A Beginner's Guide to HTML

Primer intro | Full-length version | Printable version | Part 1 | Part 2 | Part 3 | PDF version

This is a primer for producing documents in HTML, the hypertext markup language used on the World Wide Web. This guide is intended to be an introduction to using HTML and creating files for the Web. Links are provided to additional information. You should also check your local bookstore; there are many volumes about the Web and HTML that could be useful.

Getting Started

Terms to Know

WWW
World Wide Web
Web
World Wide Web
SGML
Standard Generalized Markup Language–a standard for describing markup languages
DTD
Document Type Definition–this is the formal specification of a markup language, written using SGML
HTML
HyperText Markup Language–HTML is an SGML DTD
In practical terms, HTML is a collection of platform-independent styles (indicated by markup tags) that define the various components of a World Wide Web document. HTML was invented by Tim Berners-Lee while at CERN, the European Laboratory for Particle Physics in Geneva.

What Isn't Covered

This primer assumes that you:

- know how to use NCSA Mosaic or some other Web browser
- have a general understanding of how Web servers and client browsers work
- have access to a Web server (or that you want to produce HTML documents for personal use in local-viewing mode)

HTML Version

This guide reflects the most current specification–HTML Version 4.0– plus some additional features that have been widely and consistently implemented in browsers. Future versions and new features for HTML are under development.

HTML Documents

What an HTML Document Is

HTML documents are plain-text (also known as ASCII) files that can be created using any text editor (e.g., Emacs or vi on UNIX machines; SimpleText on a Macintosh; Notepad on a Windows machine). You can also use word-processing software if you remember to save your document as "text only with line breaks".

HTML Editors

Some WYSIWYG editors are also available (e.g., Claris Home Page or Adobe PageMill, both for Windows and Macintosh). You may wish to try one of them after you learn some of the basics of HTML tagging. WYSIWYG is an acronym for "what you see is what you get"; it means that you design your HTML document visually, as if you were using a word processor, instead of writing the markup tags in a plain-text file and imagining what the resulting page will look like. It is useful to know enough HTML to code a document before you determine the usefulness of a WYSIWYG editor, in case you want to add HTML features that your editor doesn't support. If you haven't already selected your software, refer to Tom Magliery's online listing of HTML editors (organized by platform) to help you in your search for appropriate software.

Getting Your Files on a Server

If you have access to a Web server at school or work, contact your *webmaster* (the individual who maintains the server) to see how you can get your files on the Web. If you do not have access to a server at work or school, check to see if your community operates a *FreeNet*, a community-based network that provides free access to the Internet. Lacking a FreeNet, you may need to contact a local Internet provider that will post your files on a server for a fee. (Check your local newspaper for advertisements or with your Chamber of Commerce for the names of companies.)

Tags Explained

An *element* is a fundamental component of the structure of a text document. Some examples of elements are heads, tables, paragraphs, and lists. Think of it this way: you use HTML tags to mark the elements of a file for your browser. Elements can contain plain text, other elements, or both.

To denote the various elements in an HTML document, you use *tags*. HTML tags consist of a left angle bracket (<), a tag name, and a right angle bracket (>). Tags are usually paired (e.g., `<H1>` and `</H1>`) to start and end the tag instruction. The end tag looks just like the start tag except a slash (/) precedes the text within the brackets. HTML tags are listed below.

Some elements may include an *attribute*, which is additional information that is included inside the start tag. For example, you can specify the alignment of images (top, middle, or bottom) by including the appropriate attribute with the image source HTML code. Tags that have optional attributes are noted below.

NOTE: *HTML is not case sensitive.* `<title>` is equivalent to `<TITLE>` or `<TiTlE>`. There are a few exceptions noted in Escape Sequences below. Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it will simply ignore it. Any text placed between a pair of unknown tags will still be displayed, however.

The Minimal HTML Document

Every HTML document should contain certain standard HTML tags. Each document consists of head and body text. The head contains the title, and the body contains the actual text that is made up of paragraphs, lists, and other elements. Browsers expect specific information because they are programmed according to HTML and SGML specifications. Required elements are shown in this sample bare-bones document:

```
<html>
  <head>
    <TITLE>A Simple HTML Example</TITLE>
  </head>
  <body>
    <H1>HTML is Easy To Learn</H1>
    <P>Welcome to the world of HTML.
      This is the first paragraph. While short it is
      still a paragraph!</P>
    <P>And this is the second paragraph.</P>
  </body>
</html>
```

The required elements are the <html>, <head>, <title>, and <body> tags (and their corresponding end tags). Because you should include these tags in each file, you might want to create a template file with them. (Some browsers will format your HTML file correctly even if these tags are not included. But some browsers won't! So make sure to include them.)

Click to see the formatted version of the example. A longer example is also available but you should read through the rest of the guide before you take a look. This longer-example file contains tags explained in the next section.

A Teaching Tool

To see a copy of the file that your browser reads to generate the information in your current window, select View Source (or the equivalent) from the browser menu. (Most browsers have a "View" menu under which this command is listed.) The file contents, with all the HTML tags, are displayed in a new window.

This is an excellent way to see how HTML is used and to learn tips and constructs. Of course, the HTML might not be technically correct. Once you become familiar with HTML and check the many online and hard-copy references on the subject, you will learn to distinguish between "good" and "bad" HTML.

Remember that you can save a source file with the HTML codes and use it as a template for one of your Web pages or modify the format to suit your purposes.

Markup Tags

HTML

This element tells your browser that the file contains HTML-coded information. The file extension `.html` also indicates this an HTML document and must be used. (If you are restricted to 8.3 filenames (e.g., `LeeHome.htm`, use only `.htm` for your extension.)

HEAD

The head element identifies the first part of your HTML-coded document that contains the title. The title is shown as part of your browser's window (see below).

TITLE

The title element contains your document title and identifies its content in a global context. The title is typically displayed in the title bar at the top of the browser window, but not inside the window itself. The title is also what is displayed on someone's hotlist or bookmark list, so choose something descriptive, unique, and relatively short. A title is also used to identify your page for search engines (such as Google, HotBot or Infoseek).

For example, you might include a shortened title of a book along with the chapter contents: *NCSA Mosaic Guide (Windows): Installation*. This tells the software name, the platform, and the chapter contents, which is more useful than simply calling the document *Installation*. Generally you should keep your titles to 64 characters or fewer.

BODY

The second–and largest–part of your HTML document is the body, which contains the content of your document (displayed within the text area of your browser window). The tags explained below are used within the body of your HTML document.

Headings

HTML has six levels of headings, numbered 1 through 6, with 1 being the largest. Headings are typically displayed in larger and/or bolder fonts than normal body text. The first heading in each document should be tagged `<H1>`. The syntax of the heading element is:
`<Hy>`*Text of heading* `</Hy>`
where *y* is a number between 1 and 6 specifying the level of the heading. Do not skip levels of headings in your document. For example, don't start with a level-one heading (<H1>) and then next use a level-three (<H3>) heading.

Paragraphs

Unlike documents in most word processors, carriage returns in HTML files aren't significant. In fact, any amount of *whitespace* – including spaces, linefeeds, and carriage returns – are automatically compressed into a single space when your HTML document is displayed in a browser. So you don't have to worry

about how long your lines of text are. Word wrapping can occur at any point in your source file without affecting how the page will be displayed.

In the bare-bones example shown in the Minimal HTML Document section, the first paragraph is coded as

```
<P>Welcome to the world of HTML.
  This is the first paragraph.
  While short it is
  still a paragraph!</P>
```

In the source file there is a line break between the sentences. A Web browser ignores this line break and starts a new paragraph only when it encounters another <P> tag.

Important: You must indicate paragraphs with <P> elements. A browser ignores any indentations or blank lines in the source text. Without <P> elements, the document becomes one large paragraph. (One exception is text tagged as "preformatted," which is explained below.) For example, the following would produce identical output as the first bare-bones HTML example:

```
<H1>Level-one heading</H1>
<P>Welcome to the world of HTML. This is the
  first paragraph. While short it is still a
  paragraph! </P> <P>And this is the second paragraph.</P>
```

To preserve readability in HTML files, put headings on separate lines, use a blank line or two where it helps identify the start of a new section, and separate paragraphs with blank lines (in addition to the <P> tags). These extra spaces will help you when you edit your files (but your browser will ignore the extra spaces because it has its own set of rules on spacing that do not depend on the spaces you put in your source file).

NOTE: The </P> closing tag may be omitted. This is because browsers understand that when they encounter a <P> tag, it means that the previous paragraph has ended. However, since HTML now allows certain attributes to be assigned to the <P> tag, it's generally a good idea to include it.

Using the <P> and </P> as a paragraph container means that you can center a paragraph by including the ALIGN=*alignment* attribute in your source file.

```
<P ALIGN=CENTER>
  This is a centered paragraph.
  [See the formatted version below.]</P>
```

This is a centered paragraph.

It is also possible to align a paragraph to the right instead, by including the ALIGN=RIGHT attribute. ALIGN=LEFT is the default alignment; if no ALIGN attribute is included, the paragraph will be left-aligned.

Lists

HTML supports unnumbered, numbered, and definition lists. You can nest lists too, but use this feature sparingly because too many nested items can get difficult to follow.

Unnumbered Lists

To make an unnumbered, bulleted list,

1. start with an opening list <UL> (for unnumbered list) tag,
2. enter the <LI> (list item) tag followed by the individual item; no closing </LI> tag is needed,
3. end the entire list with a closing list </UL> tag.

Below is a sample three-item list:

```
<UL>
  <LI> apples
  <LI> bananas
  <LI> grapefruit
</UL>
```

The output is:

- apples
- bananas
- grapefruit

The <LI> items can contain multiple paragraphs. Indicate the paragraphs with the <P> paragraph tags.

Numbered Lists

A numbered list (also called an *ordered list*, from which the tag name derives) is identical to an unnumbered list, except it uses <OL> instead of <UL>. The items are tagged using the same <LI> tag. The following HTML code:

```
<OL>
  <LI> oranges
  <LI> peaches
  <LI> grapes
</OL>
```

produces this formatted output:

1. oranges
2. peaches
3. grapes

Definition Lists

A definition list (coded as <DL>) usually consists of alternating a *definition term* (coded as <DT>) and a definition definition (coded as <DD>). Web browsers generally format the definition on a new line and indent it. The following is an example of a definition list:

```
<DL>
  <DT> NCSA
    <DD> NCSA, the National Center for Supercomputing
      Applications, is located on the campus of the
      University of Illinois at Urbana-Champaign.
  <DT> Cornell Theory Center
    <DD> CTC is located on the campus of Cornell
      University in Ithaca, New York.
</DL>
```

The output looks like:

NCSA
: NCSA, the National Center for Supercomputing Applications, is located on the campus of the University of Illinois at Urbana-Champaign.

Cornell Theory Center
: CTC is located on the campus of Cornell University in Ithaca, New York.

The <DT> and <DD> entries can contain multiple paragraphs (indicated by <P> paragraph tags), lists, or other definition information.

The COMPACT attribute can be used routinely in case your definition terms are very short. If, for example, you are showing some computer options, the options may fit on the same line as the start of the definition.

```
<DL COMPACT>
  <DT> -i
    <DD>invokes NCSA Mosaic for Microsoft Windows
      using the initialization file defined in the path
  <DT> -k
    <DD>invokes NCSA Mosaic for Microsoft Windows in
      kiosk mode
</DL>
```

The output looks like:

-i
: invokes NCSA Mosaic for Microsoft Windows using the initialization file defined in the path.

-k
: invokes NCSA Mosaic for Microsoft Windows in kiosk mode.

Nested Lists

Lists can be nested. You can also have a number of paragraphs, each containing a nested list, in a single list item. Here is a sample nested list:

```
<UL>
  <LI> A few New England states:
    <UL>
      <LI> Vermont
      <LI> New Hampshire
      <LI> Maine
    </UL>
  <LI> Two Midwestern states:
    <UL>
      <LI> Michigan
      <LI> Indiana
    </UL>
</UL>
```

The nested list is displayed as

- A few New England states:
  - Vermont
  - New Hampshire
  - Maine
- Two Midwestern states:
  - Michigan
  - Indiana

Preformatted Text

Use the<PRE> tag (which stands for "preformatted") to generate text in a fixed-width font. This tag also makes spaces, new lines, and tabs significant – multiple spaces are displayed as multiple spaces, and lines break in the same locations as in the source HTML file. This is useful for program listings, among other things. For example, the following lines:

```
<PRE>
  #!/bin/csh
  cd $SCR
  cfs get mysrc.f:mycfsdir/mysrc.f
  cfs get myinfile:mycfsdir/myinfile
  fc -02 -o mya.out mysrc.f
  mya.out
  cfs save myoutfile:mycfsdir/myoutfile
  rm *
</PRE>
```

display as:

```
  #!/bin/csh
  cd $SCR
  cfs get mysrc.f:mycfsdir/mysrc.f
  cfs get myinfile:mycfsdir/myinfile
  fc -02 -o mya.out mysrc.f
  mya.out
  cfs save myoutfile:mycfsdir/myoutfile
  rm *
```

The <PRE> tag can be used with an optional WIDTH attribute that specifies the maximum number of characters for a line. WIDTH also signals your browser to choose an appropriate font and indentation for the text.

Hyperlinks can be used within <PRE> sections. You should avoid using other HTML tags within <PRE> sections, however. Note that because <, >, and & have special meanings in HTML, you must use their escape sequences (<, >, and &, respectively) to enter these characters. See the section Escape Sequences for more information.

Extended Quotations

Use the <BLOCKQUOTE> tag to include lengthy quotations in a separate block on the screen. Most browsers generally change the margins for the quotation to separate it from surrounding text. In the example:

```
<P>Omit needless words.</P>
<BLOCKQUOTE>
  <P>Vigorous writing is concise. A sentence should
    contain no unnecessary words, a paragraph no unnecessary
    sentences, for the same reason that a drawing should have
    no unnecessary lines and a machine no unnecessary parts.
  </P>
  <P>&mdash; William Strunk, Jr., 1918 </P>
</BLOCKQUOTE>
```

the result is:

Omit needless words.

> Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts.
>
> — William Strunk, Jr., 1918

Forced Line Breaks / Postal Addresses

The <BR> tag forces a line break with no extra (white) space between lines. Using <P> elements for short lines of text such as postal addresses results in unwanted additional white space. For example, with

```
National Center for Supercomputing Applications<BR>
605 East Springfield Avenue<BR>
Champaign, Illinois 61820-5518<BR>
```

the output is:
National Center for Supercomputing Applications
605 East Springfield Avenue
Champaign, Illinois 61820-5518

Horizontal Rules

The <HR> tag produces a horizontal line the width of the browser window. A horizontal rule is useful to separate major sections of your document. You can vary a rule's size (thickness) and width (the percentage of the window covered by the rule). Experiment with the settings until you are satisfied with the presentation. For example:

```
<HR SIZE=4 WIDTH="50%">
```

displays as:

Character Formatting

HTML has two types of styles for individual words or sentences: logical and physical. *Logical styles* tag text according to its meaning, while *physical styles* indicate the specific appearance of a section. For example, in the preceding sentence, the words "logical styles" was tagged as "emphasis." The same effect (formatting those words in italics) could have been achieved via a different tag that tells your browser to "put these words in italics."

Logical Versus Physical Styles

If physical and logical styles produce the same result on the screen, why are there both?

In the ideal SGML universe, content is divorced from presentation. Thus SGML tags a level-one heading as a level-one heading, but does not specify that the level-one heading should be, for instance, 24-point bold Times centered. The advantage of this approach (it's similar in concept to style sheets in many word processors) is that if you decide to change level-one headings to be 20-point left-justified Helvetica, all you have to do is change the definition of the level-one heading in your Web browser. Indeed, many browsers today let you define how you want the various HTML tags rendered on-screen using what are called *cascading style sheets*, or CSS. CSS is more advanced than HTML, though, and will not be covered in this Primer. (You can learn more about CSS at the World Wide Web Consortium CSS site.)

Another advantage of logical tags is that they help enforce consistency in your documents. It's easier to tag something as `<H1>` than to remember that level-one headings are 24-point bold Times centered or whatever. For example, consider the `<STRONG>` tag. Most browsers render it in bold text. However, it is possible that a reader would prefer that these sections be displayed in red instead. (This is possible using a local cascading style sheet on the reader's own computer.) Logical styles offer this flexibility.

Of course, if you want something to be displayed in italics (for example) and do not want a browser's setting to display it differently, you should use physical styles. Physical styles, therefore, offer consistency in that something you tag a certain way will always be displayed that way for readers of your document.

Try to be consistent about which type of style you use. If you tag with physical styles, do so throughout a document. If you use logical styles, stick with them within a document. Keep in mind that future releases of HTML might not support certain logical styles, which could mean that browsers will not display your logical-style coding. (For example, the `<DFN>` tag – short for "definition", and typically displayed in italics – is not widely supported and will be ignored if the reader's browser does not understand it.)

Logical Styles

`<DFN>`
: for a word being defined. Typically displayed in italics. (*NCSA Mosaic* is a World Wide Web browser.)

`<EM>`
: for emphasis. Typically displayed in italics. (*Consultants cannot reset your password unless you call the help line.*)

`<CITE>`
: for titles of books, films, etc. Typically displayed in italics. (*A Beginner's Guide to HTML*)

`<CODE>`
: for computer code. Displayed in a fixed-width font. (The `<stdio.h>` header file)

`<KBD>`
: for user keyboard entry. Typically displayed in plain fixed-width font. (Enter `passwd` to change your password.)

`<SAMP>`
for a sequence of literal characters. Displayed in a fixed-width font. (`Segmentation fault: Core dumped.`)
`<STRONG>`
for strong emphasis. Typically displayed in bold. (**NOTE:** Always check your links.)
`<VAR>`
for a variable, where you will replace the variable with specific information. Typically displayed in italics. (`rm` *filename* deletes the file.)

Physical Styles

`<B>`
bold text
`<I>`
italic text
`<TT>`
typewriter text, e.g. fixed-width font.

Escape Sequences (a.k.a. Character Entities)

Character entities have two functions:

- escaping special characters
- displaying other characters not available in the plain ASCII character set (primarily characters with diacritical marks)

Three ASCII characters–the left angle bracket (<), the right angle bracket (>), and the ampersand (&)–have special meanings in HTML and therefore cannot be used "as is" in text. (The angle brackets are used to indicate the beginning and end of HTML tags, and the ampersand is used to indicate the beginning of an escape sequence.) Double quote marks may be used as-is but a character entity may also be used ("). To use one of the three characters in an HTML document, you must enter its *escape sequence* instead:

`&lt;`
the escape sequence for <
`&gt;`
the escape sequence for >
`&amp;`
the escape sequence for &

Additional escape sequences support accented characters, such as:

`&ouml;`
a lowercase o with an umlaut: ö
`&ntilde;`
a lowercase n with a tilde: ñ
`&Egrave;`
an uppercase E with a grave accent: È

You can substitute other letters for the *o*, *n*, and *E* shown above. Visit the World Wide Web Consortium for a complete list of special characters.

NOTE: Unlike the rest of HTML, the escape sequences are case sensitive. You cannot, for instance, use `&LT;` instead of `&lt;`.

Linking

The chief power of HTML comes from its ability to link text and/or an image to another document or section of a document. A browser highlights the identified text or image with color and/or underlines to indicate that it is a *hypertext link* (often shortened to *hyperlink* or just *link*). HTML's single hypertext-related tag is `<A>`, which stands for *anchor*. To include an anchor in your document:

1. start the anchor with `<A` (include a space after the `A`),
2. specify the document you're linking to by entering the parameter `HREF="`*filename*`"`, followed by a closing right angle bracket (`>`),
3. enter the text that will serve as the hypertext link in the current document,
4. enter the ending anchor tag: `</A>` (no space is needed before the end anchor tag).

Here is a sample hypertext reference in a file called `US.html`:

```
<A HREF="MaineStats.html">Maine</A>
```

This entry makes the word *Maine* the hyperlink to the document `MaineStats.html`, which is in the same directory as the first document.

Relative Pathnames Versus Absolute Pathnames

You can link to documents in other directories by specifying the *relative path* from the current document to the linked document. For example, a link to a file `NYStats.html` located in the subdirectory `AtlanticStates` would be:

```
<A HREF="AtlanticStates/NYStats.html">New York</A>
```

These are called *relative links* because you are specifying the path to the linked file relative to the location of the current file. You can also use the absolute pathname (the complete URL) of the file, but relative links are more efficient in accessing a server. They also have the advantage of making your documents more "portable" – for instance, you can create several web pages in a single folder on your local computer, using relative links to hyperlink one page to another, and then upload the entire folder of web pages to your web server. The pages on the server will then link to other pages on the server, and the copies on your hard drive will still point to the other pages stored there.

It is important to point out that UNIX is a case-sensitive operating system where filenames are concerned, while DOS and the MacOS are not. For instance, on a Macintosh, "DOCUMENT.HTML", "Document.HTML", and "document.html" are all the same name. If you make a relative hyperlink to "DOCUMENT.HTML", and the file is actually named "document.html", the link will still be valid. But if you upload all your pages to a UNIX web server, the link will no longer work. Be sure to check your filenames before uploading.

Pathnames use the standard UNIX syntax. The UNIX syntax for the parent directory (the directory that contains the current directory) is "..". (For more information consult a beginning UNIX reference text such as *Learning the UNIX Operating System* from O'Reilly and Associates, Inc.)

If you were in the `NYStats.html` file and were referring to the original document `US.html`, your link would look like this:

```
<A HREF="../US.html">United States</A>
```

In general, you should use relative links whenever possible because:

1. it's easier to move a group of documents to another location (because the relative path names will still be valid)
2. it's more efficient connecting to the server
3. there is less to type

However, use absolute pathnames when linking to documents that are not directly related. For example, consider a group of documents that comprise a user manual. Links within this group should be relative links. Links to other documents (perhaps a reference to related software) should use absolute pathnames instead. This way if you move the user manual to a different directory, none of the links would have to be updated.

URLs

The World Wide Web uses Uniform Resource Locators (URLs) to specify the location of files on other servers. A URL includes the type of resource being accessed (e.g., Web, gopher, FTP), the address of the server, and the location of the file. The syntax is:

scheme`://`*host.domain* `[:`*port*`]` `/`*path*`/` *filename*

where *scheme* is one of

`file`
: a file on your local system

`ftp`
: a file on an anonymous FTP server

`http`
: a file on a World Wide Web server

`gopher`
: a file on a Gopher server

`WAIS`
: a file on a WAIS server

`news`
: a Usenet newsgroup

`telnet`
: a connection to a Telnet-based service

The *port* number can generally be omitted. (That means unless someone tells you otherwise, leave it out.) For example, to include a link to this primer in your document, enter:

```
<A HREF="http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html">
NCSA's Beginner's Guide to HTML</A>
```

This entry makes the text *NCSA's Beginner's Guide to HTML* a hyperlink to this document. There is also a `mailto` scheme, used to hyperlink email addresses, but this scheme is unique in that it uses only a colon (`:`) instead of `://` between the scheme and the address. You can read more about `mailto` below. For more information on URLs, refer to:

- *WWW Names and Addresses, URIs, URLs, URNs*
- *A Beginner's Guide to URLs*

Links to Specific Sections

Anchors can also be used to move a reader to a *particular section* in a document (either the same or a different document) rather than to the top, which is the default. This type of an anchor is commonly called a *named anchor* because to create the links, you insert HTML names within the document.

This guide is a good example of using named anchors in one document. The guide is constructed as one document to make printing easier. But as one (long) document, it can be time-consuming to move through when all you really want to know about is one bit of information about HTML. Internal hyperlinks are used to create a "table of contents" at the top of this document. These hyperlinks move you from one location in the document to another location in the same document.

(Go to the top of this document and then click on the *Links to Specific Sections* hyperlink in the table of contents. You will wind up back here.)

You can also link to a specific section in another document. That information is presented first because understanding that helps you understand linking within one document.

Links Between Sections of Different Documents

Suppose you want to set a link from document A (documentA.html) to a specific section in another document (MaineStats.html). Enter the HTML coding for a link to a named anchor:

```
documentA.html:
In addition to the many state parks, Maine is also home to
<a href="MaineStats.html#ANP">Acadia National Park</a>.
```

Think of the characters after the hash (#) mark as a tab within the MaineStats.html file. This tab tells your browser what should be displayed at the top of the window when the link is activated. In other words, the first line in your browser window should be the Acadia National Park heading. Next, create the *named anchor* (in this example "ANP") in MaineStats.html:

```
<H2><A NAME="ANP">Acadia National Park</a></H2>
```

With both of these elements in place, you can bring a reader directly to the Acadia reference in MaineStats.html.

NOTE: You cannot make links to specific sections within a different document unless either you have write permission to the coded source of that document or that document already contains in-document named anchors. For example, you could include named anchors to this primer in a document you are writing because there are named anchors in this guide (use View Source in your browser to see the coding). But if this document did *not* have named anchors, you could not make a link to a specific section because you cannot edit the original file on NCSA's server.

Links to Specific Sections within the Current Document

The technique is the same except the filename is *omitted*. For example, to link to the ANP anchor from within MaineStats, enter:

```
...More information about
<A HREF="#ANP">Acadia National Park</a>
is available elsewhere in this document.
```

Be sure to include the <A NAME=> tag at the place in your document where you want the link to jump to (<A NAME="ANP">Acadia National Park). Named anchors are particularly useful when you think readers will print a document in its entirety or when you have a lot of short information you want to place online in one file.

Mailto

You can make it easy for a reader to send electronic mail to a specific person or mail alias by including the mailto attribute in a hyperlink. The format is:

```
<A HREF="mailto:emailinfo@host">Name</a>
```

For example, enter:

```
<A HREF="mailto:pubs@ncsa.uiuc.edu">
  NCSA Publications Group</a>
```

to create a mail window that is already configured to open a mail window for the NCSA Publications Group alias. (You, of course, will enter another mail address!)

Inline Images

Most Web browsers can display inline images (that is, images next to text) that are in X Bitmap (XBM), GIF, or JPEG format. Other image formats are also being incorporated into Web browsers [e.g., the Portable Network Graphic (PNG) format]. Each image takes additional time to download and slows down the initial display of a document. Carefully select your images and the number of images in a document. To include an inline image, enter:

```
<IMG SRC=ImageName>
```

where *ImageName* is the URL of the image file. The syntax for `<IMG SRC>` URLs is identical to that used in an anchor `HREF`. If the image file is a GIF file, then the filename part of *ImageName* must end with `.gif`. Filenames of X Bitmap images must end with `.xbm`; JPEG image files must end with `.jpg` or `.jpeg`; and Portable Network Graphic files must end with `.png`.

Image Size Attributes

You should include two other attributes on `<IMG>` tags to tell your browser the size of the images it is downloading with the text. The `HEIGHT` and `WIDTH` attributes let your browser set aside the appropriate space (in pixels) for the images as it downloads the rest of the file. (You can get the pixel size from your image-processing software, such as Adobe Photoshop. Some browsers will also display the dimensions of an image file in the title bar if the image is viewed by itself without an enclosing HTML document.) For example, to include a self portrait image in a file along with the portrait's dimensions, enter:

```
<IMG SRC=SelfPortrait.gif HEIGHT=100 WIDTH=65>
```

NOTE: Some browsers use the `HEIGHT` and `WIDTH` attributes to stretch or shrink an image to fit into the allotted space when the image does not exactly match the attribute numbers. Not all browser developers think stretching/shrinking is a good idea, so don't plan on your readers having access to this feature. Check your dimensions and use the correct ones.

Aligning Images

You have some flexibility when displaying images. You can have images separated from text and aligned to the left or right or centered. Or you can have an image aligned with text (here: `<IMG SRC="hotlist.gif" ALT="[HOTLIST]" ALIGN=RIGHT>`). Try several possibilities to see how your information looks best.

Aligning Text with an Image


By default the bottom of an image is aligned with the following text, as shown in this paragraph. You can align images to the top or center of a paragraph using the `ALIGN=` attributes `TOP` and `MIDDLE`.


This text is aligned with the top of the image (`<IMG SRC="hotlist.gif" ALT="[HOTLIST]" ALIGN=TOP>`). Notice how the browser aligns only one line and then jumps to the bottom of the image for the rest of the text.

 And this text is centered on the image (`<IMG SRC="hotlist.gif" ALT="[HOTLIST]" ALIGN=MIDDLE>`). Again, only one line of text is centered; the rest is below the image.

Images without Text

To display an image without any associated text (e.g., your organization's logo), make it a separate paragraph. Use the paragraph `ALIGN=` attribute to center the image or adjust it to the right side of the window as shown below:

```
<p ALIGN=CENTER>
  <IMG SRC="hotlist.gif" ALT="[HOTLIST]">
</p>
```

which results in:

The image is centered; this paragraph starts below it and left justified.

Alternate Text for Images

Some World Wide Web browsers – primarily the text-only browsers such as Lynx – cannot display images. Some users turn off image loading even if their software can display images (especially if they are using a modem or have a slow connection). HTML provides a mechanism to tell readers what they are missing on your pages if they can't load images. The `ALT` attribute lets you specify text to be displayed instead of an image. For example:

```
<IMG SRC="UpArrow.gif" ALT="Up">
```

where `UpArrow.gif` is the picture of an upward pointing arrow. With graphics-capable viewers that have image-loading turned on, you see the up arrow graphic. With a text-only browser or if image-loading is turned off, the word *Up* is shown in your window in place of the image. You should try to include alternate text for each image you use in your document, which is a courtesy for your readers – or, for users who might be visually impaired, a necessity.

Images as Hyperlinks

Inline images can be used as hyperlinks just like plain text. This HTML code:

```
<A HREF="hotlist.html"><IMG SRC="hotlist.gif" ALT="[HOTLIST]"></A>
```

produces the following result:

(Note that this link doesn't actually go anywhere.) The blue border that surrounds the image indicates that it's a clickable hyperlink. You may not always want this border to be displayed, though. In this case you can use the BORDER attribute of the IMG tag to make the image appear as normal. Adding the BORDER attribute and setting it to zero:

```
<A HREF="hotlist.html"><IMG SRC="hotlist.gif" BORDER=0 ALT="[HOTLIST]"></A>
```

produces the following result:

The BORDER attribute can also be set to non-zero values, whether or not the image is used as a hyperlink. In this case, the border will appear using the default text color for the web page. For instance, if you wanted to give your image a plain black border to help it stand out on the page, you might try this:

```
<IMG SRC="hotlist.gif" BORDER=6 ALT="[HOTLIST]">
```

And get the following result:

Background Graphics

Newer versions of Web browsers can load an image and use it as a background when displaying a page. Some people like background images and some don't. In general, if you want to include a background, make sure your text can be read easily when displayed on top of the image. Background images can be a texture (linen finished paper, for example) or an image of an object (a logo possibly). You create the background image as you do any image.

However you only have to create a small piece of the image. Using a feature called tiling, a browser takes the image and repeats it across and down to fill your browser window. In sum you generate one image, and the browser replicates it enough times to fill your window. This action is automatic when you use the background tag shown below. The tag to include a background image is included in the <BODY> statement as an attribute:

```
<BODY BACKGROUND="filename.gif">
```

Background Color

By default browsers display text in black on a gray background. However, you can change both elements if you want. Some HTML authors select a background color and coordinate it with a change in the color of the text.

Always preview changes like this to make sure your pages are readable. (For example, many people find red text on a black background difficult to read!) In general, try to avoid using high-contrast images or images that use the color of your text anywhere within the graphic. You change the color of text, links, visited links, and active links (links that are currently being clicked on) using further attributes of the <BODY> tag. For example:

```
<BODY BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#9690CC">
```

This creates a window with a black background (BGCOLOR), white text (TEXT), and silvery hyperlinks (LINK). The six-digit number and letter combinations represent colors by giving their RGB (red, green, blue) value. The six digits are actually three two-digit numbers in sequence, representing the amount of red, green, or blue as a hexadecimal value in the range 00-FF. For example, 000000 is black (no color at all), FF0000 is bright red, 0000FF is bright blue, and FFFFFF is white (fully saturated with all three colors).

These number and letter combinations are generally rather cryptic. Fortunately an online resource is available to help you track down the combinations that map to specific colors and there is software available for you to do this on your workstation:

- hIdaho ColorCenter

For some basic colors – typically those in the standard sixteen-color Windows 3.1 palette – you can also use the name of the color instead of the corresponding RGB value. For example, "black", "red", "blue", and "cyan" are all valid for use in place of RGB values. However, while not all browsers will understand all color names, any browser that can display colors will understand RGB values, so use them whenever possible.

External Images, Sounds, and Animations

You may want to have an image open as a separate document when a user activates a link on either a word or a smaller, inline version of the image included in your document. This is called an external image, and it is useful if you do not wish to slow down the loading of the main document with large inline images. To include a reference to an external image, enter:

```
<A HREF="MyImage.gif">link anchor</A>
```

You can also use a smaller image as a link to a larger image. Enter:

```
<A HREF="LargerImage.gif"><IMG SRC="SmallImage.gif"></A>
```

The reader sees the `SmallImage.gif` image and clicks on it to open the `LargerImage.gif` file. Use the same syntax for links to external animations and sounds. The only difference is the file extension of the linked file. For example,

```
<A HREF="AdamsRib.mov">link anchor</A>
```

specifies a link to a QuickTime movie. Some common file types and their extensions are:

plain text
`.txt`
HTML document
`.html`
GIF image
`.gif`
TIFF image
`.tiff`
X Bitmap image
`.xbm`
JPEG image
`.jpg` or `.jpeg`
PostScript file
`.ps`
AIFF sound file
`.aiff`
AU sound file
`.au`
WAV sound file
`.wav`
QuickTime movie
`.mov`
MPEG movie
`.mpeg` or `.mpg`

Keep in mind your intended audience and their access to software. Most UNIX workstations, for instance, cannot view QuickTime movies.

Tables

Before HTML tags for tables were finalized, authors had to carefully format their tabular information within <PRE> tags, counting spaces and previewing their output. Tables are very useful for presentation of tabular information as well as a boon to creative HTML authors who use the table tags to present their regular Web pages. (Check out the NCSA home page for an excellent example of using tables to control page layout.)

Think of your tabular information in light of the coding explained below. A table has heads where you explain what the columns/rows include, rows for information, cells for each item. In the following table, the first column contains the header information, each row explains an HTML table tag, and each cell contains a paired tag or an explanation of the tag's function.

Table Elements

Element	Description
<TABLE> ... </TABLE>	defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border.
<CAPTION> ... </CAPTION>	defines the caption for the title of the table. The default position of the title is centered at the top of the table. The attribute ALIGN=BOTTOM can be used to position the caption below the table. **NOTE:** Any kind of markup tag can be used in the caption.
<TR> ... </TR>	specifies a table row within a table. You may define default attributes for the entire row: ALIGN (LEFT, CENTER, RIGHT) and/or VALIGN (TOP, MIDDLE, BOTTOM). See Table Attributes at the end of this table for more information.
<TH> ... </TH>	defines a table header cell. By default the text in this cell is bold and centered. Table header cells may contain other attributes to determine the characteristics of the cell and/or its contents. See Table Attributes at the end of this table for more information.
<TD> ... </TD>	defines a table data cell. By default the text in this cell is aligned left and centered vertically. Table data cells may contain other attributes to determine the characteristics of the cell and/or its contents. See Table Attributes at the end of this table for more information.

Table Attributes

NOTE: Attributes defined within <TH> ... </TH> or <TD> ... </TD> cells override the default alignment set in a <TR> ... </TR>. The WIDTH attribute is deprecated.

Attribute	Description
ALIGN (LEFT, CENTER, RIGHT)	Horizontal alignment of a cell.
VALIGN (TOP, MIDDLE, BOTTOM)	Vertical alignment of a cell.
COLSPAN=*n*	The number (*n*) of columns a cell spans.
ROWSPAN=*n*	The number (*n*) of rows a cell spans.
NOWRAP	Turn off word wrapping within a cell.

General Table Format

The general format of a table looks like this:

```
<TABLE>
<!-- start of table definition -->
<CAPTION> caption contents </CAPTION>
<!-- caption definition -->
<TR>
<!-- start of header row definition -->
    <TH WIDTH=40%> first header cell contents </TH>
    <TH WIDTH=60%> last header cell contents </TH>
</TR>
<!-- end of header row definition -->
<TR>
<!-- start of first row definition -->
    <TD> first row, first cell contents </TD>
    <TD> first row, last cell contents </TD>
</TR>
<!-- end of first row definition -->
<TR>
<!-- start of last row definition -->
    <TD> last row, first cell contents </TD>
    <TD> last row, last cell contents </TD>
</TR>
<!-- end of last row definition -->
</TABLE>
<!-- end of table definition -->
```

You can cut-and-paste the above code into your own HTML documents, adding new rows or cells as necessary. The above example looks like this when rendered in a browser.

The <TABLE> and </TABLE> tags **must** surround the entire table definition. The first item inside the table is the CAPTION, which is optional. Then you can have any number of rows defined by the <TR> and </TR> tags. Within a row you can have any number of cells defined by the <TD>...</TD> or <TH>...</TH> tags. Each row of a table is, essentially, formatted independently of the rows above and below it. This lets you easily display tables like the one above with a single cell, such as Table Attributes, spanning columns of the table.

Tables for Nontabular Information

Some HTML authors use tables to present nontabular information. For example, because links can be included in table cells, some authors use a table with no borders to create "one" image from separate images. Browsers that can display tables properly show the various images seamlessly, making the created image seem like an *image map* (one image with hyperlinked quadrants). Using table borders with images can create an impressive display as well. Experiment and see what you like.

Fill-out Forms

Web forms let a reader return information to a Web server for some action. For example, suppose you collect names and email addresses so you can email some information to people who request it. For each person who enters his or her name and address, you need some information to be sent and the respondent's particulars added to a data base.

This processing of incoming data is usually handled by a script or program written in Perl or another language that manipulates text, files, and information. If you cannot write a program or script for your incoming information, you need to find someone who can do this for you.

The forms themselves are not hard to code. They follow the same constructs as other HTML tags. What could be difficult is the program or script that takes the information submitted in a form and processes it. Because of the need for specialized scripts to handle the incoming form information, fill-out forms are not discussed in this primer. Check the Additional Online Reference section for more information.

Troubleshooting

Avoid Overlapping Tags

Consider this example of HTML:

```
<B>This is an example of <I>overlapping</B> HTML tags.</I>
```

The word *overlapping* is contained within both the <B> and <I> tags. A browser might be confused by this coding and might not display it the way you intend. The only way to know is to check each popular browser (which is time-consuming and impractical).

In general, avoid overlapping tags. Look at your tags and try pairing them up. Tags (with the obvious exceptions of elements whose end tags may be omitted, such as paragraphs) should be paired without an intervening tag in between. Look again at the example above. You cannot pair the bold tags without another tag in the middle (the first definition tag). Try matching your coding up like this to see if you have any problem areas that should be fixed before you release your files to a server.

Embed Only Anchors and Character Tags

HTML protocol allows you to embed links within other HTML tags:

```
<H1><A HREF="Destination.html">My heading</A></H1>
```

Do not embed HTML tags within an anchor:

```
<A HREF="Destination.html">
  <H1>My heading</H1>
</A>
```

Although most browsers currently handle this second example, the official HTML specifications do not support this construct and your file will probably not work with future browsers. Remember that browsers can be forgiving when displaying improperly coded files. But that forgiveness may not last to the next version of the software! When in doubt, code your files according to the HTML specifications (see For More Information below). Character tags modify the appearance of the text within other elements:

```
<UL>
  <LI><B>A bold list item</B>
  <LI><I>An italic list item</I>
</UL>
```

Avoid embedding other types of HTML element tags. For example, you might be tempted to embed a heading within a list in order to make the font size larger:

```
<UL>
  <LI><H1>A large heading</H1>
  <LI><H2>Something slightly smaller</H2>
</UL>
```

Although some browsers handle this quite nicely, formatting of such coding is unpredictable (because it is undefined). For compatibility with all browsers, avoid these kinds of constructs. (The Netscape <FONT> tag, which lets you specify how large individual characters will be displayed in your window, is not currently part of the official HTML specifications.)

What's the difference between embedding a <B> within a <LI> tag as opposed to embedding a <H1> within a <LI>? Within HTML the semantic meaning of <H1> is that it's the main heading of a document and that it should be followed by the content of the document. Therefore it doesn't make sense to find a <H1> within a list. Character formatting tags also are generally not additive. For example, you might expect that:

```
<B><I>some text</I></B>
```

would produce bold-italic text. On some browsers it does; other browsers interpret only the innermost tag.

Do the Final Steps

Validate Your Code

When you put a document on a Web server, be sure to check the formatting and each link (including named anchors). Ideally you will have someone else read through and comment on your file(s) before you consider a document finished.

You can run your coded files through one of several on-line HTML validation services that will tell you if your code conforms to accepted HTML. If you are not sure your coding conforms to HTML specifications, this can be a useful teaching tool. Fortunately the service lets you select the level of conformance you want for your files (i.e., strict, level 2, level 3). If you want to use some codes that are not officially part of the HTML specifications, this latitude is helpful.

Dummy Images

When an `<IMG SRC>` tag points to an image that does not exist, a dummy image is substituted by your browser software. When this happens during your final review of your files, make sure that the referenced image does in fact exist, that the hyperlink has the correct information in the URL, and that the file permission is set appropriately (world-readable). Then check online again!

Update Your Files

If the contents of a file are static (such as a biography of George Washington), no updating is probably needed. But for documents that are time sensitive or covering a field that changes frequently, **remember to update your documents!**

Updating is particularly important when the file contains information such as a weekly schedule or a deadline for a program funding announcement. Remove out-of-date files or note why something that appears dated is still on a server (e.g., the program requirements will remain the same for the next cycle so the file is still available as an interim reference).

Browsers Differ

Web browsers display HTML elements differently. Remember that not all codes used in HTML files are interpreted by all browsers. Any code a browser does not understand is usually ignored though.

You could spend a lot of time making your file "look perfect" using your current browser. If you check that file using another browser, it will likely display (a little or a lot) differently. Hence these words of advice: code your files using correct HTML. Leave the interpreting to the browsers and hope for the best.

Commenting Your Files

You might want to include comments in your HTML files. Comments in HTML are like comments in a computer program–the text you enter is not used by the browser in any formatting and is not directly viewable by the reader just as computer program comments are not used and are not viewable. The comments are accessible if a reader views the source file, however. Comments such as the name of the person updating a file, the software and version used in creating a file, or the date that a minor edit was made are the norm. To include a comment, enter:

```
<!-- your comments here -->
```

You must include the exclamation mark and the hyphens as shown.

For More Information

This guide is only an **introduction** to HTML, not a comprehensive reference. Below are additional online sources of information. Remember to check a bookstore near you for Web and HTML books.

Style Guides

The following offer advice on how to write "good" HTML:

- *Composing Good HTML*
- W3C's style guide for online hypertext

Other Introductory Documents

These cover similar information as this guide:

- *How to Write HTML Files*
- *Introduction to HTML*
- the Yale Center for Advanced Instructional Media pages
- *The HTML Quick Reference Guide*, which provides a comprehensive listing of HTML codes
- *HTML/CSS Tutorial*

Additional Online References

- Official HTML specification
- A description of SGML, the Standard Generalized Markup Language
- NCSA HTTPd server software information
- Background images
- Forms and the essential scripts explained
- Java, snazzy stuff that is really perking up the Web

Thanks

NCSA acknowledges and thanks the many Web users who have commented on this guide. Thanks also to the NCSA reviewers and contributors as well as the author of the first version of this guide.