SUCH INFORMATION OR MATERIAL OR THE RESULTS TO BE OBTAINED FROM USING SUCH INFORMATION OR MATERIAL. NEITHER ZIFF-DAVIS PRESS NOR THE AUTHOR SHALL BE RESPONSIBLE FOR ANY CLAIMS ATTRIBUTABLE TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN THE INFORMATION OR MATERIAL CONTAINED IN THIS BOOK, AND IN NO EVENT SHALL ZIFF-DAVIS PRESS OR THE AUTHOR BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF SUCH INFORMATION OR MATERIAL.

ISBN 1-56276-300-8

Manufactured in the United States of America
10 9 8 7 6 5 4 3

Writing HTML Documents

GENERAL HTML PRINCIPLES

GOOD HTML STYLE

CREATING A HOME PAGE

CONVERTING AN EXISTING DOCUMENT TO
HTML

Chapter 3

HTML is so easy that it's tempting to jump right in and starting writing text with markup tags, checking the files with Mosaic or one of the other graphical browsers and making corrections as you go. You can put information on the World Wide Web very quickly. In a day or two, you can create an HTML hypertext work that will establish a solid presence on the Web for you or your organization. However, it's just as easy to create a sloppy hypertext work as it is to create a neat one. In this chapter you'll find a discussion of the principles of writing good World Wide Web pages, plus step-by-step walkthroughs of two kinds of Web applications—building a personal home page and converting an existing document to hypertext.

Of course, all of this is rapidly growing and changing. Perhaps the most widely observed convention found in Web documents is the inclusion of the phrase "Under Construction." You can write a home page in a few hours, but you'll never be finished with it—it will grow as you do. This is one reason observing a few principles of good design can be so important. The creation of a personal home page, in particular, is an act of creative expression in a brand new medium. It is the setting up of your booth in Cyberspace to provide information, goods, and services, and to define who you are to the Global Electronic Village.

GENERAL HTML PRINCIPLES

Another reason good design is so important with Web applications is that you have no control over the context from which people will establish links to your Web pages. Think of your Web application as a house in Cyberspace; the door

is always open. Each HTML page is a room in this house. Most people will enter via your home page. A good home page takes care to properly welcome its visitors and let them know where they are and what interesting resources are to be found inside. The navigation controls of their browser will let the reader exit the way they came in; still, it's nice when the home page provides suggestions and links to other places in Cyberspace to visit.

Not everybody will enter your Web application through its home page. Some people will come in through the windows of other rooms in your Cyberspace house. There are a number of automated programs that continually explore the World Wide Web, building databases of titles, headings, and URLs as they link from one Web server to another. These are sometimes called *robots*, *spiders*, *worms*, or *web walkers*. There is a page on the Web with information on known robots at <http://web.nexor.co.uk/mak/doc/robots/robots.html>. You could ask the World Wide Web Worm, for example, to provide a list of all Web pages that have the word *fractal* in their title. Such links are independent of the structure the authors of those pages intended. The point is that readers will find ways you didn't anticipate to enter your hypertext work. Help these people out; at a minimum, provide a link back to your home page from every other page you put on the Web. Don't leave lost readers feeling more lost than when they entered.

Remember also that your HTML documents—the source code of your Web application—are readily available to anyone who can access the Web. Other HTML authors may copy elements of your pages and incorporate them into their Web applications. Hypertext works on the World Wide Web are living, growing structures. If you keep this in mind, with a little preparation, practice, and planning, your hypertext works can grow and evolve as smoothly as the Web does.

Probably the best preparation for writing HTML documents for the World Wide Web is reading World Wide Web HTML documents. Get a feel for what other authors have put on the Web and the approaches they've taken in organizing and formatting their work. You'll need a graphical browser to fully appreciate what others have done with HTML. You should at least have NCSA Mosaic. Other browsers may have more features than Mosaic; however, most of the documents currently on the Web have been written with Mosaic in mind. Mosaic is great software. It comes in versions for Microsoft Windows, Apple Macintosh, and Unix X/Windows, and it's free—a heck of a deal.

The home page for all three versions of Mosaic, <http://www.ncsa.uiuc.edu/SDG/Software/Mosaic>, is a good place to start your study of Web pages. Another good place is NCSA's What's New page on the same server, <http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/whats-new.html>. This page is updated every two weeks with links to Web pages that have just been created.

It's a good sampling of what people in many different fields are doing right now on the Web with HTML.

Mosaic and most of the other graphical browsers can save any HTML document on the World Wide Web in its original form as an HTML file. It is then a simple matter to open that HTML file in a text editor to see how HTML was used to create the display seen in Mosaic's window. As practice in writing HTML documents, edit these files and change the tags. Mosaic has an Open Local... choice under its File menu that you can use to view the reedited pages to see your changes reflected in the browser's display. After looking at a number of HTML documents and playing with the elements of HTML, you should start to have an idea of what you can do with your own applications. But before you start, it's good to review a few principles that apply to computer programming in general but have special application to HTML documents on the World Wide Web.

General principle number 1: Keep it simple. Emphasize content over form. You have little control over the exact look your document will have when viewed by the various browsers readers have at their disposal, so don't waste a lot of effort trying to get something to look just right; instead, spend your time making the content—the information you want to convey to the reader—clear and compelling. If the typography of a document must be exact, consider putting a version of that document on your Web server in a format that can be downloaded by the reader—a Microsoft Word document in RTF (Rich Text Format) or an Adobe PostScript or Acrobat file, for example.

Make sure the images on your Web page are informative. A small picture of you on your home page provides readers with information that words alone cannot convey. A picture of your car, unless you've done something extraordinary with it, does not. Avoid putting up Web pages that emulate automated teller machines with large graphic buttons for links. They take too long to load and are harder to maintain. HTML provides different list structures, one of which should do to organize any set of links. As the application evolves, you'll find it a lot easier to change these list structures than you would a set of pictures. Your Web page should not look like a magazine cover, either. Readers will choose to link to your page because they are browsing related information, not because of your cool graphics. Web pages are not in competition with each other for the reader's attention.

The Internet, and the World Wide Web in particular, is growing so fast that many Internet experts are worried that bandwidth, essentially the networks' capacity to handle traffic, is starting to get scarce. So it's considered polite practice (good netiquette) to keep the amount of data you're asking others to move across their networks on your behalf to a minimum. Large graphic elements eat up

bandwidth. Your PC graphics program is probably great at creating beautiful three-dimensional bar charts with gradient color fills, but you can probably provide the same information with a two-dimensional black-and-white graph. One approach commonly found on the Web is the use of a small version of a graphic, often called a thumbnail, as a link to a larger version. For example, this bit of HTML

```
Click <A HREF="Large_AE.GIF"> Albert
<IMG SRC="Small_AE.GIF" ALT="Einstein" ALIGN=MIDDLE>
</A> to see a large (90k bytes) picture of the scientist.
```

creates the Mosaic display shown in Figure 3.1.

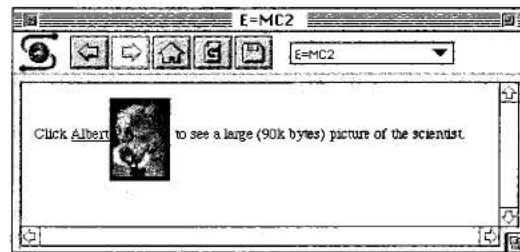


Figure 3.1: Using a thumbnail to access a larger image.

The relative URL addressing (using a partial URL to refer to the location of a file relative to the URL of the current page) used in generating Figure 3.1 requires that the two image files referenced in the HTML anchors—*Large_AE.GIF* and *Small_AE.GIF*—be in the same directory as the Web page containing the links. The name *Albert* and the small image are together the anchor of a link to the larger image file, *Large_AE.GIF*. Clicking either one will fetch the larger file and pass it to a helper application to display it in a separate window as an external image. By telling readers in the text how large the image file is, you provide them with the information needed to estimate how long it will take to load. Note the use of the ALT attribute in the IMG tag to direct non-graphical browsers to display the text “Click *Albert Einstein* to see a large (90k bytes) picture of the scientist.”

General principle number 2: Good work is never done. It is not until after your information is made available on the Web that you’ll begin to appreciate the hypertext structure that is natural for it. You should expect to frequently update and revise any work you put on the Web. As you add to and change the information on a page, you’ll have the opportunity to work with its structure, improve its looks, and replace any dead URLs—links to Web pages or servers

that no longer exist. You'll also have the benefit of feedback from other people who have read your work. It's a good idea—in fact it's an accepted Web convention—to include your signature and e-mail address on your work. And don't be embarrassed to ask for comments.

A decade ago, a speaker at a computer language conference I attended formulated this principle into the following law which he named after himself. Herewith is Biddlestone's law:

The requirements of any system are a function of the experience gained installing that system for the user.

One implication of this law is that whatever information you have, in whatever state it's in, you should put it up on your server and let people see it. It's better to make the information available with an explanation than to not make it available at all. It's not uncommon to come across a page on the World Wide Web (or on a gopher server) with a heading following by the phrase *Under construction* or *Work in progress*, so you should not be embarrassed to place unfinished or unverified information on your Web pages, as long as you inform your readers of this fact.

General principle number 3: Have fun :-).

The way you approach creating a hypertext work for the World Wide Web depends on what kind of information you want to serve and how much of is already in digital form. Broadly speaking, there are two approaches: top-down and bottom-up. If you are starting a work from scratch and there is little or no existing information already on line, then work from the top down. If there is already a lot of information available that needs organizing, or if there's an existing work to be converted to hypertext, then start from the bottom and work your way up. Of course, not all hypertext applications fall easily into one of these two categories. Most real-world projects are a combination of new work and existing material. This is typical of organizations that already make use of distributed information.

A third approach you might find useful is stealing. Yes, theft—if you find something you like, copy it. Edit other people's pages and replace their information with yours. You may prefer to think of this as *borrowing* ideas; if so, you should pay back the ideas with interest.

Stealing from other Web pages can be very productive. I must recommend, however, that you only copy the structure and the hypertext links, not the content from somebody else's Web page. Avoid copying anything from pages that have explicit copyright statements or that are connected to organizations with large financial and legal resources. When in doubt, ask. And give credit where it's due.

GOOD HTML STYLE

As the Web continues to grow, it becomes ever more important to write HTML that conforms to certain guidelines and styles. Right now there are about a dozen Web browsers available. A year from now I fully expect that number to triple. The major online services all have plans to provide Web browsers as part of their offerings. The new releases of the most popular operating systems, Microsoft Windows 95, Macintosh System 7.5, and OS/2 Warp have built-in Internet connectivity. This will greatly expand the potential Internet and World Wide Web user base; expect the major software publishers to respond with a flood of Web products.

So it's important to write HTML that will look good on any client, not just on Mosaic and the current generation of browsers. This section will offer some guidelines as to the do's and don'ts of writing good HTML documents that are easy to maintain and will produce presentable results on any browser. If you follow these guidelines, your document may not look its best on every browser, but it will not look ugly on any browser. Please keep in mind that none of this is written in stone. The Web, large as it is, is still in its infancy, and nobody wants to inhibit its growth—or, for that matter, yours.

It is a good idea to sign and date all documents you put on the Web so that your readers can form some impression of the authority of the document—how recent it is and how reliable the source of the information is. On a home page or any page that serves as an introduction to a hypertext work, your signature should include your full name and e-mail address so readers can send you comments on your work. You can make your name into a link to your personal home page. On less important pages of the work, your signature can just be your initials linked back to the authorship information on the home page.

You can add an external signature to a Web page by placing a link element in the heading of the page. It should be written

```
<LINK REV="made" HREF="mailto:email_address">
```

with your e-mail address appearing in place of *email_address* in the example. Place it after the title tags and before the closing `</HEAD>` tag. The REV (reverse relationship) indicates how this document is related to the object pointed to by the HREF URL—your e-mail address. The value *made* says that your e-mail address “made” this document. This convention is relatively new but gaining wide acceptance as a way to provide servers and browsers with your identity.

Remember that your documents are going on a *World Wide Web*, so when dating a document, use a long date format with the name of the month spelled

out or abbreviated—in other words, October 1, 1994, or 1-Oct-94. Formats such as 10/1/94 can be ambiguous.

Probably the most prevalent kind of error in writing HTML is the misuse of paragraph breaks. In part this comes from working so much with one browser that you begin to accept its handling of white space as common; it also comes from the misconception that the `<P>` element signals an end of paragraph, rather than a paragraph break. According to the HTML specification, “`<P>` is used between two pieces of text which otherwise would be flowed together”.

Usually this is not important. In certain contexts, however, use of extra paragraph tags should always be avoided, such as before or after any element that already implies a paragraph break. Avoid placing the `<P>` element either before or after headings or text marked with ADDRESS, BLOCKQUOTE, or PRE tags. You should not place paragraph tags immediately before or after a list structure or between the items of a list, either. The list item tags `<LI>`, `<DT>`, and `<DD>` already imply paragraph breaks. Some clarification is needed with the glossary list tags `<DT>` and `<DD>`. It is not legal HTML to have more than one `<DD>` tag following a defining term `<DT>` tag. If the definition part of the entry requires more than one paragraph of text, the use of paragraph tags to separate them is correct; the use of multiple `<DD>` tags per `<DT>` is not.

Another common source of error is not properly closing an HTML element. With character entities this means forgetting the trailing semicolon or having blanks separate the character entity from the rest of the text. It's also easy to forget that the ampersand is the escape character. Make sure you write `AT&T` and not `AT&T`. Forgetting one of the double quote marks that should enclose a URL is also a common error. Some browsers don't care if a URL is in quotes, but some do; many will have a problem if one quote mark is there and the other isn't.

With tag elements, errors can occur when the closing right angle bracket (`>`) is missing. Many browsers will properly render strings that contain a single right angle bracket with no matching left bracket (`<`) as if that character were part of the text. For example, `<lt;This is not a tag>` will be displayed as: `<This is not a tag>`. However, it is recommended that the character entity `>` be used for the right angle bracket—`<lt;This is not a tag>`—because if there are any other tag errors in the document, having an extra `>` around will only make matters worse. With nonempty markup tags, forgetting the slash (`/`) that begins the ending element will cause errors, as will having blanks on either side of the slash. In both cases, most (but not all) browsers will ignore the incorrect ending tag and produce the same errors as they would if you forgot the ending tag entirely. Whatever tag was in effect, its formatting would continue into the following text, possibly to the end of the page.

The nesting of tags should be done carefully. As a general rule, tags that define styles should be inside of tags that imply structure. Without enumerating all the possible combinations, here are a few guidelines:

Avoid nesting other tags inside of a heading. The big exception to this rule is anchors marking the text of the heading as the start or destination of a hypertext link. Headings should never contain any tags that imply paragraph breaks. This includes other headings, paragraph tags, horizontal rules, list structures, blockquotes, addresses, and preformatted style tags. If you want to create a multiple line heading, use the line break tag `<BR>`. Likewise, headings should never be enclosed in tags other than `<BODY></BODY>` and `<FORM></FORM>` and, of course, the `<HTML></HTML>` tags that define the document. Enclosing headings with any other tags doesn't make sense, and the results are unpredictable.

The use of style tags to change the rendering of a heading should be avoided except when applied to a small part of the heading text, for example:

```
<H3>Some <STRONG>Important</STRONG> phone numbers</H3>.
```

Image tags can be used inside of a heading to provide a small graphic counterpart to the heading text. Having the image inside the heading tags is necessary for the browser to recognize the `ALIGN` attribute and place the image properly with respect to the heading text.

anchors should be the innermost items of a set of nested tags. For example:

```
<H3>William Shakespeare</H3>
An English playwright ....etc., etc.,
<UL>
<LI><CITE><A HREF="...">Macbeth</A></CITE>
<LI>...
</UL>
```

One last point with regard to tags is that you should avoid using any obsolete tags; they can creep into your work if you copy parts of it from other pages on the Web. These elements include `<PLAINTEXT></PLAINTEXT>`, `<XMP></XMP>`, `<LISTING></LISTING>`, `<HPx></HPx>`, and `<COMMENT></COMMENT>`. The first three should be replaced with the preformatted tags `<PRE></PRE>`; `<HP></HP>` (highlighted phrase) should be replaced with appropriate style tags; and `<COMMENT></COMMENT>` should be replaced with SGML comments which are enclosed by the strings `<!--` and `-->`.

URLs can be a source of errors. A URL error won't affect the rendering of the page in a browser's display, but a badly composed URL may be incorrectly interpreted by some browsers. Relative URLs, which are used in the next two

sections of this chapter, have strong advantages—they're shorter and they make a collection of documents more portable. However, relative URLs should be used with care since the URL does not contain all the information necessary to construct the link. The missing server and path information is taken by the browser from the URL of the document that contains the link. Unfortunately, not all browsers do this exactly the same way. What is always safe is a relative reference to a file in the same directory as the current page. Files in a subdirectory can usually be referenced by using slash—a forward slash (/), not the backslash (\) used in DOS path names—as in:

```
our clerk, <A HREF="staff/Cratchet.html">Bob Cratchet</A>
```

You can also refer to the same file above by using a dot to refer to the current directory:

```
our clerk, <A HREF="./staff/Cratchet.html">Bob Cratchet</A>
```

You can refer to the parent directory by using two dots (..). However, this may not work with all browsers and servers. To be absolutely safe, any files that are not in the same directory as the current page should be referenced by their full path and file names.

Be careful with the use of the *file* URL method. What this is supposed to do is reference the file on the local host of whoever is browsing the document. This might be a file on your hard disk or it might be a file on your Internet server provider's computer. It is not supposed to be a file on the Web server pointed to by the current URL. Unfortunately, some early browsers treat *file* as synonymous with *ftp*. Unless you know what you are doing, avoid using this URL method.

These are only some of the more common sources of error. Developing a good HTML style is a matter of practice, studying the work of others, and finally, good common sense.

CREATING A HOME PAGE

A home page is the one hypertext document within a work that is intended as the primary starting point of the work. This is where the work is introduced and placed in the larger context in which it's contained. This definition has to be a bit abstract as it depends partially on the reader's point of view. The World Wide Web can be viewed as a single hypertext work, albeit one with many authors. The default home page for the Web is at CERN, the European Center for High Energy Physics Research. The URL is

```
http://info.cern.ch/hypertext/WWW/TheProject.html
```

Your personal home page is the starting point for all of your hypertext works. Since it will be created from scratch, I recommend using a top-down approach, starting with a simple outline like this:

- I. Your Name
 1. A welcome message
 2. Information about you
 3. A statement of your goals
 4. Current activities/announcements
 5. Related information
 6. Signature, address, time stamps

You needn't follow the above outline exactly, but this is the information readers generally expect to see on a personal home page.

Before you begin editing the HTML files you need somewhere to put them. Create a new directory for these files. A subdirectory of the one that contains Mosaic or your favorite browser is a good location as it will keep your path names short. Figure 3.3 refers to this directory as *WebSpace*; you can name your directory anything you want. Into this new directory place a text file containing an HTML template. Give this file an obvious file name such as *template.html*. If your operating system supports stationary or read-only documents, you should save this template as such to prevent overwriting it accidentally. Make sure you also save it as a plain text file instead of saving in your word processor's normal format. Next start with a copy of the template file and fill in the TITLE and BODY sections based on the outline above, like so:

```
<HTML>
<HEAD>
<TITLE>Johannes Kepler's Home Page</TITLE>
</HEAD>

<BODY>
<H1>Johannes Kepler</H1>
<STRONG>Welcome to my Home Page!</STRONG>
<HR>
<H2>Who am I</H2>
<H2>What is this Document</H2>
<H2>Current Projects</H2>
<H2>Related Information</H2>
<HR>
<ADDRESS>
```

```

Johannes Kepler &lt;kepler@nasa.gov&gt;
</ADDRESS>
Created: September 26, 1994.
Updated: September 27, 1994
</BODY>
</HTML>

```

and save it in WebSpace directory as homepage.html. All HTML documents should have filenames that end with the extension *.html* unless the files reside on a DOS system, in which case they should have the extension *.htm*. If you now load this file into Mosaic, you'll get the display shown in Figure 3.2.

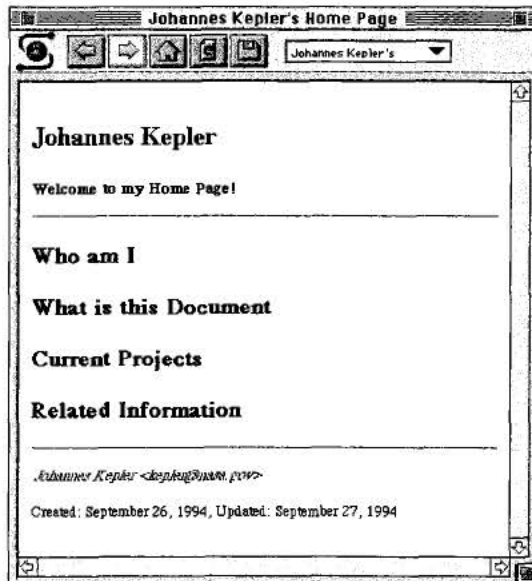


Figure 3.2: Johannes Kepler's home page—first draft

From here on, it's just a matter of filling in paragraphs of narrative text under each heading with perhaps some subheadings and maybe a picture of yourself. At this stage, don't worry about the hypertext structure; that will evolve as you develop the page from the top downward. If you find that under some heading you are just listing items, then use one of the HTML list structures. A common feature of many home pages is a list of links to the author's favorite Internet sites. Put such lists under the *Related Information* heading.

Now read what you've written. Did you mention the World Wide Web? If you did, then make that reference into a link to the World Wide Web's home page at CERN:

```
<A HREF="http://info.cern.ch/hypertext/WWW/TheProject.html">
World Wide Web</A>
```

Likewise for any organizations that you refer to in the text that have home pages on the Web.

Do you have a copy of your résumé on line? If you do, then create a plain text version of that file and save it in the WebSpace directory as *resume.txt*. The *.txt* extension (and the lack of `<HTML></HTML>` tags) tells browsers not to interpret the file as a hypertext document—in other words, to respect the carriage returns and other white space in the file and present it as is. Later on you can create an HTML version of this text file, but this will do for now. Your file structure should now look something like the directory listing (slightly simplified) in Figure 3.3. In the section that tells the reader who you are, make a link to the text file, like this:

```
...can see my <A HREF="resume.txt">resume</A>, if you wish.
```

Try to incorporate the text that anchors hypertext links into the natural flow of the paragraph's prose, as the example above does and not refer to the anchor text explicitly, as in:

```
Click <A HREF="resume.txt">here</A> to see my resume.
```

The word *here* has no semantic relation to the file that it points to. That is, if a program should scan your Web page recording each anchor URL and its associated anchor text, the word *resume* would provide additional information about the file it points to; the word *here* would not. Besides, you can pretty well assume that the readers know what is and what is not a link by the time they get to your page.

The entire home page should not be more than a few screens. If any section starts getting more than a few paragraphs long, it's time to create a new page and link to it from your home page. For example, suppose under the heading *Who am I* you list your favorite activity, astronomy. In the same manner that you created your home page, create an astronomy page containing a description of your equipment, your favorite planets, and perhaps, a link to the Usenet newsgroup `news:sci.astro`. Save the astronomy page in the WebSpace directory and on your home page, making the word "astronomy" into a link, like so:

```
blah, blah, blah, favorite activity:
<A HREF="astronomy.html">astronomy</A>.
```

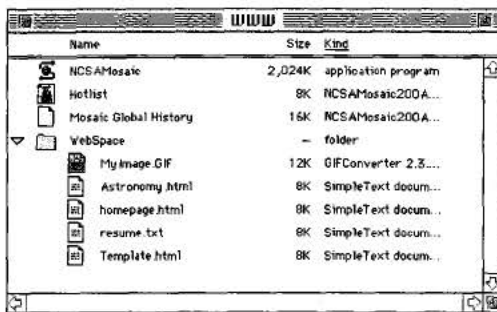


Figure 3.3: Directory structure of a simple home page

By keeping all the files in the same directory as your home page, all the links can be specified relatively with simple file names. When you copy these documents to a directory on a Web server, these links will still be valid. This way, you maintain a local working copy of your Web application that mirrors what you have available on the Web.

The final touch is to add your picture to your home page. There are many ways to get a picture onto your computer—scanner, digital camera, Photo CD—and the image file formats differ from one operating system to another. However you got your picture onto your computer and whatever file format it is in, it must be converted into one of two image file formats, GIF or JPEG, before you put it up in your WebSpace as a part of your home page.

Graphic Interchange Format (GIF) is the older and more commonly found format. It was invented by Compuserve and is widely used on the Internet and BBSs for downloadable images. It is the preferred format for inline images. JPEG, a standard developed by the Joint Photographic Equipment Group, is newer and not yet supported by many browsers. It incorporates data compression so that an image file in JPEG format can be considerably smaller than the same image in GIF format. There are many utilities programs, both shareware and freeware, for converting files from one image format to another.

When an image is expanded in a Web page, it is treated as if it were a character of text. If the image follows an explicit or implied paragraph break, it will be positioned at the left margin of the page. If the image begins with or is embedded in a line of text, that text will be aligned with the top, middle, or bottom of the image according to the value of the ALIGN attribute. No space is inserted either before or after the image to separate it from the text it is embedded in; you must explicitly provide that space, if you want it there.

It's quite permissible to place an image in a heading, for example:

```
<H2><IMG SRC="kepler.gif" ALIGN=MIDDLE> Johannes Kepler</H2>
```

or to use an image as the anchor of a link, as in

```
<A HREF="kepler.html">
<IMG SRC="kepler.gif" ALT="Johannes Kepler" ALIGN=BOTTOM>
</A>
```

Note the use of the ALT attribute; without it, readers with nongraphical browsers wouldn't know what the link was. All they would see is [\[IMAGE\]](#).

Unfortunately, there's no provision in HTML level 2 for wrapping text around an image. One thing that you can do, however, is make a transparent GIF image. The GIF format allows one color on the image pallet to be transparent so that when the image is placed on top of the background color of the display page, the background shows through whatever pixels have that color. Transparent GIFs are especially useful for placing images on a page so that they appear without a border, such as lettering in a fancy or colored typeface. See Chapter 4, Figure 4.7, for a home page with a transparent GIF image.

CONVERTING AN EXISTING DOCUMENT TO HTML

In contrast to creating a home page, converting an existing document to hypertext is best approached from the bottom up. Suppose you have a user's guide for some aspect of your business. It could even be a guide to using the Internet. We will assume that the document is *rich text*, a general term for any document that's more than a simple text file. A rich-text document uses type styles and sizes, margins, and line spacing. The ability to create rich-text documents is what distinguishes word processing programs, such as Microsoft Word, WordPerfect, and MacWrite, from text editors. SGML languages such as HTML provide machine-independent methods for adding rich-text information to a plain text document. Hypertext documents on World Wide Web servers are text files, so first you have to create a plain text version of the guide before you build it back up to rich text with HTML elements.

Working from the bottom up, you'll create a series of versions, each a refinement of the previous one. When the conversion of the guide is completed, you'll have a full hypertext version consisting of linked files, a single file version suitable for printing, a text-only version for readers with nongraphical browsers, and the original version in a stuffed and encoded form for downloading via ftp. On top of all this, you can create a home page for the guide that describes the work and has links to the various versions as well as authorship and status information and links to related works.

Let's assume that our user's guide is written and maintained in Microsoft Word. The first thing to do is create a new directory for the project—let's call it *Guide*. It can be a subdirectory of WebSpace or a sibling, whatever is appropriate to the relation of this work to your other hypertext works. Make working copies of the file (don't work on the original), one in Microsoft Word's normal format and one in text-only format, and put them in the Guide directory. Give the latter a file name ending in the extension *.txt*. This file is what you will point out to readers with line-mode browsers. Later you may want to edit this text-only version and put in notations where the figures and illustrations were.

Go back to the working copy in normal format. It will be the base from which the other versions are created. You can create the ftp version from this file by stuffing and encoding it. *Stuffing* compacts the file by removing excess space, *encoding* creates a 7-bit portable version of the file, allowing it to be moved and stored as a text file on any kind of operating system. Place the new file in the Guide directory.

Before you can add any markup tags to the working copy of the file you must replace the special characters that are needed for HTML markup with appropriate character entities. First do a global replace of the ampersand with string *&*. This will ensure that the final HTML document will be free of any unintended character entities. Next, replace all occurrences of the left angle bracket (<) and right angle bracket (>) with the strings *<* and *>*, respectively, to ensure that the only markup tags in the document are the ones you intended to be there. Then try to find and replace any of the special characters in the text from the ISO Latin-1 character set (such as å or ç), curly quotes, and symbols.

Horizontal tabs present a problem as they are not interpreted uniformly by all browsers, although most browsers will ignore them completely except in preformatted text. It's generally a good idea to get rid of as many tabs as possible by replacing them with an appropriate number of blanks. Delete any tabs that are used only for paragraph indentation and enclose any tables or other non-paragraph text with the preformatted style markup tags `<PRE>` and `</PRE>` to preserve their line breaks and spacing. Later on, you can go back and replace some of the preformatted sections with other HTML structures.

Note: The specification for HTML+ includes a TABLE markup. At the time of this writing, that specification is not complete and I can't recommend using it. This may change by the time you read this.

If there are figures and illustrations in the file (figures have captions, illustrations do not), they will have to be copied to individual files and removed from the text. Use a file-naming convention that preserves the image's location—for

example, GUIDE03-F04.PICT—for the fourth figure in Chapter 3 of the guide. The leading zeros in the file name will insure proper sorting in a directory listing if you have more than nine items. The .PICT extension used here indicates that this file is in the Macintosh PICT format. These image files now have to be encoded into the GIF format. As you remove each figure and illustration, leave an in-line image tag in its place with an ALT attribute describing it—for example

```
<IMG SRC="GUIDE03-F04.GIF" ALT="Fig. 3-4. A typical Gopher menu">
```

As with any other computer project, save frequently and make backups.

Remove any headers and footers from the guide, since you have no control over how many lines a browser will display per page or screen and page numbering is not important in hypertext works. At the beginning of the file create an HTML head section with the main title enclosed in title tags, followed by the starting tag for the body section, followed by the main title again as a level 1 heading. Place the matching end tags at the end of the file. It should look something like this:

```
<HTML>
<HEAD>
<TITLE>Internet Guide</TITLE>
</HEAD>
<!-- Guide.html, a hypertext guide to the Internet -->
<!-- Converted from MS Word file: INT_GUIDE -->
<!-- J. Kepler, September 29, 1994. -->
<BODY>
<H1>Internet Guide</H1>
...
{ rest of the file }
...
</BODY>
</HTML>
```

Save this file as a text-only file. Give it a file name with *.html* as the extension—Guide.html will do. Next, starting from the beginning of the file, work your way through, placing line breaks (
) and paragraph (<P>) tags where needed and enclosing headings in heading tags. Use heading level 2 for the major divisions of the guide, level 3 for the next level, and so on. As you go through the file, enclose any styled text with appropriate tags— for boldface text, for underlined text, <CITE></CITE> for text in italics, and so forth. Try to avoid using the explicit style tags , <U></U>, and <I></I> unless that style is explicitly referred to within the text. Look for places in the text where HTML lists can be used to structure information you previously marked as preformatted text. When you've completed this pass, the first draft of the HTML

version of the guide is finished. Save this file. Your directory should now look something like Figure 3.4:

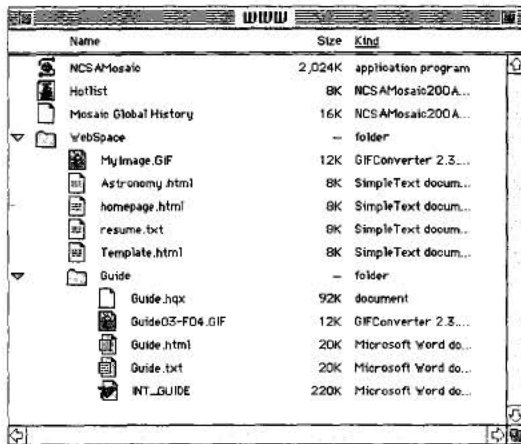


Figure 3.4: Directory listing with "Guide" files

It's time to load your work into Mosaic and see what it looks like. Print it out and show it around. You may at this point want to work with an HTML editor or a text editor with HTML extensions; See Appendix C for some of the software now available.

The next step is to go through the file adding anchors and links. Make the text of each of the major headings a named anchor. Use the heading's index from the table of contents as the name, for example:

```
<H3><A NAME="#G32">3.2 Gopher</A></H3>
```

In the table of contents, create a link to each of these headings:

```
<LI><A HREF="#G32">3.2 Gopher</A>
```

Do the same with any footnotes in the work. If there are references to other documents on your network or references to resources on the Internet, create links to them as well. If you're not sure about the URL, make a guess. An incorrect URL will not crash the system; you can repair these bad links when you have the correct information.

When you are finished with this process you'll have the long hypertext version of the guide. This version will be more suitable for printing than your final version will be, but will still be suitable for online browsing (although it may take some time to load over a slow line). Take some time to clean up and test this version, using different browsers if possible. If the original guide had a

glossary, an index, or a quick reference section, now is the time to decide how to implement these features in hypertext.

When the long version is fairly stable, decide how to break it up into separate files. Having a hypertext work in a series of linked files has several advantages. The individual files are faster to load and the reader can have more than one section of the work displayed at the same time. Ideally, no single part of the guide should be more than a dozen screens worth of information. The exception to this is very long list structures, which should go into separate files for easier maintenance. The table of contents will go into the guide's home page. If a section of the guide is long and has many subsections, consider creating a mini-home page for that section with its own, linked mini-table of contents.

Each file will have to begin and end with the HTML tags defining the head and body of the page. Copy the title of the work as a whole and make it a level 1 heading as the very first element of the page body of each file so your readers know what they're reading. Put a horizontal rule under the level 1 heading and you have a page header. Here is what the beginning of the Chapter 3 page might look like:

```
<HTML>
<HEAD>
<TITLE>Internet Guide, Chap. 3</TITLE>
</HEAD>
<!-- Guide03.html, Chapter 3 of Internet Guide      -->
<!-- Converted from MS Word file: INT_GUIDE        -->
<!-- J. Kepler, September 29, 1994.                 -->
<BODY>
<H1>Internet Guide</H1>
<HR>
<H2><A NAME="G30">Chapter 3 - Clients</A></H2>
A nice introductory paragraph should go here.
<H3>Contents</H3>
<MENU COMPACT>
<LI><A HREF="#G31">ftp</A>
<LI><A HREF="#G32">Gopher</A>
<LI><A HREF="#G33">...</A>
</MENU>
<H3><A NAME="G31">3.1 ftp</A></H3>
...
```

Any links to anchors that are now in different files must be updated to include the file name of the destination. Links in other files to anchors in this file, Guide03.html, will also have to be updated. The home page for the entire guide will contain the table of contents, the introduction, the guide's authorship information (make your name a link to your personal home page) and links to related works.

A nice touch is to add a set of navigation buttons to the bottom of each page in the guide. This next bit of HTML, at the end of the file for Chapter 3, creates a set of four text buttons that link to other files of the guide. The last link, [CONTENTS], goes to an anchor named *ToC* on the guide's home page, the file *GuideHome.html*. You can also use small icons for these buttons, but each icon will have to be in a separate file. Following the buttons is a link back to the authorship information on the guide's home page.

```
<HR>
<A HREF="GuideHome.html">[TOP]</A>
<A HREF="Guide02.html">[PREVIOUS]</A>
<A HREF="Guide04.html">[NEXT]</A>
<A HREF="GuideHome.html#ToC">[CONTENTS]</A>
<ADDRESS>
<A HREF="Guide.Home#author">JK</A>
</ADDRESS>
</BODY>
</HTML>
```

Figure 3.5 shows what Chapter 3 page (minus a lot of the middle) looks like in Mosaic.

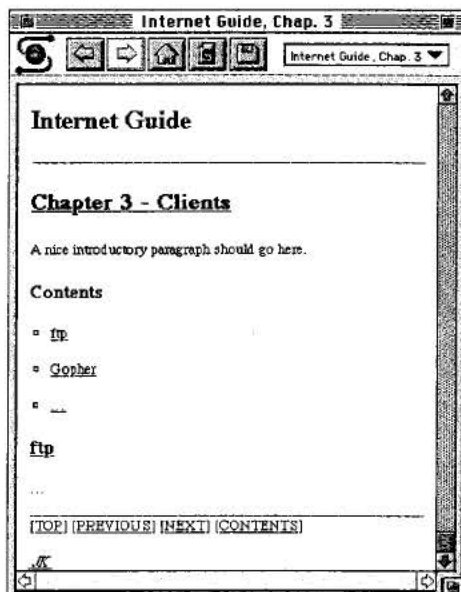


Figure 3.5: Home page for a chapter