A BEGINNER'S GUIDE TO HTML

This is a primer for producing documents in HTML, the markup language used by the World Wide Web.

- For More Information
  - Fill-out Forms
  - Style Guides
  - Other Introductory Documents
  - Additional References

Acronym Expansion

WWW World Wide Web (or Web, for short).

SGML
: Standard Generalized Markup Language -- this is a standard for describing markup languages.

DTD
: Document Type Definition -- this is a specific markup language, written using SGML.

HTML
: HyperText Markup Language -- HTML is a SGML DTD. In practical terms, HTML is a collection of styles (indicated by markup tags) that define the various components of a World Wide Web document.

What This Primer Doesn't Cover

This primer assumes that you have:

- at least a passing knowledge of how to use NCSA Mosaic or some other Web browser
- a general understanding of how Web servers and client browsers work
- access to a Web server for which you would like to produce HTML documents, or that you wish to produce HTML documents for personal use

Creating HTML Documents

HTML documents are in plain (also known as ASCII) text format and can be created using any text editor (e.g., Emacs or vi on UNIX machines). A couple of Web browsers (tkWWW for X Window System machines and CERN's Web browser for NeXT computers) include rudimentary HTML editors in a WYSIWYG environment. There are also some WYSIWIG editors available now (e.g. HotMetal for Sun Sparcstations, HTML Edit for Macintoshes). You may wish to try one of them first before delving into the details of HTML.

You can preview a document in progress with NCSA Mosaic (and some other Web browsers). Open it with the Open Local command under the File menu.

After you edit the source HTML file, save the changes. Return to NCSA Mosaic and Reload the document. The changes are reflected in the on-screen display.

THE MINIMAL HTML DOCUMENT

Here is a bare-bones example of HTML:

```
<TITLE>The simplest HTML example</TITLE>
<H1>This is a level-one heading</H1>
Welcome to the world of HTML.
This is one paragraph.<P>
And this is a second.<P>
```

Click here to see the formatted version of the example.

HTML uses markup tags to tell the Web browser how to display the text. The above example uses:

* the <TITLE> tag (and corresponding </TITLE> tag), which specifies the title of the document
* the <H1> header tag (and corresponding </H1>)
* the <P> paragraph-separator tag

HTML tags consist of a left angle bracket (<), (a ``less than" symbol to mathematicians), followed by name of the tag and closed by a right angular bracket (>). Tags are usually paired, e.g. <H1> and </H1>. The ending tag looks just like the starting tag except a slash (/) precedes the text within the brackets. In the example, <H1> tells the Web browser to start formatting a level-one heading; </H1> tells the browser that the heading is complete.

The primary exception to the pairing rule is the <P> tag. There is no such thing as </P>.

NOTE: HTML is not case sensitive. <title> is equivalent to <TITLE> or <TiTlE>.

Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it just ignores it.

BASIC MARKUP TAGS

Title

Every HTML document should have a title. A title is generally displayed separately from the document and is used primarily for document identification in other contexts (e.g., a WAIS search). Choose about half a dozen words that describe the document's purpose.

In the X Window System and Microsoft Windows versions of NCSA Mosaic, the Document Title field is at the top of the screen just below the pulldown menus. In NCSA Mosaic for Macintosh, text tagged as <TITLE> appears as the window title.

Headings

HTML has six levels of headings, numbered 1 through 6, with 1 being the most prominent. Headings are 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 tag is:

```
<Hy>Text of heading </Hy >
```

where y is a number between 1 and 6 specifying the level of the heading.

For example, the coding for the ``Headings'' section heading above is

```
<H3>Headings</H3>
```

Title versus first heading

In many documents, the first heading is identical to the title. For multipart documents, the text of the first heading should be suitable for a reader who is already browsing related information (e.g., a chapter title), while the title tag should identify the document in a wider context (e.g., include both the book title and the chapter title, although this can sometimes become overly long).

Paragraphs

Unlike documents in most word processors, carriage returns in HTML files aren't significant. Word wrapping can occur at any point in your source file, and multiple spaces are collapsed into a single space. (There are couple of exceptions; space following a <P> or <Hy> tag, for example, is ignored.) Notice that in the bare-bones example, the first paragraph is coded as

```
Welcome to HTML.
This is the first 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 reaches a <P> tag.

Important: You must separate paragraphs with <P>. The browser ignores any indentations or blank lines in the source text. HTML relies almost entirely on the tags for formatting instructions, and without the <P> tags, the document becomes one large paragraph. (The exception is text tagged as ``preformatted,'' which is explained below.) For instance, the following would produce identical output as the first bare-bones HTML example:

```
<TITLE>The simplest HTML example</TITLE><H1>This is a level
one heading</H1>Welcome to the world of HTML. This is one
paragraph.<P>And this is a second.<P>
```

However, to preserve readability in HTML files, headings should be on separate lines, and paragraphs should be separated by blank lines (in addition to the <P> tags).

NCSA Mosaic handles <P> by ending the current paragraph and inserting a blank line.

In HTML+, a successor to HTML currently in development, <P> becomes a ``container'' of text, just as the text of a level-one heading is ``contained'' within<H1> ... </H1>:

```
<P>
This is a paragraph in HTML+.
</P>
```

The difference is that the </P> closing tag can always be omitted. (That is, if a browser sees a <P>, it knows that there must be an implied </P> to end the previous paragraph.) In other words, in HTML+, <P> is a beginning-of-paragraph marker.

The advantage of this change is that you will be able to specify formatting options for a paragraph. For example, in HTML+, you will be able to center a paragraph by coding

```
<P ALIGN=CENTER>
This is a centered paragraph. This is HTML+, so you can't do it yet.
```

This change won't effect any documents you write now, and they will continue to look just the same with HTML+ browsers.

LINKING TO OTHER DOCUMENTS

The chief power of HTML comes from its ability to link regions of text (and also images) to another document. The browser highlights these regions (usually with color and/or underlines) to indicate that they are hypertext links (often shortened to hyperlinks or simply links).

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 . (There's a space after the A.)
2. Specify the document that's being pointed 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>.

Here is an sample hypertext reference:

```
<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. 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 NJStats.html located in the subdirectory AtlanticStates would be:

```
<A HREF="AtlanticStates/NJStats.html">New Jersey</A>
```

These are called relative links. You can also use the absolute pathname of the file if you wish. Pathnames use the standard UNIX syntax.

Relative Links Versus Absolute Pathnames

In general, you should use relative links, because

1. You have less to type.
2. It's easier to move a group of documents to another location, because the relative path names will still be valid.

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 full path names. This way, if you move the user manual to a different directory, none of the links would have to be updated.

Uniform Resource Locator

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., gopher, WAIS), 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, or 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
: an 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, you would use

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

This would make the text ``NCSA's Beginner's Guide to HTML" a hyperlink to this document.

For more information on URLs, look at

* WWW Names and Addresses, URIs, URLs, URNs, written by people at CERN
* A Beginner's Guide to URLs, located on the NCSA Mosaic Help menu

Links to Specific Sections in Other Documents

Anchors can also be used to move to a particular section in a document. Suppose you wish to set a link from document A and a specific section in document B. (Call this file documentB.html.) First you need to set up a named anchor in document B. For example, to set up an anchor named ``Jabberwocky" to document B, enter

```
Here's <A NAME = "Jabberwocky">some text</a>
```

Now when you create the link in document A, include not only the filename, but also the named anchor, separated by a hash mark (#).

```
This is my <A HREF = "documentB.html#Jabberwocky">link</A> to document B.
```

Now clicking on the word ``link" in document A sends the reader directly to the words ``some text" in document B.

Links to Specific Sections Within the Current Document

The technique is exactly the same except the filename is omitted.

For example, to link to the Jabberwocky anchor from within the same file (Document B), use

```
This is <A HREF = "#Jabberwocky">Jabberwocky link</A> from within Document B.
```

Additional Markup Tags

The preceding is sufficient to produce simple HTML documents. For more complex documents, HTML has tags for several types of lists, preformatted sections, extended quotations, character formatting, and other items.

LISTS

HTML supports unnumbered, numbered, and definition lists.

Unnumbered Lists

To make an unnumbered list,

1. Start with an opening list <UL> tag.
2. Enter the <LI> tag followed by the individual item. (No closing </LI> tag is needed.)
3. End with a closing list </UL> tag.

Below an example two-item list:

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

The output is:

```
 * apples
 * bananas
```

The <LI> items can contain multiple paragraphs. Just separate 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 usually consists of alternating a term (abbreviated as DT) and a definition (abbreviated as DD). Web browsers generally format the definition on a new line.

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. NCSA is one of the participants in the
     National MetaCenter for Computational Science and Engineering.
<DT> Cornell Theory Center
<DD> CTC is located on the campus of Cornell University in Ithaca,
     New York. CTC is another participant in the National MetaCenter
     for Computational Science and Engineering.
</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. NCSA is one of the participants in the
       National MetaCenter for Computational Science and Engineering.
Cornell Theory Center
       CTC is located on the campus of Cornell University in Ithaca,
       New York. CTC is another participant in the National MetaCenter
       for Computational Science and Engineering.
```

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

Nested Lists

Lists can be arbitrarily nested, although in practice you probably should limit the nesting to three levels. You can also have a number of paragraphs, each containing a nested list, in a single list item.

An example nested list:

```
<UL>
<LI> A few New England states:
   <UL>
   <LI> Vermont
   <LI> New Hampshire
   </UL>
<LI> One Midwestern state:
   <UL>
   <LI> Michigan
   </UL>
</UL>
```

The nested list is displayed as

```
  * A few New England states:
      + Vermont
      + New Hampshire
  * One Midwestern state:
      + Michigan
```

PREFORMATTED TEXT

Use the <PRE> tag (which stands for ``preformatted") to generate text in a fixed-width font and cause spaces, new lines, and tabs to be significant. (That is, 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. 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 *
```

Hyperlinks can be used within <PRE> sections. You should avoid using other HTML tags within <PRE> sections, however.

Note that because <, >, and & have special meaning in HTML, you have to use their escape sequences (&lt;, &gt;, and &amp;, respectively) to enter these characters. See the section Special Characters for more information.

EXTENDED QUOTATIONS

Use the <BLOCKQUOTE> tag to include quotations in a separate block on the screen. Most browsers generally indent to separate it from surrounding text.

An example:

```
<BLOCKQUOTE>
I still have a dream. It is a dream deeply rooted in the
American dream. <P>
I have a dream that one day this nation will rise up and
live out the true meaning of its creed. We hold these truths
to be self-evident that all men are created equal. <P>
</BLOCKQUOTE>
```

The result is:

> I still have a dream. It is a dream deeply rooted in the American dream.
>
> I have a dream that one day this nation will rise up and live out the true meaning of its creed. We hold these truths to be self-evident that all men are created equal.

ADDRESSES

The <ADDRESS> tag is generally used to specify the author of a document and a means of contacting the author (e.g., an email address). This is usually the last item in a file.

For example, the last line of the online version of this guide is

```
<ADDRESS>
A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu
</ADDRESS>
```

The result is
A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu

NOTE: <ADDRESS> is not used for postal addresses. See ``Forced Line Breaks'' on page 10 to see how to format postal addresses.

Character Formatting

You can code individual words or sentences with special styles. There are two types of styles: logical and physical. Logical styles tag text according to its meaning, while physical styles specify the specific appearance of a section. For example, in the preceding sentence, the words ``logical styles'' was tagged as a ``definition.'' The same effect (formatting those words in italics), could have been achieved via a different tag that specifies merely ``put these words in italics.''

PHYSICAL VERSUS LOGICAL: USE LOGICAL STYLES WHEN POSSIBLE

If physical and logical styles produce the same result on the screen, why are there both? We devolve, for a couple of paragraphs, into the philosophy of SGML, which can be summed in a Zen-like mantra: ``Trust your browser."

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 on the top of a page. 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 the presentation device (i.e., your World Wide Web browser).

The other 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 or whatever. The same is true for character styles. 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. Logical styles offer this flexibility.

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. (Watch out for pickpockets.)

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

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

<KBD> for user keyboard entry. Should be displayed in a bold fixed-width font, but many browsers render it in the plain fixed-width font. (Enter passwd to change your password.)

<SAMP>
for computer status messages. Displayed in a fixed-width font. (Segmentation fault: Core dumped.)

<STRONG>

for strong emphasis. Typically displayed in bold. (Important)

<VAR> for a ``metasyntactic" variable, where the user is to replace the variable with a specific instance. 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.

USING CHARACTER TAGS

To apply a character style,

1. Start with <tag>, where tag is the desired character formatting tag, to indicate the beginning of the tagged text.
2. Enter the tagged text.
3. End the passage with </tag>.

SPECIAL CHARACTERS

Escape Sequences

Four characters of the ASCII character set -- the left angle bracket (<), the right angle bracket (>), the ampersand (&) and the double quote (") -- have special meaning within 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.)

To use one of these 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 &
&quot;
the escape sequence for "

Additional escape sequences support accented characters. For example:

&ouml;
the escape sequence for a lowercase o with an umlaut: