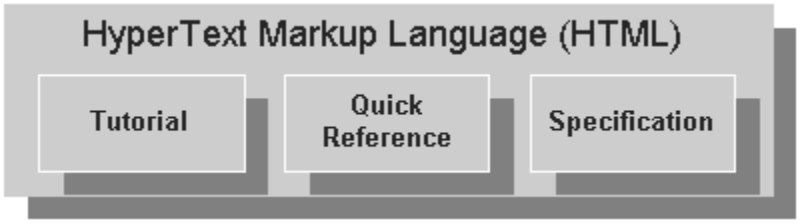
Hypertext Markup Language Assistant



HTML Quick Reference

Basic HTML Elements

<html>...</html>

Used to call out the area of the document subject to HTML: parsing. Usually the first and last lines in the file.

<head>...</head>

Used to set off the header area of the file, in which should be found the TITLE and other header elements

<body>...</body>

Used to set off the body of the file.

<isindex>

Causes most browsers to display a window for launching searches of the file in question.

<title>...</title>

Names the document. Does not create a visible header with the title information.

<nextid>

Links the file in question to another file. ARCHAIC.

<base>

Specifies the base path name for the file set.

<p>

Causes a paragraph break with space.

Causes a line break with no space.

<pre>...</pre>

Turns off HTML parsing of information between the tags.

<listing>...</listing>

Turns off HTML parsing of information between the tags

<plaintext>

Treats everything to the end of the file as flat text (no HTML parsing)

<blockquote>...</blockquote>

Used to set off a quotation. Typically causes indents on left and right margins.

...

Identifies an anchor in the document: a place to jump TO.

...

Identifies a link to an anchor in this or another file, determined by the presence of a URL or file specification (linking to another file) or a "#" designator, which signals a link to an anchor in this file.

<h1>...</h1>

Creates a top-level heading.

<h2>...<.h2>

Creates a second-level heading.

<h3>...</h3>

Creates a third-level heading.

<h4>...</h4>

Creates a fourth-level heading.

<h5>...</h5>

Creates a fifth-level heading.

<h6>...</h6>

Creates a sixth-level heading.

...

Denotes emphasis. Typically treated as bold (...)

...
<code>...</code>
<samp>...</samp>
<kdb>...</kbd>
<var>...</var>
<dfn>...</dfn>
<cite>...</cite>

...

Bold.

...

Italics.

<u>...</u>

Underlined text.

<tt>...</tt>

Typewriter/unispace font.

<dl>...<dt>...<dd>...</dl>

Definition List. Terms are called out by <dt> and entries by <dd>.

**
...
...
**

Unordered list. Typically creates a bulleted list.

**
...
...
**

Ordered list. Typically creates a numbered list.

<menu>...<li>...</menu>

Creates a menu of options.

<dir>...<li>...</dir>

Creates a directory listing.

<!--....-->

Comments. Used like this: <!-- THIS IS A COMMENT>

<address>...</address>

Used to add information to the trailer/end of the document, after </body>

<img src=....>...</img>

Used to identify an image to be inserted into the document at the point when <img src= is detected.

The HTML Specification

Hypertext Markup Language (HTML)

A Representation of Textual Information and MetaInformation for Retrieval and Interchange

Status of this Document

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are working documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

Distribution of this document is unlimited.The document is a draft form of a standard for interchange of information on the network which is proposed to be registered as a MIME (RFC1341) content type. Please send comments to timbl@info.cern.ch or the discussion list www-talk@info.cern.ch.

This is version 1.2 of this draft. This document is available in hypertext on the World-Wide Web as http://info.cern.ch/hypertext/WWW/MarkUp/HTML.html

Abstract

HyperText Markup Language (HTML) can be used to represent

- Hypertext news, mail, online documentation, and collaborative hypermedia;

- Menus of options;

- Database query results;

- Simple structured documents with inlined graphics.

- Hypertext views of existing bodies of information

The World Wide Web (W3) initiative links related information throughout the globe. HTML provides one simple format for providing linked information, and all W3 compatible programs are required to be capable of handling HTML. W3 uses an Internet protocol (Hypertext Transfer Protocol, HTTP), which allows transfer representations to be negotiated between client and server, the result being returned in an extended MIME

message. HTML is therefore just one, but an important one, of the representations used with W3.

HTML is proposed as a MIME content type.

Implementations of HTML parsers and generators can be found in the various W3 servers and browsers, in the public domain W3 code, and may also be built using various public domain SGML parsers such as [SGMLS] . HTML is an SGML document type with fairly generic
semantics appropriate for representing information from a wide range of applications. It is more generic than many specific SGML applications, but is still completely device-independent.

IN THIS DOCUMENT

This document contains the following parts:

Vocabulary used in this document, degrees of imperative.

HTML and MIME with discussion of character sets.

HTML and SGML and the relationship between them, and Structured text : an introduction for beginners to SGML.

HTML Elements A list with description, example, and typical rendering.

HTML Entities Entities used to describe characters.

The HTML DTDThe text of the SGML DTD for HTML

Link relationship values . A provisional list. Not part of the standard.

Vocabulary

This specification uses the words below with the precise meaning given.

Representation The encoding of information for interchange. For example, HTML is a representation of hypertext.

Rendering The form of presentation to information to the human reader.

IMPERATIVES

may The implementation is not obliged to follow this in any way.

must If this is not followed, the implementation does not conform to this specification.

shall as "must"

should If this is not followed, though the implementation officially conforms to the standard, undesirable results may occur in practice.

typical Typical rendering is described for many elements. This is not a mandatory part of the standard but is given as guidance for designers and to help explain the uses for which the elements were intended.

NOTES

Sections marked "Note:" are not mandatory parts of the specification but for guidance only.

STATUS OF FEATURES

Mainstream All parsers must recognize these features. Features are mainstream unless otherwise mentioned.

Extra Standard HTML features which may safely be ignored by parsers. It is legal to ignore these, treat the contents as though the tags were not there. (e.g. EM, and any undefined elements)

Obsolete Not standard HTML. Parsers should implement these features as far as possible in order to preserve back-compatibility with previous versions of this specification.

INTRODUCTION

The HyperText Markup Language is defined in terms of the ISO Standard Generalized Markup Language []. SGML is a system for defining structured document types and markup languages to represent instances of those document types.

Every SGML document has three parts:

An SGML declaration, which binds SGML processing quantities and syntax token names to specific values. For example, the SGML declaration in the HTML DTD specifies that the string that opens a tag is </ and the maximum length of a name is 40 characters.

A prologue including one or more document type declarations, which specifiy the

element types, element relationships and attributes, and references that can be represented by markup. The HTML DTD specifies, for example, that the HEAD element contains at most one TITLE element.

An instance, which contains the data and markup of the document.

We use the term HTML to mean both the document type and the markup language for representing instances of that document type.

All HTML documents share the same SGML declaration an prologue. Hence implementations of the WorldWide Web generally only transmit and store the instance part of an HTML document. To construct an SGML document entity for processing by an SGML parser, it is necessary to prefix the text from ``HTML DTD'' on page 10 to the HTML instance.

Conversely, to implement an HTML parser, one need only implement those parts of an SGML parser that are needed to parse an instance after parsing the HTML DTD.

Structured Text

An HTML instance is like a text file, except that some of the characters are interpreted as markup. The markup gives structure to the document.

The instance represents a hierarchy of elements. Each element has a name , some attributes , and some content. Most elements are represented in the document as a start tag, which gives the name and attributes, followed by the content, followed by the end tag.

For example:

```
 <HTML>
  <TITLE>
A sample HTML instance
  </TITLE>
  <H1>
An Example of Structure
  </H1>
  Here's a typical paragraph.
  <P>
  <UL>
<LI>
Item one has an
<A NAME="anchor">
 anchor
</A>
<LI>
Here's item two.
```

```
  </UL>
</HTML>
```

Some elements (e.g. P, LI) are empty. They have no content. They show up as just a start tag.

For the rest of the elements, the content is a sequence of data characters and nested elements. Note that the HTML DTD in fact severely limits the amount of nesting which is allowed: most things cannot be nested, in fact.No elements may be recursively nested. Anchors and character highlighting may be put inside other constructs.

Tags

Every element starts with a tag, and every non-empty element ends with a tag. Start tags are delimited by < and >, and end tags are delimited by </ and >.

Names

The element name immediately follows the tag open delimiter. Names consist of a letter followed by up to 33 letters, digits, periods, or hyphens. Names are not case sensitive.

Attributes

In a start tag, whitespace and attributes are allowed between the element name and the closing delimiter. An attribute consists of a name, an equal sign, and a value. Whitespace is allowed around the equal sign.

The value is specified in a string surrounded by single quotes or a string surrounded by double quotes. (See: other tolerated forms @@)

The string is parsed like RCDATA (see below) to determine the attribute value. This allows, for example, quote characters in attribute values to be represented by character references.

The length of an attribute value (after parsing) is limited to 1024 characters.

ELEMENT TYPES

The name of a tag refers to an element type declaration in the HTML DTD. An element type declaration associates an element name witha list of attributes and their types and statuses.

A content type (one of EMPTY, CDATA, RCDATA, ELEMENT, or MIXED) which determines the syntax of the element's content

A content model, which specifies the pattern of nested elements and data

Empty Elements

Empty elements have the keyword EMPTY in their declaration. For example:

```
<!ELEMENT NEXTID - O EMPTY>
<!ATTLIST NEXTID N NUMBER #REQUIRED>
```

This means that the following:

```
<nextid n=''27''>
```

is legal, but these others are not:

```
<nextid>
<nextid n=''abc''>
```

Character Data

The keyword CDATA indicates that the content of an element is character data. Character data is all the text up to the next end tag open delimiter-in-context. For example:

<!ELEMENT XMP - - CDATA>

specifies that the following text is a legal XMP element:

```
<xmp>Here's an example. It looks like it has
<tags> and <!--comments-->
in it, but it does not. Even this
</ is data.</xmp>
```

The string </ is only recognized as the opening delimiter of an end tag when it is ``in context,'' that is, when it is followed by a letter. However, as soon as the end tag open delimiter is recognized, it terminates the CDATA content. The following is an error:

```
<xmp>There is no way to represent </end> tags
in CDATA </xmp>
```

Replaceable Character Data

Elements with RCDATA content behave much like those with CDATA, except for character references and entity references. Elements declared like:

<!ELEMENT TITLE - - RCDATA>

can have any sequence of characters in their content.

Character References

To represent a character that would otherwise be recognized as markup, use a character reference. The string &# signals a character reference when it is followed by a letter or a digit. The delimiter is followed by the decimal character number and a semicolon. For example:

```
<title>You can even represent &#60;/end> tags in RCDATA
</title>
```

Entity References

The HTML DTD declares entities for the less than, greater than, and ampersand characters and each of the ISO Latin 1 characters so that you can reference them by name rather than by number.

The string & signals an entity reference when it is followed by a letter or a digit. The delimiter is followed by the entity name and a semicolon. For example:

```
Kurt G&ouml;del was a famous logician and mathematician.
```

Note: To be sure that a string of characters has no markup, HTML writers should represent all occurrences of <, >, and & by character or entity references.

Element Content

Some elements have, in stead of a keyword that states the type of content, a content model, which tells what patterns of data and nested elements are allowed. If the content model of an element does not include the symbol #PCDATA , the content is element content.

Whitespace in element content is considered markup and ignored. Any characters that are not markup, that is, data characters, are illegal.

For example:

<!ELEMENT HEAD - - (TITLE? & ISINDEX? & NEXTID? & LINK*)>

declares an element that may be used as follows:

```
<head>
 <isindex>
 <title>Head Example</title>
</head>
```

But the following are illegal:

```
<head> no data allowed! </head>
<head><isindex><title>Two isindex
tags</title><isindex></head>
```

Mixed Content

If the content model includes the symbol #PCDATA, the content of the element is parsed as mixed content. For example:

```
<!ELEMENT PRE - - (#PCDATA | A | B | I | U | P)+>
<!ATTLIST PRE
  WIDTH NUMBER #implied
  >
```

This says that the PRE element contains one or more A, B, I, U, or P elements or data characters. Here's an example of a PRE element:

```
<pre>
<b>NAME</b>
 cat -- concatenate<a href=''terms.html#file''>files</a>
<b>EXAMPLE</b>
 cat <xyz
</pre>
```

The content of the above PRE element is:

A B element

The string ``cat -- concatenate''

An A element

The string ``\n''

Another B element

The string ``\ncat <xyz''

COMMENTS AND OTHER MARKUP

To include comments in an HTML document that will be ignored by the parser, surround them with <!-- and -->. After the comment delimiter, all text up to the next occurrence of -- is ignored. Hence comments cannot be nested. Whitespace is allowed between the closing -- and >. (But not between the opening <! and --.)

For example:

```
<HEAD>
<TITLE>HTML Guide: Recommended Usage</TITLE>
<!-- $Id: recommended.html,v 1.3 93/01/06 18:38:11 connolly
Exp $ -->
</HEAD>
```

There are a few other SGML markup constructs that are deprecated or illegal.

DelimiterSignals...

<? Processing instruction. Terminated by >.

<![Marked section. Marked sections are deprecated. See the SGML standard for complete information.

<! Markup declaration. HTML defines no short reference maps, so these are errors. Terminated by >.

LINE BREAKS

A line break character is considered markup (and ignored) if it is the first or last piece of content in an element. This allows you to write either

```
<PRE>some example text</pre>
```

or

```
<pre>
some example text
</pre>
```

and these will be processed identically.

Also, a line that's not empty but contains no content will be ignored altogether. For example, the element

```
<pre>
<!-- this line is ignored, including the linebreak character
-->
first line

third line<!-- the following linebreak is content: -->
fourth line<!-- this one's ignored because it's the last
piece of cont
ent: -->
```

```
    </pre>
```

contains only the strings

first line

third line
fourth line.

SPACES AND TABS

Space characters must be rendered as horizontal white space. In HTML, multiple spaces should be rendered as proportionally larger spaces.

The rendering of a horizontal tab (HT) character is not defined, and HT should therefore not be used, except within a PRE (or obsolete XMP, LISTING or PLAINTEXT) element.

Neither spaces nor tabs should be used to make SGML source layout more attractive or easier to read.

SUMMARY OF MARKUP SIGNALS

The following delimiters may signal markup, depending on context.

DelimiterSignals

<!-- Comment

&# Character reference

& Entity reference

</ End tag

<! Markup declaration

]]>Marked section close (an error)

< Start tag

HTML ELEMENTS

This is a list of elements used in the HTML language. Documents should (but need not absolutely) contain an initial HEAD element followed by a BODY element.

Old style documents may contain a just the contents of the normal HEAD and BODY elements, in any order. This is deprecated but must be supported by parsers.

See also: Status of elements

Properties of the whole document

Properties of the whole document are defined by the following elements. They should appear within the HEAD element. Their order is not significant.

TITLE The title of the document

ISINDEX Sent by a server in a searchable document

NEXTIDA parameter used by editors to generate unique identifiers

LINK Relationship between this document and another. See also the Anchor element , Relationships . A document may have many LINK elements.

BASE A record of the URL of the document when saved

Text formatting

These are elements which occur within the BODY element of a document. Their order is the logical order in which the elements should be rendered on the output device.

Headings Several levels of heading are supported.

Anchors Sections of text which form the beginning and/or end of hypertext links are called "anchors" and defined by the A tag.

Paragraph marksThe P element marks the break between two paragraphs.

Address style An ADDRESS element is displayed in a particular style.

Blockquote style A block of text quoted from another source.

Lists Bulleted lists, glossaries, etc.

Preformatted text Sections in fixed-width font for preformatted text.

Character highlighting Formatting elements which do not cause paragraph breaks.

Graphics

IMGThe IMG tag allows inline graphics.

Obsolete elements

The other elements are obsolete but should be recognised by parsers for back-compatibility.

HEAD

The HEAD element contains all information about the document in general. It does not contain any text which is part of the document: this is in the BODY. Within the head element, only certain elements are allowed.

BODY

The BODY element contains all the information which is part of the document, as opposed information about the document which is in the HEAD .

The elements within the BODY element are in the order in which they should be presented to the reader.

See the list of things which are allowed within a BODY element .

Anchors

An anchor is a piece of text which marks the beginning and/or the end of a hypertext link.

The text between the opening tag and the closing tag is either the start or destination (or both) of a link. Attributes of the anchor tag are as follows.

HREF OPTIONAL. If the HREF attribute is present, the anchor is sensitive text: the start of a link. If the reader selects this text, (s)he should be presented with another document whose network address is defined by the value of the HREF attribute . The format of the network address is specified elsewhere . This allows for the form HREF="#identifier" to refer to another anchor in the same document. If the anchor is in another document, the attribute is a relative name , relative to the documents address (or specified base address if any).

NAME OPTIONAL. If present, the attribute NAME allows the anchor to be the destination of a link. The value of the attribute is an identifier for the anchor. Identifiers are arbitrary strings but must be unique within the HTML document.Another document can then make a reference explicitly to this anchor by putting the identifier after the address, separated by a hash sign .

RELOPTIONAL. An attribute REL may give the relationship (s) described by the hypertext link. The value is a comma-separated list of

relationship values.Values and their semantics will be registered by the HTML registration authority . The default relationship if none other is given is void. REL should not be present unless HREF is present. See Relationship values , REV .

REVOPTIONAL. The same as REL , but the semantics of the link type are in the reverse direction. A link from A to B with REL="X" expresses the same relationship as a link from B to A with REV="X".An anchor may have both REL and REV attributes.

URNOPTIONAL. If present, this specifies a uniform resource number for the document. See note .

TITLE OPTIONAL. This is informational only. If present the value of this field should equal the value of the TITLE of the document whose address is given by the HREF attribute. See note .

METHODS OPTIONAL. The value of this field is a string which if present must be a comma separated list of HTTP METHODS supported by the object for public use. See note .

All attributes are optional, although one of NAME and HREF is necessary for the anchor to be useful. See also: LINK .

EXAMPLE OF USE:

```
  See <A HREF="http://info.cern.ch/">CERN</A>'s information
for
  more details.

  A <A NAME=serious>serious</A> crime is one which is
associated
  with imprisonment.
...
  The Organization may refuse employment to anyone convicted
  of a <a href="#serious">serious</A> crime.
```

NOTE : UNIVERSAL RESOURCE NUMBERS

URNs are provided to allow a document to be recognized if duplicate copies are found. This should save a client implementation from picking up a copy of something it already has.

The format of URNs is under discussion (1993) by various working groups of the Internet Engineering Task Force.

NOTE: TITLE ATTRIBUTE OF LINKS

The link may carry a TITLE attribute which should if present give the title of the document whose address is given by the HREF attribute.

This is useful for at least two reasons

The browser software may chose to display the title of the document as a preliminary to retrieving it, for example as a margin note or on a small box while the mouse is over the anchor, or during document fetch.

Some documents -- mainly those which are not marked up text, such as graphics, plain text and also Gopher menus, do not come with a title themselves, and so putting a title in the link is the only way to give them a title. This is how Gopher works. Obviously it leads to duplication of data, and so it is dangerous to assume that the title attribute of the link is a valid and unique title for the destination document.

NOTE: METHODS ATTRIBUTE OF LINKS

The METHODS attributes of anchors and links are used to provide information about the functions which the user may perform on an object. These are more accurately given by the HTTP protocol when it is used, but it may, for similar reasons as for the TITLE attribute, be useful to include the information in advance in the link.

For example, The browser may chose a different rendering as a function of the methods allowed (for example something which is searchable may get a different icon)

Address

This element is for address information, signatures, authorship,etc, often at the top or bottom of a document.

Typically, an address element is italic and/or right justified or indented. The address element implies a paragraph break. Paragraph marks within the address element do not cause extra white space to be inserted.

```
<ADDRESS><A HREF="Author.html">A.N.Other</A></ADDRESS>


<ADDRESS>
Newsletter editor<p>
J.R. Brown<p>
JimquickPost News, Jumquick, CT 01234<p>
Tel (123) 456 7890
</ADDRESS>
```

BASE

This element allows the URL of the document itself to be recorded in situations in which the document may be read out of context.URLs within the document may be in a "partial" form relative to this base address.

Where the base address is not specified, the reader will use the URL it used to access the document to resolve any relative URLs.

The one attribute is:

HREF the URL

BLOCKQUOTE

The BLOCKQUOTE element allows text quoted from another source to be rendered specially.

A typical rendering might be a slight extra left and right indent, and/or italic font. BLOCKQUOTE causes a paragraph break, and typically a line or so of white space will be allowed between it and any text before or after it.

Single-font rendition may for example put a vertical line of ">" characters down the left margin to indicate quotation in the Internet mail style.

```
I think it ends
<BLOCKQUOTE>Soft you now, the fair Ophelia. Nymph, in thy
orisons,
be all my sins remembered.
</BLOCKQUOTE>
but I am not sure.
```

Headings

Six levels of heading are supported. (Note that a hypertext node within a hypertext work tends to need less levels of heading than a work whose only structure is given by the nesting of headings.)

A heading element implies all the font changes, paragraph breaks before and after, and white space (for example) necessary to render the heading. Further character emphasis or paragraph marks are not required in HTML.

H1 is the highest level of heading, and is recommended for the start of a hypertext node.It is suggested that the the text of the first heading be suitable for a reader who is already browsing in related information, in contrast to the title tag which should identify

the node in a wider context.

The heading elements are

<H1>, <H2>, <H3>, <H4>, <H5>, <H6>

It is not normal practice to jump from one header to a header level more than one below, for example for follow an H1 with an H3. Although this is legal, it is discouraged, as it may produce strange results for example when generating other representations from the HTML.

```
<H1>This is a heading</H1>
Here is some text
<H2>Second level heading</H2>
Here is some more text.
```

Parsers should not require any specific order to heading elements, even if the heading level increases by more than one betweensuccessive headings.

H1 Bold very large font, centered. One or two lines clear space between this and anything following. If printed on paper, start new page.

H2 Bold, large font,, flush left against left margin, no indent. One or two clear lines above and below.

H3 Italic, large font, slightly indented from the left margin. One or two clear lines above and below.

H4 Bold, normal font, indented more than H3. One clear line above and below.

H5 Italic, normal font, indented as H4. One clear line above.

H6 Bold, indented same as normal text, more than H5. One clear line above.

These typical values are just an indication, and it is up to the designer of the presentation software to define the styles. The reader may have options to customize these. When writing documents, you should assume that whatever is done it is designed to have the same sort of effect as the styles above.

The rendering software is responsible for generating suitable vertical white space between elements, so it is NOT normal or required to follow a heading element with a paragraph mark.

IMG: Embedded Images

The IMG element allows another document to be inserted inline. The document is normally an icon or small graphic, etc. This element is NOT intended for embedding other HTML text.

Browsers which are not able to display inline images ignore IMG elements. Authors should note that some browsers will be able to display (or print) linked graphics but not inline graphics. If the graphic is essential, it may be wiser to make a link to it rather than to put it inline. If the graphic is essentially decorative, then IMG is appropriate.

The IMG element is empty: it has no closing tag. It has two attributes:

SRCThe value of this attribute is the URL of the document to be embedded. Its syntax is the same as that of the HREF attribute of the A tag. SRC is mandatory.

ALIGN Take values TOP or MIDDLE or BOTTOM, defining whether the tops or middles of bottoms of the graphics and text should be aligned vertically.

ALT Optional alternative text as an alternative to the graphics for display in text-only environments.

Note that IMG elements are allowed within anchors.

EXAMPLE

```
  Warning: < IMG SRC ="triangle.gif" ALT="Warning:"> This
must be done by a   qualified technician.

  < A HREF="Go">< IMG SRC ="Button"> Press to start</A>
```

I**SINDEX**

This element informs the reader that the document is an index document. As well as reading it, the reader may use a keyword search.

The node may be queried with a keyword search by suffixing the node address with a question mark, followed by a list of keywords separated by plus signs. See the network address format .

Note that this tag is normally generated automatically by a server. If it is added by hand to an HTML document, then the client will assume that the server can handle a search on the document.

Obviously the server must have this capability for it to work: simply adding <ISINDEX>

in the document is not enough to make searches happen if the server does not have a search engine!

Status: standard.

LINK

The LINK element occurs within the HEAD element of an HTML document. It is used to indicate a relationship between the document and some other object. A document may have any number of LINK elements.

The LINK element is empty, but takes the same attributes as the anchor element .

Typical uses are to indicate authorship, related indexes and glossaries, older or more recent versions, etc. Links can indicate a static tree structure in which the document was authored by pointing to a "parent" and "next" and "previous" document, for example.

Servers may also allow links to be added by those who do not have the right to alter the body of a document.

Forms of list in HTML

GLOSSARIES

A glossary (or definition list) is a list of paragraphs each of which has a short title alongside it. Apart from glossaries, this element is useful for presenting a set of named elements to the reader. The elements within a glossary follow are

DT The "term", typically placed in a wide left indent

DD The "definition", which may wrap onto many lines

These elements must appear in pairs. Single occurrences of DT without a following DD are illegal. The one attribute which DL can take is

COMPACT suggests that a compact rendering be used, because the enclosed elements are individually small, or the whole glossary is rather large, or both.

Typical rendering

The definition list DT, DD pairs are arranged vertically.For each pair, the DT element is

on the left, in a column of about a third of the display area, and the DD element is in the right hand two thirds of the display area. The DT term is normally small enough to fit on one line within the left-hand column. If it is longer, it will either extend across the page, in which case the DD section is moved down to separate them, or it is wrapped onto successive lines of the left hand column.

White space is typically left between successive DT,DD pairs unless the COMPACT attribute is given. The COMPACT attribute is appropriate for lists which are long and/or have DT,DD pairs which each take only a line or two. It is of course possible for the rendering software to discover these cases itself and make its own decisions, and this is to be encouraged.

The COMPACT attribute may also reduce the width of the left-hand (DT) column.

Examples of use

```
<DL>
<DT>Term the first<DD>definition paragraph is reasonably
long but is still displayed clearly
<DT>Term2 follows<DD>Definition of term2
</DL>

<DL COMPACT>
<DT>Term<DD>definition paragraph
<DT>Term2<DD>Definition of term2
</DL>
```

LISTS

A list is a sequence of paragraphs, each of which may be preceded by a special mark or sequence number. The syntax is:

```
<UL>
<LI> list element
<LI> another list element ...
</UL>
```

The opening list tag may be any of UL, OL, MENU or DIR. It must be immediately followed by the first list element.

Typical rendering

The representation of the list is not defined here, but a bulleted list for unordered lists,

and a sequence of numbered paragraphs for an ordered list would be quite appropriate. Other possibilities for interactive display include embedded scrollable browse panels.

List elements with typical rendering are:

UL A list of multi-line paragraphs, typically separated by some white space and/or marked by bullets, etc.

OL As UL, but the paragraphs are typically numbered in some way to indicate the order as significant.

MENU A list of smaller paragraphs. Typically one line per item, with a style more compact than UL.

DIRA list of short elements, typically less than 20 characters. These may be arranged in columns across the page, typically 24 character in width. If the rendering software is able to optimize the column width as function of the widths of individual elements, so much the better.

Example of use

```
<OL>
<LI> When you get to the station, leave
by the southern exit, on platform one.
<LI>Turn left to face toward the mountain
<LI>Walk for a mile or so until you reach the
"Asquith Arms" then
<LI>Wait and see...
</OL>

< MENU >
<LI>The oranges should be pressed fresh
<LI>The nuts may come from a packet
<LI>The gin must be good quality
</MENU>

< DIR >
<LI>A-H<LI>I-M
<LI>M-R<LI>S-Z
</DIR>
```

Next ID

This tag takes a single attribute which is the number of the next document-wide numeric identifier to be allocated of the form z123.

When modifying a document, old anchor ids should not be reused, as there may be references stored elsewhere which point to them. This
is read and generated by hypertext editors. Human writers of HTML usually use mnemonic alphabetical identifiers. Browser software may ignore this tag.

```
<NEXTID N=27>
```

P: Paragraph mark

The empty P element indicates a paragraph break. The exact rendering of this (indentation, leading, etc) is not defined here, and may be a function of other tags, style sheets etc.

<P> is used between two pieces of text which otherwise would be flowed together.

You do NOT need to use <P> to put white space around heading, list, address or blockquote elements which imply a paragraph break. It is the responsibility of the rendering software to generate that white space.A paragraph mark which is preceded or followed by such elements which imply a paragraph break is has undefined effect and should be avoided.

TYPICAL RENDERING

Typically, <P> will generate a small vertical space (of a line or half a line) between the paragraphs. This is not the case (typically) within ADDRESS or (ever) within PRE elements.With some implementations, in normal text, <P> may generate a small extra left indent on the first line.

EXAMPLES OF USE

```
<h1>What to do</h1>
This is a one paragraph.< p >This is a second.
< P >
This is a third.
```

BAD EXAMPLE

```
<h1><P>What not to do</h1>
<p>I found that on my XYZ browser it looked prettier to
me if I put some paragraph marks
<p>
<ul><p><li>Around lists, and
<li>After headings.
</ul>
```

```
<p>
None of the paragraph marks in this example should
be there.
```

PRE: Preformatted text

Preformatted elements in HTML are displayed with text in a fixed width font, and so are suitable for text which has been formatted for a teletype by some existing formatting system.

The optional attribute is:

WIDTHThis attribute gives the maximum number of characters which will occur on a line. It allows the presentation system to select a suitable font and indentation. Where the WIDTH attribute is not recognized, it is recommended that a width of 80 be assumed. Where WIDTH is supported, it is recommended that at least widths of 40, 80 and 132 characters be presented optimally, with other widths being rounded up.

Within a PRE element,

Line boundaries within the text are rendered as a move to the beginning of the next line, except for one immediately following or immediately preceding a tag.

The <p> tag should not be used. If found, it should be rendered as a move to the beginning of the next line.

Anchor elements and character highlighting elements may be used.

Elements which define paragraph formatting (Headings, Address, etc) must not be used.

The ASCII Horizontal Tab (HT) character must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended however.

Example of use

```
<PRE WIDTH="80">
This is an example line
</PRE>
```

Note: Highlighting

Within a preformatted element, the constraint that the rendering must be on a fixed horizontal character pitch may limit or prevent the ability of the renderer to render highlighting elements specially.

Note: Margins

The above references to the "beginning of a new line" must not be taken as implying that the renderer is forbidden from using a (constant) left indent for rendering preformatted text.The left indent may of course be constrained by the width required.

TITLE

The title of a document is specified by the TITLE element. The TITLE element should occur in the HEAD of the document.

There may only be one title in any document. It should identify the content of the document in a fairly wide context.

The title is not part of the text of the document, but is a property of the whole document. It may not contain anchors, paragraph marks, or highlighting. The title may be used to identify the node in a history list, to label the window displaying the node, etc. It is not normally displayed in the text of a document itself. Contrast titles with headings . The title should ideally be less than 64 characters in length. That is, many applications will display document titles in window titles, menus, etc where there is only limited room. Whilst there is no limit on the length of a title (as it may be automatically generated from other data), information providers are warned that it may be truncated if long.

Examples of use

Appropriate titles might be

```
<TITLE>Rivest and Neuman. 1989(b)</TITLE>

<TITLE>A Recipe for Maple Syrup Flap-Jack</TITLE>

<TITLE>Introduction -- AFS user's Guide</TITLE>
```

Examples of inappropriate titles are those which are only meaningful within context,

```
<TITLE>Introduction</TITLE>
```

or too long,

```
  <TITLE>Remarks on the Quantum-Gravity effects of "Bean
  Pole" diversification in Mononucleosis patients in
Developing
```

```
  Countries under Economic Conditions Prevalent during
  the Second half of the Twentieth Century, and Related
Papers:
  a Summary</TITLE>
```

Character highlighting

Status: Extra

These elements allow sections of text to be formatted in a particular way, to provide emphasis, etc. The tags do NOT cause a paragraph break, and may be used on sections of text within paragraphs.

Where not supported by implementations, like all tags, these tags should be ignored but the content rendered.

All these tags have related closing tags, as in

This is <EM>emphasized</EM> text.

Some of these styles are more explicit than others about how they should be physically represented. The logical styles should be used wherever possible, unless for example it is necessary to refer to the formatting in the text. (Eg, "The italic parts are mandatory".)

Note:

Browsers unable to display a specified style may render it in some alternative, or the default, style, with some loss of quality for the reader. Some implementations may ignore these tags altogether, so information providers should attempt not to rely on them as essential to the information content.

These element names are derived from TeXInfo macro names.

PHYSICAL STYLES

TT Fixed-width typewriter font.

B Boldface, where available, otherwise alternative mapping allowed.

I Italic font (or slanted if italic unavailable).

U Underline.

LOGICAL STYLES

EM Emphasis, typically italic.

STRONGStronger emphasis, typically bold.

CODE Example of code. typically monospaced font. (Do not confuse with PRE)

SAMP A sequence of literal characters.

KBDin an instruction manual, Text typed by a user.

VARA variable name.

DFNThe defining instance of a term. Typically bold or bold italic.

CITE A citation. Typically italic.

EXAMPLES OF USE

```
  This text contains an <em>emphasized</em> word.
<strong>Don't assume</strong> that it will be italic!   It
was made using the <CODE>EM</CODE> element. A citation is
typically italic and has no formal necessary structure:
<cite>Moby Dick</cite> is a book title.
```

Obsolete elements

The following elements of HTML are obsolete. It is recommended that client implementors implement the obsolete forms for compatibility with old servers.

Plaintext

Status: Obsolete .

The empty PLAINTEXT tag terminates the HTML entity. What follows is not SGML. In stead, there's an old HTTP convention that what follows is an ASCII (MIME "text/plain") body.

An example if its use is:

```
<PLAINTEXT>
0001 This is line one of a ling listing
```

```
0002 file from <any@host.inc.com> which is sen
</PLAINTEXT>
```

This tag allows the rest of a file to be read efficiently without parsing. Its presence is an optimization. There is no closing tag. The rest of the data is not in SGML.

XMP and LISTING:Example sections

Status: Obsolete . This are in use and should be recognized by browsers. New servers should use <PRE> instead.

These styles allow text of fixed-width characters to be embedded absolutely as is into the document. The syntax is:

```
<LISTING>
  ...
</LISTING>

or

<XMP>
  ...
</XMP>
```

The text between these tags is to be portrayed in a fixed width font, so that any formatting done by character spacing on successive lines will be maintained. Between the opening and closing tags:

The text may contain any ISO Latin printable characters, but not the end tag opener. (See Historical note)

Line boundaries are significant, except any occurring immediately after the opening tag or before the closing tag. and are to be rendered as a move to the start of a new line.

The ASCII Horizontal Tab (HT) character must be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended however.

The LISTING element is portrayed so that at least 132 characters will fit on a line. The XMP elementis portrayed in a font so that at least 80 characters will fit on a line but is otherwise identical to LISTING.

Highlighted Phrase HP1 etc

Status: Obsolete . These tags like all others should be ignored if not implemented. Replaced will more meaningful elements -- see character highlighting .

Examples of use:

```
<HP1>...</HP1><HP2>... </HP2> etc.
```

Comment element

Status: Obsolete

A comment element used for bracketing off unneed text and comment has been introduced in some browsers but will be replaced by theSGML command feature in new implementations.

ENTITIES

List Of Entities

The following entity names are used in HTML , always prefixed by ampersand (&) and followed by a semicolon as shown. They represent particular graphic characters which have special meanings in places in the markup, or may not be part of the character set available to the writer.

< The less than sign <

> The "greater than" sign >

& The ampersand sign & itself.

"The double quote sign "

Æ capital AE diphthong (ligature)

Ácapital A, acute accent

Â capital A, circumflex accent

Àcapital A, grave accent

Å capital A, ring

Ãcapital A, tilde

Ä capital A, dieresis or umlaut mark

Çcapital C, cedilla

Ðcapital Eth, Icelandic

Écapital E, acute accent

Ê capital E, circumflex accent

Ècapital E, grave accent

Ë capital E, dieresis or umlaut mark

Ícapital I, acute accent

Î capital I, circumflex accent

Ìcapital I, grave accent

Ï capital I, dieresis or umlaut mark

Ñcapital N, tilde

Ócapital O, acute accent

Ô capital O, circumflex accent

Òcapital O, grave accent

Øcapital O, slash

Õcapital O, tilde

Ö capital O, dieresis or umlaut mark

Þ capital THORN, Icelandic

Úcapital U, acute accent

Û capital U, circumflex accent

Ùcapital U, grave accent

Ü capital U, dieresis or umlaut mark

Ýcapital Y, acute accent

ásmall a, acute accent

â small a, circumflex accent

æ small ae diphthong (ligature)

àsmall a, grave accent

å small a, ring

ãsmall a, tilde

ä small a, dieresis or umlaut mark

çsmall c, cedilla

ésmall e, acute accent

ê small e, circumflex accent

èsmall e, grave accent

ðsmall eth, Icelandic

ë small e, dieresis or umlaut mark

ísmall i, acute accent

î small i, circumflex accent

ìsmall i, grave accent

ï small i, dieresis or umlaut mark

ñsmall n, tilde

ósmall o, acute accent

ô small o, circumflex accent

òsmall o, grave accent

øsmall o, slash

õsmall o, tilde

ö small o, dieresis or umlaut mark

ß small sharp s, German (sz ligature)

þ small thorn, Icelandic

úsmall u, acute accent

û small u, circumflex accent

ùsmall u, grave accent

ü small u, dieresis or umlaut mark

ýsmall y, acute accent

ÿ small y, dieresis or umlaut mark

About This Hypertext

This tutorial is a reformatted and reorganized version of the standard HTML tutorial prepared by the National Center for Supercomputer Applications (NCSA). The orginal document is available via
http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html

In sections of the original document, where the advice of the NCSA sages differ from that I would give, or when I have something to add, that text will appear in ***bold italics with an at the beginning.***

I have also added new sections to the material. These are marked with an in the list below.

Marc Demarest
demarest@hevanet.com

January 1995

HMTL Tutorial

About This Hypertext

Theory

What Is HTML?
How Does HTML Work?

Practice

The Structure Of A Hypertext Web
The Structure Of An HTML Document
Creating HTML Documents
The Minimal HTML Document
Basic Markup Tags
Titles
Headings
Paragraphs
Comments
Linking to Other Documents
Relative Links Versus Absolute Pathnames
Uniform Resource Locators
Anchors to Specific Sections in Other Documents
Anchors to Specific Sections Within the Current Document
Additional Markup Tags
Unnumbered Lists
Numbered Lists
Definition Lists
Nested Lists
Preformatted Text
Extended Quotes
Addresses
Character Formatting
Physical Versus Logical -- Use Logical Tags When Possible
Logical Styles
Physical Styles
Special Characters
Forced Line Breaks
Horizontal Rules
In-line Images, External Images, Sounds, and Animations
Troubleshooting
Overlapping And Embedded Tags
Check Your Links
A Longer Example
For More Information

Fill-out Forms
Style Guides
Other Introductory Documents
Additional References

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.

WebEdit is of course another of these editors.

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>
```

This immediately brings up the question of development style. I would have written the same file as follows:

```
<HTML>
<HEAD>
<TITLE>The simplest HTML example</TITLE>
</HEAD>
<BODY>
<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>
</BODY>
</HTML>
```

I believe that, other than titles, headers and hyperlinks, all HTML code elements should be separated from the text they modify by line breaks. Since the browsers ignore the line breaks, you don't affect the way the document is displayed; you just make it easier for you (and other people modifying your code) to see what you are doing.

This is ultimately a matter of personal style; you will discover a way of formatting HTML files that suits you. But pay attention to style, and do all of your HTML files the same way -- other people will look at and use your code.

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

Titles

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.

The title of your document is used by the browsers that interpret it and by Web search tools that try to discover your pages for use by others. Take the title seriously and make it clear and descriptive.

This is a good title:

```
<TITLE>Joe Batz's Electronic Commerce Web Site
Listing</TITLE>
```

This is not a good title:

```
<TITLE>My Home Page</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.

```
<H1>First Level Heading</H1>
<H2>Second Level Heading</H2>
<H3>Third Level Heading</H3>
<H4>Fourth Level Heading</H4>
<H5>Fifth Level Heading</H5>
<H6>Sixth Level Heading</H6>
```

The first heading in each document should be tagged <H1>...</H1>.

I disagree with this position strongly. Most browsers treat H1 headers in truly ugly ways by default -- the text is too large and too bold. I always start with an H2 head, and NEVER go to more than H4 heads, on the principle that, if you need more than three levels of heading in a document, you should chunk it up. I also ALWAYS make the first heading in the document the document title, because some broswers display the document title in strange places (like NOT in the window title bar, like it OUGHT to be done). So, I would say THIS is good design practice:

```
<HTML>
<HEAD>
<TITLE>The Elements Of Hypertext Style</TITLE>
</HEAD>
<BODY>
<H2><A NAME="The Elements Of Hypertext Style">The Elements
Of Hypertext Style</H2>
....
</BODY>
</HTML>
```

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.

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
:

```
<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).

This is an opinion about programming style. There are other opinions, as I mentioned earlier. Your goal is to find a consistent programming style that suits you and is highly readable by others, and stick to it.

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>
```

This will make any browser not compliant with HTML+ die today (except NetScape, which will figure it out).

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.

Don't use <P> after head tags <H?>...</H?>. It just adds unnecessary white space to the file, since all browsers will put extra white space after heads. On the other hand, it is usually a good idea to put <P> after </UL> and </OL> in lists, to give a little extra white space between the end of a list and the next paragraph.

Linking to Other Documents

This whole section in NCSA's document seems confusing to me. Let's try a different model.

You can link a document (a node) to one of three things:

a specific location in the same document (an intranode link)

the "top" of another document (an internode link)

a specific place in another document (an internode link).

When you create any link, you specify two things:

the reference to the other side of the link (where to go)

some text that the user can click on to traverse the link.(the HOT TEXT)

The generic format for this HTML directive is:

```
<A HREF="WHERE TO GO">HOT TEXT</A>
```

Linking one document to the "top" of another document is the easiest kind of link. It looks like this:

```
<A HREF="UPDATES.HTML">Update Information</A>
```

This directive says to the browser parsing it: "In the same directory you got the file you are parsing now, there is another file called UPDATES.HTML. Go and get that file when the user clicks on Update Information."

To link to a specific location in a file, whether you are creating an internode link or an intranode link, you need to specify more than a file name: you also need an anchor name. For example, suppose I wanted to link to the "Latest Edits" section of UPDATES.HTML. The UPDATES.HTML file would have to have an anchor in it, specified link this:

```
<A NAME="LATEST EDITS"></A>
```

I typically make all head anchors, by creating head designations like this:

```
<H2><A NAME="LATEST EDITS">LATEST EDITS</A></H2>
```

This makes a header an anchor, or an anchor a header, depending on your perspective. WebEdit does this for you automatically when you ask WebEdit to

create heads.

To jump to this specific section of UPDATES.HTML from another location within UPDATES.HTML (an intranode link), I would use this hyperlink reference

```
<A HREF="#LATEST EDITS">Look at latest edits</A>
```

To jump to this specific section of UPDATES.HTML from another file (an internode link), I would use this hyperlink reference:

```
<A HREF="UPDATES.HTML#LATEST EDITS">Look at latest edits</A>
```

In both cases, the user would see the hot text "Look at latest edits".

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:

Start the anchor with <A... (There's a space after the A.)

Specify the document that's being pointed to by entering the parameter HREF="filename", followed by a closing right angle bracket:, >.

Enter the text that will serve as the hypertext link in the current document.

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 (visible hot text)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, even when the system on which the file is located is not a UNIX system.

.

Here is a more typical hypertext reference:

```
<a href="http://www.sequent.com/people/marc">Marc
Demarest</a>
```

You'll notice that, in the first example, the file name (MaineStats.html) is called out explicitly. This reference says to the browser reading it: "In the same directory as the document you are currently interpreting, you will find another file called MainStats.html". In the second example, we use a complete uniform resource locator (URL) to specify a protocol for the browser to use (HTTP, the hypertext transfer protocol on which the Web is based), a new network node to retrieve the material from (www.sequent.com) and a new file path: /people/marc. We do not specify a file name, only a directory path. This means we want the file index.html in the directory /people/marc (the browser/server pair always make this assumption).

Relative Links Versus Absolute Pathnames

An absolute pathname specifies the name of the file from the root of the area of the filesystem containing hypertext. In:

```
<A
HREF="HTTP://www.sequent.com/people/marc/dist/index.html">Ma
rc's Shareware</A>
```

the /people/marc/dist/index.html is an absolute path specification.

Suppose I had a file in the same directory called doslist.html that listed the DOS shareware applications I was making available. Once the browser had retrieved that file, I could refer to the file doslist.html in the same directory as:

```
<A HREF="DOSLIST.HTML">Marc's DOS Shareware Applications</A>
```

If I had a file called macdist.html in the directory /PEOPLE/MARC/MACDIST, I could refer to that file in two ways: as either ../MACDIST/MACLIST.HTML or as /PEOPLE/MARC/MACDIST/MACLIST.HTML. The first is a relative pathname (it only makes sense relative to where you are) and the second is an absolute pathname (it makes sense anywhere).

In general, you should use relative links, because

> You have less to type.
>
> 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.

Actually, you should use relative path names only within a directory structure than you control. You decide when these paths change. When you make a reference outside any area of your control, do it completely and explicitly.

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 schemeis 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), ftp (a file on an ftp server), or for some browsers that support SMTP messaging protocols mailto: (followed by a user name, rather than a file specification).

The port number can generally be omitted. (That means unless someone tells you otherwise, leave it out.)

For example, to include a link to the HTML version of primer in your document, you would use

```
<A HREF =
"http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.ht
ml">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

HTTP://INFO.CERN.CH/HYPERTEXT/WWW/ADDRESSING/ADDRESSING.HTM

HTTP://WWW.NCSA.UIUC.EDU/DEMOWEB/URL-PRIMER.HTML

Let's try this one again.

A URL has three parts:

a protocol part

a system name part

a filename part.

The protocols are few and easy to tell apart. They are:

HTTP	***the Hypertext Transfer Protocol***
FTP	***the FTP protocol***
TELNET	***the TELNET protocol***
GOPHER	***the GOPHER protocol***
WAIS	***the Wide Are Information Search protocol***
FILE	***file system access***
MAILTO	***mail message protocol***

There are some others, but these are the important ones. You will use HTTP for the most part.

The system names are in the Internet naming convention format you should be familiar with. Typically, this part of the URL will look like this:

host.domain.type

Where type is EDU (educational institution), COM (commercial institution), ORG (non-profit organization) or NET (network provider). When the system is outside the US, it will frequently have the country code in its name, as in:

monash.edu.au

whene AU is the universal country code for Australia..

The filename parts are absolutre and relative pathnames.

Anchors 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.

Anchors 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.
```

Unnumbered Lists

An unnumbered list is typically treated by a browser as a bulleted list like this:

- ***Item 1***
- ***Item 2***
- ***Item 3***

To make an unnumbered list,

Start with an opening list <UL> tag.

Enter the <LI> tag followed by the individual item. (No closing </LI> tag is needed.)

End with a closing list </UL> tag.

Below is 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.

This is also a popular way to display multiple hyperlinks to form a sort of menu, as in:

```
<UL>
<LI>
<A HREF="HTTP://WWW.SEQUENT.COM/PEOPLE">Home Pages</A>
<LI>
<A HREF="HTTP://WWW.SEQUENT.COM/PRODUCTS">Product
Specifications</A>
<LI>
<A HREF="HTTP:/WWW.SEQUENT.COM/SERVICES">Service
Offerings</A>
```

```
</UL>
```

which produces, visually:

- ***Home Pages***
- ***Product Specifications***
- ***Service Offerings***

In my opinion, this leads to shoddy design unless you describe the link in some detail.

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

I***f you are using nested lists to display hyperlinks, I believe that you have a design flaw; if you have them, it means that you haven't chunked the text properly. While it is true that hierarchies are essential elements of hypertext design, they are not necessarily good navigational aids (unless you assume, as a fundamental design tenet, that your readers are impatient). You should expose the hierarchy one level at a time; theu user will click through it faster.***

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 *
```

The point here is that, when you use <PRE> and </PRE>, you can get away with (a) lots of spaces and (b) line breaks to format text appearance on the page; the browser will not try to reformat the text between the <PRE> and </PRE> tags. This is useful if you are rtying to build a table or chart in HTML 1 or 2 documents. The font for text between <PRE> and </PRE> wi ll be monospace typewriter font.

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 (<, > , and &, respectively) to enter these characters.

Extended Quotes

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>
```

Typically, browsers will treat block quotes as indented from the left margin and sometimes the right margin as well.

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.

Actually, pretend it's not even called <ADDRESS>. Pretend it's called <TRAILER> and </TRAILER> and think of it as the place where you put standard, predictable kinds of housekeeping and navigational information. For example, I always have this line in by ADDRESS section

```
Last updated on <A HREF="UPDATES.HTML">XX-YY-ZZ</A> by <A
HREF="HTTP://WWW.SEQUENT.COM/PEOPLE/MARC">Marc Demarest</A>
<A HREF="MAILTO:MARC@SEQUENT.COM">(marc@sequent.com)</A>
```

This produces a line that looks like this:

Last updated on XX-YY-ZZ by Marc Demarest (marc@sequent.com).

and ALWAYS allows my readers to (a) check the update history, (b) look at my home page or (c) send me mail. I have also seen people use this section for their standard "Go bak to previous page" and "go to next page" pointers and this works well too. The issue is this: PREDICTABILITY. Whatever you do in this space, do it ALWAYS so your readers expect it (and find it).

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.

Physical Versus Logical: Use Logical Tags 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.

All true. Yet common design practice seems to favor physical tagging overwhelmingly.

Logical Styles

I did a little unscientific survey and discovered that, although we are supposed to be using these "logical" styles instead of "physical styles", almost all authors are using physical styles instead. I do that, too.

<DFN>...</DFN> for a word being defined. Typically displayed in *italics.*

<EM>...</EM> for emphasis. Typically displayed in *italics*

<CITE>...</CITE> for titles of books, films, etc. Typically displayed in *italics.*
)

<CODE>...</CODE> for snippets of computer code. Displayed in a `fixed-width font.`

<KBD>... </KBD> for user keyboard entry. Should be displayed in a **`bold fixed-width fon`**t, but many browsers render it in the `plain fixed-width font`

<SAMP>...</SAMP> for computer status messages. Displayed in a `fixed-width font`

<STRONG>...</STRONG> for strong emphasis. Typically displayed in **bold.**

<VAR>...</VAR> for a ``metasyntactic'' variable, where the user is to replace the variable with a specific instance. Typically displayed in *italics.*

Physical Styles

These are officially less desirable than logical styles, but I use them instead.

<B>...</B> **bold text**

```
<B>BOLD TEXT</B>
```

<I>...</I> *italic text*

```
<I>ITALIC TEXT</I>
```

<TT>...</TT> `typewriter text, e.g. fixed-width font`

```
<TT>monospace typewriter font</TT>
```

<UL>...</UL> Underlined Text

```
<UL>Underlined Text</UL>
```

Using Character Tags

Special Characters

I have yet to use one of these; maybe I am not doing enough cool stuff.

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:

< the escape sequence for <

> the escape sequence for >

& the escape sequence for &

" the escape sequence for " (double quote(

Additional escape sequences support accented characters. For example:

ö the escape sequence for a lowercase o with an umlaut: ö

ñ the escape sequence for a lowercase n with an tilde: ñ

È the escape sequence for an uppercase E with a grave accent:
È </DL>

A full list of supported characters can be found at CERN
("HTTP://INFO.CERN.CH/HYPERTEXT/WWW/MARKUP/ISOLAT1.HTML)
.
NOTE: Unlike the rest of HTML, the escape sequences are case sensitive. You cannot, for instance, use < instead of <.

Forced Line Breaks

The
 tag forces a line break with no extra space between lines. (By contrast, most browsers format the <P> paragraph tag with an additional blank line to more clearly indicate the beginning the new paragraph.)

One use of
 is in formatting addresses:

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

which produces

National Center for Supercomputing Applications
605 East Springfield Avenue
Champaign, Illinois 61820-5518

The HTML code

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

would produce

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.

This code

```
<H2>Head 2</H2>
A paragraph of text with some information in it.
<HR>
<H2>Another Head 2</H2>
Another paragraph of text.
<HR>
```

produces

Head 2

A paragraph of text with some information in it.

Another Head 2

Another paragraph of text.

Horizontal rules make great bounding lines for the body of the document as well. I always put one between the end of the body (</BODY>) and the beginning of the trailer (<ADDRESS>)

The line scales with the browser's width.

In-line Images, External Images, Sounds, and Animations

Most Web browsers can display in-line images (that is, images next to text) that are in X Bitmap (XBM) or GIF format. Each image takes time to process and slows down the initial display of the document, so generally you should not include too many or overly large images.

The whole question of images is a design issue. Most people think the "multimedia" aspect of the Web is its really exciting aspect, but -- truth to tell -- most of the images people put in their pages are junk; they add nothing to the navigability of the pages and very little to the aesthetics. The general design rule is: if the graphic doesn't (a) convey information or (b) provide navigational assistance, then TAKE IT OUT. I confess I don't follow this rule myself. My own homepage has a perfectly worthless GIF in it that I put in because I like it.

But -- practically speaking -- you should use GIF and JPG images; these are the only images guaranteed (or almost guaranteed) to work with all browsers. GIFs are typically smaller than JPGs, so GIFS are best. Compared to BMP files, GIFS are spectacular; I can routinely get a 200K BMP down to a 10K GIF. Since people are sucking this stuff over the network (and often over a slow connection) smaller is better.

If you do advance things with pictures, like creating maps, make sure you offer textual alternatives for those people who (a) don't have a graphical browser or (b) turn graphics off to get speed.

To include an in-line image, use

```
<IMG SRC="FILENAME""></IMG>
```

You can also specify alignment using the ALIGN= parameter. This will tell the browser to align the text following the image with the TOP, MIDDLE or BOTTOM of the image. BOTTOM is the default.

To specify the text that should be used in place of the graphic for browsers that can't do graphics, use the ALT= parameter.

A full blown image specification would look like this:

```
<IMG ALIGN=TOP ALT="Marc's Picture" SRC="marcpic.gif"></IMG>
```

One of the useful things you can do with images is use them as part of a hyperlink, to convey information about the quality of the hyperlink. For example, I attach a green diamond to every hyperlink in my pages that points to a place that costs money to use. The hyperlink specification looks like this:

```
<A HREF="HTTP://DOWVISION.COM/WELCOME.HTML"><IMG
ALIGN=MIDDLE SRC="../ICONS/GREEN.GIF"></IMG> DowVision</A>
```

The entire IMG specification is treated as part of the hot text; the green diamond and the text "DowVision" are hot.

Use the same syntax is for links to external animations and sounds. The only difference is the file extension of the linked file. For example,

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

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

File Type	Extension
Plain text	.txt
HTML document	.html
GIF image	.gif
TIFF image	.tiff
XBM bitmap image	.xbm
JPEG image	.jpg or .jpeg
PostScript file	.ps
AIFF sound	.aiff
AU sound	.au
QuickTime movie	.mov
MPEG movie	.mpeg or .mpg

Make sure your intended audience has the necessary viewers. Most UNIX workstations, for instance, cannot view QuickTime movies.

This section confuses me, since it sounds like they are advocating using IMG for embedded these kinds of things. Don't do that. Whenever you include these multimedia objects in an HTMLpage, you should do the following:

1. Make the link optional. (use A HREF= to set up a link)

2. Be very specific about what kind of multimedia object is on the other end of the link. Use the long hand version (a QuickTime movie) not the file suffix (.MOV). And remember that most browsers allow their users to decide what to do with files of particular types.

3. Specify exactly how large the multimedia object is in kilobytes.

4. Whenever possible, provide the tools for viewing the object with the object.

Avoid Overlapping Tags

Consider this snippet of HTML:

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

The word ``overlapping'' is contained within both the <B> and <DFN> tags. How does the browser format it? You won't know until you look, and different browsers will likely react differently.

In general, avoid overlapping tags.

It is acceptable to embed anchors within another HTML element:

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

Do not embed a heading or another HTML element within an anchor:

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

Although most browsers currently handle this example, it is forbidden by the official HTML and HTML+ specifications, and will not work with future browsers.

This will work and is legal as far as I know:

```
<H2><A HREF="Destination.html">My heading</A></H1>
```

Just reverse the precedence of the tags. I'm clueless as to why you would put a hyperlink in a head; I think you should be putting anchors in heads.

Character tags modify the appearance of other tags:

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

However, avoid embedding other types of HTML element tags. For example, it is tempting to embed a heading within a list, in order to make the font size larger:

```
<UL><LI><H1>A large heading</H1>
<UL>
```

```
<LI><H2>Something slightly smaller</H2>
</UL>
```

Although some browsers, such as NCSA Mosaic for the X Window System, format this construct quite nicely, it is unpredictable (because it is undefined) what other browsers will do. For compatibility with all browsers, avoid these kinds of constructs.

What's the difference between embedding a <B> within a <LI> tag as opposed to embedding a <H1> within a <LI> ? This is again a question of SGML. 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.Thus it doesn't make sense to find a <H1> within a list. Character formatting tags also are generally not additive. 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 (here, the italics).

Check Your Links

When an <IMG> tag points at an image that does not exist, a dummy image is substituted.

When this happens, 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).

You can sometimes tell the difference between (a) a correctly formatted <IMG> statement that references an unavailable graphic and (b) an incorrectly formatted <IMG> statement because (a) produces a dummy graphic (a torn page icon, for example) and (b) produces an "ERROR" icon.

Also, ALWAYS CHECK YOUR EXTERNAL REFERENCES PERIODICALLY. If you reference any documents outside the area of the Web that you control, you shoulc check those references once a week to make sure you are not holding onto stale references (pointers to documents that are still on the Web but have moved) or dead references (pointers to documents that are no longer available. Most good WebMasters, when they move a heavily-accessed document, leave a redirection page where the document used to be, telling readers (often in the form of a hyperlink) how to get to the document in its new location.

A Longer Example

Here is a longer example of an HTML document
:

```
<HEAD>
<TITLE>A Longer Example</TITLE>
</HEAD>
<BODY>
<H1>A Longer Example</H1>
This is a simple HTML document. This is the first
paragraph. <P>
This is the second paragraph, which shows special effects.
This is a
word in <I>italics</I>.  This is a word in <B>bold</B>.
Here is an in-lined GIF image: <IMG SRC = "myimage.gif">.
<P>
This is the third paragraph, which demonstrates links.  Here
is
a hypertext link from the word <A HREF =
"subdir/myfile.html">foo</A>
to a document called "subdir/myfile.html". (If you
try to follow this link, you will get an error screen.) <P>
<H2>A second-level header</H2>
Here is a section of text that should display as a
fixed-width font: <P>
<PRE>
On the stiff twig up there
Hunches a wet black rook
Arranging and rearranging its feathers in the rain ...
</PRE>
This is a unordered list with two items: <P>
<UL>
<LI> cranberries
<LI> blueberries
</UL>
This is the end of my example document. <P>
<ADDRESS>Me (me@mycomputer.univ.edu)</ADDRESS>
</BODY>
```

This is how I would format this HTML document:

```
<HEAD>
<TITLE>A Longer Example</TITLE>
</HEAD>
<BODY>
<H1>A Longer Example</H1>
This is a simple HTML document. This is the first paragraph.
<P>
```

```
This is the second paragraph, which shows special effects.
This is a
word in <I>italics</I>.  This is a word in <B>bold</B>.
Here is an in-lined GIF image: <IMG SRC = "myimage.gif">.
<P>
This is the third paragraph, which demonstrates links.  Here
is
a hypertext link from the word <A HREF =
"subdir/myfile.html">foo</A>
to a document called "subdir/myfile.html". (If you
try to follow this link, you will get an error screen.)
<P>
<H2>A second-level header</H2>
Here is a section of text that should display as a fixed-
width font:
<P>
<PRE>
On the stiff twig up there
Hunches a wet black rook
Arranging and rearranging its feathers in the rain ...
</PRE>
This is a unordered list with two items: <P>
<UL>
<LI>
cranberries
<LI>
blueberries
</UL>
This is the end of my example document.
<P>
<HR>
<ADDRESS>
Me (me@mycomputer.univ.edu)
</ADDRESS>
</HTML>
```

As you can see, the differences are "cosmetic" -- the browser will parse the document exactly the same way as it would the version that the NCSA guys propose. But I believe my version is (a) easier to work with and (b) easier to understand at a glance than the NCSA version.

Fill-out Forms

One major feature not discussed here is fill-out forms, which allows users to return information to the World Wide Web server.

T***his is an advanced feature that requires work on the server itself. When you are ready to do forms, there are HTML documents at NCSA and CERN to help you.***

Style Guides

The following offer advice on how to write ``good'' HTML:

HTTP://WWW.WILLAMETTE.EDU/HTML-COMPOSITION/STRICT-HTML.HTML) COMPOSING GOOD HTML</A>

CERN's style guide is available at HTTP://INFO.CERN.CH/HYPERTEXT/WWW/PROVIDER/STYLE/INTRODUCTION.HTML

Other Introductory Documents

HTTP://WWW.UCC.IE/INFO/NET/HTMLDOC.HTML HOW TO WRITE HTML FILES

HTTP://MELMAC.HARRIS-ATD.COM/ABOUT_HTML.HTML INTRODUCTION TO HTML

Additional References

HTTP://KUHTTP.CC.UKANS.EDU/LYNX_HELP/HTML_QUICK.HTML THE HTML QUICK REFERENCE GUIDE which provides a comprehensive listing of HTML codes

The official HTML specification

HTTP://INFO.CERN.CH/HYPERTEXT/WWW/MARKUP/SGML.HTML the Standard Generalized Markup Language

HTTP://WWW.HAL.COM/~CONNOLLY/DRAFTS/HTML-DESIGN.HTML Dan Connolly's HTML DESIGN NOTEBOOK Dan Connolly is one of the originators of HTML.

What Is HTML?

HTML stands for Hypertext Markup Language, and it is a language (a programming language) for describing to a browser (the software application that reads HTML file) how a file ought to look and to what other files and applications the browser ought to direct its user.

If you are familiar with (a) old DOS or Apple word processors, (b) TROFF on UNIX, (c) Standard Generalized Markup Language (SGML) or (d) typesetting systems, HTML will look very familar to you. If you are not familar with any of these, read on.

HTML consists of a set of **directives**, called **tags**, combined with plain old text, to make up a HTML document. The text is the raw material of the HTML file, and the tags describe how the text should be handled.

The directives are **instructions** to something called a **browser**: the software application that reads and processes (interprets) HTML files for a user. Popular browsers include NetScape (from Mosaic Communications), NCSA Mosaic (from NCSA) and Air Mosaic from Spry.

You write HTML files like you would a word processing document, using either a plain old editor, or an HTML-aware editor (like WinWeb) or a document publishing system (like Frame or Interleaf) that can produce HTML without human intervention.

You store HTML files on a Web server.

Other people use their browsers to request the HTML files you have stored on the Web server.

When an HTML file is requested, the Web Server sends it over the network to the Web browser who has requested it.

The Web browser then (a) reads the HTML file, (b) examines the tags amd (c) decides how to display it for its user.

It is important to note that, although you are the author of HTML documents, you ultimately don't control exactly how the documents you write will be interpreted. You may, for example, specify a text string (like "What is HTML?") as a Level 1 Heading, imagining that it will be displayed in 36 point bold black text. But if I have instructed my browser to display Level 1 Headings as 18 point red italic text, that is how I will see your document.

In a nutshell then, HTML is a language for describing hypertext to a broswer, and an language for the browser itself to create a hypertext document.

How Does HTML Work?

As I mentioned earlier, you can think of HTML as playing two different roles:

- as an authoring language

- as a display language.

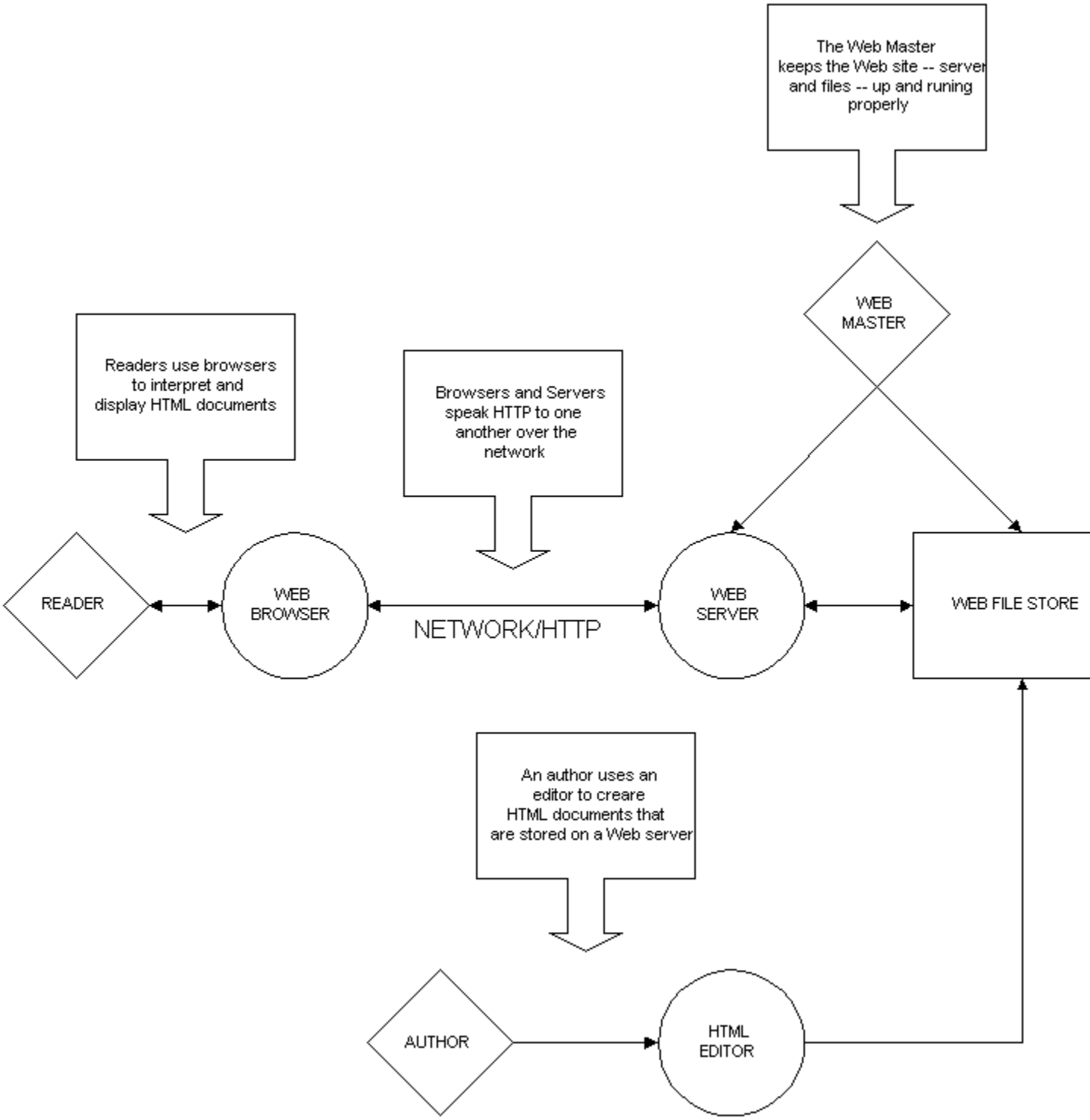
As an HTML author, you use HTML to describe how you want a particular document to look (its visual appearance) and how you want it to behave (the links between documents, the relationships among parts of a single document or a set of related documents).

You store these documents on a Web server for people to read.

When the documents are requested -- via a browser -- they are shipped from the Web server to the browser over the network. Normally, the Web server does not look at or interpret or change documents that it serves to a browser elsewhere in the network.

When the browser receives a document, it "parses" or reads the document, paying particular attention to the HTML tags (or directives). Using its own internal logic (and sometimes additional rules set up for it by its user), the broswer interprets the HTML document and displays it for its user.

The overall process looks like this:



There are some important things for you to note about this flow of work:

The HTML documents you create are not typically under your control when they are stored (they are controlled by the WebMaster) and they are not under your

control when they are read -- they are shipped over the network to the reader's browser, and -- using the browser's built-in capabilities -- the reader can do many things with your text.

You should know who your WebMaster is and understand her rules for correct use of the Web server on which you are storing your Web files.

An incorrectly-designed HTML file can go undetected until it has already been shipped over the network to a browser. To prevent wasted network bandwidth, text your files with multiple browsers (or the least functional/tolerant browser, Mosaic) before you put them in the production area of your Web server.

There are some obscure exceptions to this rule, mostly for something called "server side includes." If you know what one of these is, you don't need to be reading this document, and if you don't know, don't worry -- you will probably never need them.

The Structure Of A Hypertext Web

At the logical level, a web is a set of nodes, held together with links. When you build a web's logical design, you can treat it visually as a set of circles (nodes) connected together with lines.

However, when you move to physical design -- that is, how the Web you are created will actually look on the computer system where it is stored, that stylized set of circles and lines has to be translated into a set of files in a set of directories with some logical relationships embedded in the directory structure, directory names and file names.

Typically, you want to use directories to store relatively-self sufficient pieces of your web. I personally store all files associated with a standalone topic in a single directory with a name that reflects to the casual reader the subject of the web the directory contains.

The Structure Of An HTML Document

I think of any HTML document as having three parts:

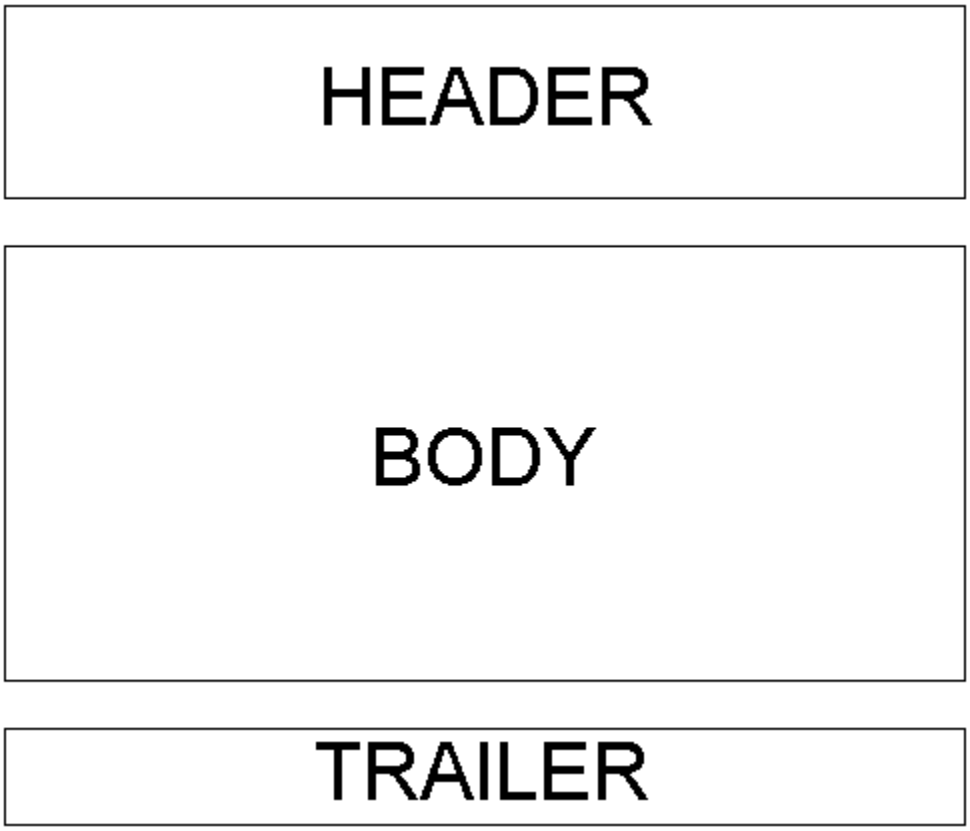
- a **header**: an area of the document that contains information about the document itself

- a **body**, which contains the matter of the document: it's content

- a **trailer**: a space used to hold administrivia.

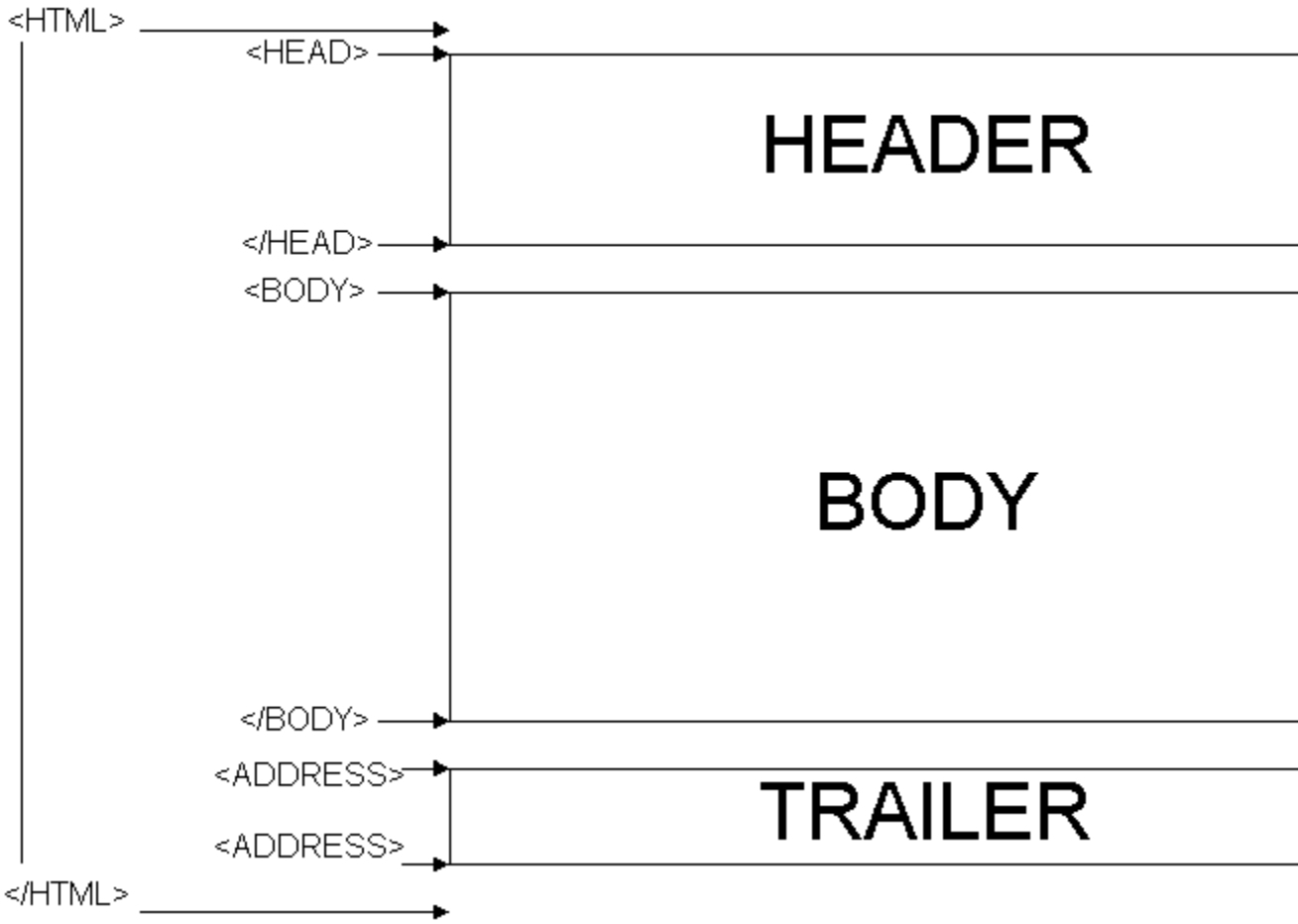
This model is convenient because it parallels other organizational metaphors we use in everyday life and in the wired world: letters and mail messages, for example, are structured this way.

Graphically, that arrangement can be expressed as follows:



The relative sizes of the three boxes are not accidental: typically the HTML document should be about 65% body, about 20% header and about 15% trailer.

The three parts of the HTML document are called out, using HTML directives, as indicated in the diagram below:



So, a perfectly structured HTML document with NO CONTENT in it would look like this:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
<ADDRESS>
</ADDRESS>
</HTML>
```

You will note that the HTML tags are **nested**: that is, they are arranged in a logical hierarchy.

```
<HTML>
      <HEAD>
      </HEAD>
```

```
        <BODY>
        </BODY>
        <ADDRESS>
        </ADDRESS>
</HTML>
```

This is the essential logical model of HTML: nesting. When you add content to one of the three parts of the document, that content is in its turnm nested inside the HTML tags that call out the beginning and end of that section of the document, as in:

```
<HTML>
        <HEAD>
                <TITLE>The Document Title</TITLE>
        </HEAD>
        <BODY>
                The body of the document
        </BODY>
        <ADDRESS>
                The trailer of the document
        </ADDRESS>
</HTML>
```

The other thing for you to remember is that certain HTML directives can only be used in certain parts of the document. For example the direction <ISINDEX>, which tells a browser that the file being read is an index, and the user should be provided with a text box in which to enter search parameters, can only be used in the header of the document. Similarly the <TITLE>...</TITLE< pair of directives, which mark the start and end of the document's title, are designed to be used in the header of the document.

Remember that no HTML directives outside the <HTML>...</HTML> tags will be interpreted by browsers, and that some browsers will become toxic (unpredictable) when they encounter HTML directives before <HTML> ot after </HTML>. For example, early on in my HTML coding, I produced a file that looked like this:

```
<!-- A comment line -->
<HTML>
...
...
</HTML>
```

The line before the <HTML> tag is a comment line; comments are just a way of making notes about the file that are not shown to users in browsers, but may be of use to you or other people editing your HTML documents. NetScape handled the comment line fine, ignoring it, but Air Mosaic became so confused by the comment line outside <HTML>...</HTML> that it refused to show any of the file, even though (other than this error) all the HTML directives in the file were legal and properly done.

Comments

Comment lines are lines that are never seen by the user and ignored by the browser.

The purpose of comment lines is to make notes on an HTML inside the HTML file itself. You can also use comments to mark of HTML code that you do not want the browser to execute.

The generic structure of a comment line is:

```
<!-- This is the comment and it can be anything at all -->
```

You should use comments to keep yourself well-organized, tell others when you do something tricky, and keep information on the file in the file. For example, the standard HTML skeleton that WebEdit produces looks like this:

```
<HTML>
<HEAD>
<!-- Created on 01-23-1995 at 00:35:59 -->
<!-- Created using WebEdit v1.1  -->
<TITLE>Minor Victorian Novelists: The George Gissing
Page</TITLE>
</HEAD>
<BODY>
<!-- PUT BODY OF HTML DOCUMENT BELOW HERE -->
<H1>Minor Victorian Novelists: The George Gissing Page</H1>


<!-- PUT BODY OF HTML DOCUMENT ABOVE HERE -->
<HR>
<ADDRESS>
Last revised on 01-23-1995 at 00:36:15 by <A
HREF="MAILTO:marc@sequent.com">Marc Demarest</A>
<!-- PUT OTHER TRAILER ELEMENTS BELOW HERE -->


<!-- PUT OTHER TRAILER ELEMENTS ABOVE HERE -->
</ADDRESS>
<HR>
</BODY>
</HTML>
```

You can see how comments are used, in this skeleton, to both keep information about the file (the comments in the HEAD section do that), and to provide directives to myself and to other HTML authors.