HotDog HTML Reference

This is an on-line reference for the syntax and use of HTML 2 elements. HotDog also supports proposed HTML 3.0 elements, but at the time of writing the HTML 3 specification has not been finalized.

Information on specific browsers, or the broader topic of 'The World Wide Web' can be obtained by reading the World Wide Web FAQ.

Also See

HTML Introduction
HTML Tutorial

The World Wide Web FAQ is posted (every four days) to the following UseNet newsgroups:

- news.answers
- comp.infosystems.www.users
- comp.infosystems.www.providers
- comp.infosystems.www.misc
- comp.infosystems.gopher
- comp.infosystems.wais

The most recent version is also always held at :
http://sunsite.unc.edu/boutell/faq/www_faq.html
The FAQ is maintained by Thomas Boutell

HTML Specification

Quick Reference

The vast range of HTML MarkUp elements specified in ″text/html; version=2.0″ Internet Media Type (RFC 1590) and MIME Content Type (RFC 1521), otherwise known as the HTML Specification, version 2.0 can be divided into seven sections. These make up the defined specification and it can be assumed that all World Wide Web user agents (Web browsers : Netscape , Mosaic, Cello, Lynx etc..) will support rendering of these elements.

Document Structure Elements
Anchor Element
Block Formatting Elements.
List Elements.
Information type and Character formatting Elements.
Image Element.
Form Elements.

Also included in this part of the reference is information purtaining to the following sections of the HTML 2.0 specification.

Character Data
Obsolete and Proposed Elements

Details of elements that will be included in the HTML 3.0 specification, but which are supported by currently available browsers, can be found here :

HTML 3.0 elements

NOTE : While a proposal for the HTML 3.0 specification *is* available, this reference will only be updated when browsers support new HTML elements. That way it will remain current with available browsers, avoiding confusion over which elements can be used.

Wherever this symbol :  appears, a screen shot showing typical rendering of the element in question is available. To see the screenshot, click the picture.

Exactly...just like you did then.

Document Structure Elements

Quick Reference

These elements are required within a HTML document. Apart from the prologue document identifier, they represent the only HTML elements which are explicitly required for a document to conform to the standard.

The essential document structure elements are :

<HTML> ... </HTML>
<HEAD> ... </HEAD>
<BODY> ... </BODY>

In order to identify a document as HTML, each HTML document should start with the prologue:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN//2.0">.
```

However, it is worth noting that if the document doesn´t contain this type declaration, a HTML user agent should infer it.

<HTML> ... </HTML>

Quick Reference

This element identifies the document as containing HTML elements. It should immediately follow the prologue document identifier and serves to surround all of the remaining text, including all other elements. That is, the document should be constructed thus :

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN//2.0">
<HTML>
Here is all the rest of the document, including any elements.
</HTML>
```

The HTML element is not visible upon HTML user agent rendering and can contain only the <HEAD> and <BODY> elements.

<HEAD> ... </HEAD>

Quick Reference

The head of an HTML document is an unordered collection of information about the document. It requires the Title element between `<HEAD>` and `</HEAD>` elements thus :

```
<HEAD>
<TITLE> Introduction to HTML </TITLE>
</HEAD>
```

The `<HEAD>` and `</HEAD>` elements do not directly affect the look of the document when rendered.

The following elements are related to the head element. While not directly affecting the look of the document when rendered, they do provide (if used) important information to the HTML user agent.

<BASE> - Allows base address of HTML document to be specified
<ISINDEX>- Allows keyword searching of the document
<LINK>- Indicate realationships between documents
<NEXTID>- Creates unique document identifiers
<TITLE>- Specifies the title of the document
<META> - Specifies document information useable by server/clients.

NOTE : The Title element is the ***only*** element described here that is ***required*** as part of the Head of a HTML document.

<BODY> ... </BODY>

Quick Reference

The body of a HTML document contains all the text and images that make up the page, together with all the HTML elements that provide the control/formatting of the page. The format is :

```
<BODY>
The document included here
</BODY>
```

The `<BODY>` and `</BODY>` elements do not directly affect the look of the document when rendered, although they are ***required*** in order for the document to conform to the specification standard.

The `<BODY>` element has been enhanced in recent Netscape versions. It is now possible to control the document background.

<BASE...>

Quick Reference

The Base 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 realtive to this base address.

Where the base address is not specified, the HTML user agent uses the URL it used to access the document to resolve any relative URLs.

The Base element has one attribute, `HREF`, which identifies the URL.

<ISINDEX...>

See Also Quick Reference

The Isindex element tells the HTML user agent that the document is an index document. As well as reading it, the reader may use a keyword search.

The document can be queried with a keyword search by adding a question mark to the end of the document address, followed by a list of keywords separated by plus signs.

NOTE : The Isindex element is usually generated automatically by a server. If added manually to a HTML document, the HTML user agent assumes that the server can handle a search on the document. To use the Isindex element, the server must have a search engine that supports this element.

NOTE : The `<ISINDEX>` element has been Netscape enhanced.

Netscape enhancement to <ISINDEX>

<LINK...>

Quick Reference

The Link element indicates a relationship between the document and some other object. A document may have any number of Link elements.

The Link element is empty (does not have a closing element), but takes the same attributes as the Anchor element.

Typical uses are to indicate authorship, realted 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.

<NEXTID...>

Quick Reference

The Nextid element is a parameter read by and generated by text editing software to create unique identifiers. This element takes a single attribute which is th enext document-winde alpha-numeric identifier to be allocated of the form z123 :

```
<NEXTID N=Z127>
```

When modifying a document, existing anchor identifiers should not be reused, as these identifiers may be referenced by other documents. Human writers of HTML usually use mnemonic alphabetic identifiers. HTML user agents may ignore the Nextid element. Support for the Nextid element does not impact HTML user agents in any way.

<TITLE> ... </TITLE>

Quick Reference

Every HTML document must have a Title element. The title should identify the contents of the document and in a global context, and may be used in history lists and as a label for the windows displaying the document. Unlike headings, titles are not typically rendered in the text of a document itself.

The Title element must occur within the head of the documet and may not contain anchors, paragraph elements, or highlighting. Only one title is allowed in a document.

> ***NOTE :*** The length of a title is not limited, however, long titles may be truncated in some applications. To minimize the possibility, titles should be fewer than 64 characters. Also keep in mind that a short title, such as 'Introduction' may be meaningless out of context. An example of a meaningful title might be 'Introduction to HTML elements'

This is the ***only*** element that is ***required*** within the Head element. The other elements described are optional and can be implemented when appropriate

```
<HEAD>
<TITLE> Introduction to HTML </TITLE>
</HEAD>
```

<META...>

Attributes See Also Quick Reference

The Meta element is used within the Head element to embed document meta-information not defined by other HTML elements. Such information can be extracted by servers/clients for use in identifying, indexing and cataloging specialised document meta-information.

Although it is generally preferable to use named elements that have well defined semantics for each type of meta-information, such as title, this element is provided for situations where strict SGML parsing is necessary and the local DTD is not extensible.

In addition, HTTP servers can read the content of the document head to generate response headers corresponding to any elements defining a value for the attribute `HTTP-EQUIV`. This provides document authors a mechanism (not necessarily the preferred one) for identifying information that should be included in the respomse headers for an HTTP request.

Attributes of the Meta element :

HTTP-EQUIV

This attribute binds the element to an HTTP response header. If the semantics of the HTTP response header named by this attribute is known, then the contents can be processed based on a well-defined syntactic mapping wether or not the DTD includes anything about it. HTTP header names are not case sensitive. If not present, the `NAME` attribute should be used to identify this meta-information and it should not be used within an HTTP response header.

NAME

Meta-information name. If the name attribute is not present, then name can be assumed equal to the value `HTTP-EQUIV`.

CONTENT

The meta-information content to be associated with the given name and/or HTTP response header.

Examples :

If the document contains :

```
<META HTTP-EQUIV="Expires" CONTENT="Tue, 04 Dec 1993 21:29:02 GMT">
<META HTTP-EQUIV="Keywords" CONTENT="Fred, Barney">
<META HTTP-EQUIV="Reply-to" CONTENT="fielding@ics.uci.edu <Roy
   Fielding">
```

Then the HTTP response header would be :

Expires: Tue, 04 Dec 1993 21:29:02 GMT
Keywords: Fred, Barney
Reply-to: fielding@ics.uci.edu (Roy Fielding)

When the `HTTP-EQUIV` attribute is not present, the server should not generate an HTTP response header for this meta-information. e.g,

```
<META NAME="IndexType" CONTENT="Service">
```

Do *not* use the Meta element to define information that should be associtaed with an existing HTML element.

Example of an inappropriate use of the Meta element :

```
<META NAME="Title" CONTENT="The Etymology of Dunsel">
```

Do *not* name an `HTTP-EQUIV` equal to a responsive header that should typically only be generated by the HTTP server. Some inappropriate names are ˝Server˝, ˝Date˝ and ˝Last-modified˝. Whether a name is inappropriate depends on the particular server implementation. It is recommended that servers ignore any Meta elements that specify HTTP-equivalents equal (case-insensitively) to their own reserved response headers.

The `META` element is particularly useful for constructing Dynamic documents.

The following attributes are allowed within the `<META ...>` element.

- HTTP-EQUIV
- NAME
- CONTENT

Dynamic Documents

Block Formatting Elements

Quick Reference

Block formatting elements are used for the formatting of whole blocks of text within a HTML document, rather than single characters. They should all (if present) be within the body of the document.

The essential block formatting elements are :

<ADDRESS> ... </ADDRESS> - Format an address section
<H1> ... </H1> - Format six levels of heading
<HR> - Renders a hard line on the page

 - Force a line break
<P> ... </P> - Specify what text constitutes a paragraph
<PRE> ... </PRE> - Use text already formatted
<BLOCKQUOTE> ... </BLOCKQUOTE> - To quote text from another source

<ADDRESS> ... </ADDRESS>

Quick Reference

The Address element specifies such information as address, signature and authorship, often at the top or bottom of a document.

Typically, an Address is rendered in an italic typeface and may be indented. The Address element implies a paragraph break before and after.

Example of use:

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

Newsletter editor
J.R. Brown
JimquickPost News, Jumquick, CT 01234
Tel (123) 456 7890

<H1> ... </H1> Headings

See Also Quick Reference

HTML defines six levels of heading. A Heading element implies all the font changes, paragraph breaks before and after, and white space necessary to render the heading.

The highest level of headings is `<H1>`, followed by `<H2>` ... `<H6>`.

Example of use:

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

The rendering of headings is determined by the HTML user agent, but typical renderings are:

`<H1> ... </H1>`
Bold, very-large font, centered. One or two blank lines above and below.

`<H2> ... </H2>`
Bold, large font, flush-left. One or two blank lines above and below.

`<H3> ... </H3>`
Italic, large font, slightly indented from the left margin. One or two blank lines above and below.

`<H4> ... </H4>`
Bold, normal font, indented more than H3. One blank line above and below.

`<H5> ... </H5>`
Italic, normal font, indented as H4. One blank line above.

`<H6> ... </H6>`
Bold, indented same as normal text, more than H5. One blank line above.

Although heading levels can be skipped (for example, from H1 to H3), this practice is discouraged as skipping heading levels may produce unpredictable results when generating other representations from HTML.

NOTE : This element is to be enhanced in HTML 3.0. Alignment attributes are to be added.

NOTE : These Headings are a screenshot of Mosaic 2.0 beta 4 heading rendering, using a default installation. The exact format can be altered within Mosaic, this screenshot is provided to show the six different headings in relation to each other.

This is Heading 1

This is Heading 2

This is Heading 3

This is Heading 4

This is Heading 5

This is Heading 6

HTML 3.0 - Headings

<HR>

See Also Quick Reference

A Horizontal Rule element is a divider between sections of text such as a full width horizontal rule or equivalent graphic.

Example of use:

```
<HR>
<ADDRESS>February 8, 1995, CERN</ADDRESS>
</BODY>
```

NOTE : The `<HR>` element has been Netscape Enhanced

See the Netscape enhanced `<HR>` page for screenshots.

Netscape enhancements to <HR>

See Also Quick Reference

The Line Break element specifies that a new line must be started at the given point. A new line indents the same as that of line-wrapped text.

Example of use:

```
<P>
Pease porridge hot<BR>
Pease porridge cold<BR>
Pease porridge in the pot<BR>
Nine days old.
```

NOTE : The `<BR>` element has been Netscape enhanced.

Netscape enhancements to

<P> ... </P>

See Also Quick Reference

The Paragraph element indicates a paragraph. The exact indentation, leading, etc. of a paragraph is not defined and may be a function of other elements, style sheets, etc.

Typically, paragraphs are surrounded by a vertical space of one line or half a line. This is typically not the case within the Address element and or is never the case within the Preformatted Text element. With some HTML user agents, the first line in a paragraph is indented.

Example of use:

```
<H1>This Heading Precedes the Paragraph</H1>
<P>This is the text of the first paragraph.
<P>This is the text of the second paragraph. Although you do not need
to start paragraphs on new lines, maintaining this convention
facilitates document maintenance.
<P>This is the text of a third paragraph.
```

NOTE : This element is to be enhanced in HTML 3.0. Alignment attributes are to be added. See this page for screenshots.

HTML 3.0 - Paragraph

<PRE> ... </PRE>

Quick Reference

The Preformatted Text element presents blocks of text in fixed-width font, and so is suitable for text that has been formatted on screen.

The `<PRE>` element may be used with the optional `WIDTH` attribute, which is a Level 1 feature. The `WIDTH` attribute specifies the maximum number of characters for a line and allows the HTML user agent to select a suitable font and indentation. If the `WIDTH` attribute is not present, a width of 80 characters is assumed. Where the `WIDTH` attribute is supported, widths of 40, 80 and 132 characters should be presented optimally, with other widths being rounded up.

Within preformatted text:

- Line breaks within the text are rendered as a move to the beginning of the next line.
- The `<P>` element 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 that define paragraph formatting (headings, address, etc.) must not be used.
- The horizontal tab character (encoded in US-ASCII and ISO-8859-1 as decimal 9) 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.

NOTE: References to the "beginning of a new line" do not imply that the renderer is forbidden from using a constant left indent for rendering preformatted text. The left indent may be constrained by the width required.

Example of use:

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

NOTE: Within a Preformatted Text element, the constraint that the rendering must be on a fixed horizontal character pitch may limit or prevent the ability of the HTML user agent to render highlighting elements specially.

<BLOCKQUOTE> ... </BLOCKQUOTE>

Quick Reference

The Blockquote element is used to contain text quoted from another source.

A typical rendering might be a slight extra left and right indent, and/or italic font. The Blockquote element causes a paragraph break, and typically provides space above and below the quote.

Single-font rendition may reflect the quotation style of Internet mail by putting a vertical line of graphic characters, such as the greater than symbol (>), in the left margin.

Example of use:

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

NOTE : This screenshot shows how Netscape 1.1 would display text using the `<BLOCKQUOTE>` element. Renderings using different HTML user agents may differ.

I think the poem ends

> Soft you now, the fair Ophelia.
> Nymph, in thy orisons, be all my
> sins remembered.

but I am not sure.

<A...> ... </A> Anchor

Atrributes Quick Reference

An Anchor element is a marked text that is the start and/or destination of a hypertext link. Anchor elements are defined by the `<A>` element. The `<A>` element accepts several attributes, but either the `NAME` or `HREF` attribute is required.

Attributes of the `<A>` element :

HREF

If the HREF attribute is present, the text between the opening and closing anchor elements becomes hypertext. If this hypertext is selected by readers, they are moved to another document, or to a different location in the current document, whose network address is defined by the value of the HREF attribute.

Example :

```
See <A HREF="http://www.hal.com/">HaL</A>'s information for more
details.
```

In this example, selecting "HaL" takes the reader to a document located at http://www.hal.com. The format of the network address is psecified in the URI specification for print readers.

With the `HREF` attribute, the form `HREF="#identifier"` can refer to another anchor in the same document.

Example :

```
The <A HREF="document.html#glossary">glossary</A> defines terms used in
the document.
```

In this example, selecting "glossary" takes the reader to another anchor (i.e. `<A NAME="glossary">Glossary</A>`) in the same document (document.html). The `NAME` attribute is described below. If the anchor is in another document, the `HREF` attribute may be relative to the document's address or the specified base address.

NAME

If present, the `NAME` attribute allows the anchor to be the target of a link. The value of the `NAME` attribute is an identifier for the anchor. Identifiers are arbitrary strings but must be unique within the HTML document.

Example of use:

```
<A NAME=coffee>Coffee</A> is an exmple of...
An example of this is <A HREF=#coffee>coffee</A>.
```

Another document can then make a reference explicitly to this anchor by putting the identifier after the address, separated by a has sign :

```
<A NAME=drinks.html#coffee>
```

TITLE

The Title attribute is informational only. If present, the Title attribute should provide the title of the document whose address is given by the `HREF` attribute.

The Title attribute is useful for at least two reasons. The HTML user agent may display the title of the document prior to retrieving it, for example, as a margin note or on a small box while the mouse is over the anchor, or while the document is being loaded. Another reason is that documents that are not marked up text, such as graphics, plain text and Gopher menus, do not have titles. The `TITLE` attribute can be used to provide a title to such documents. When using the `TITLE` attribute, the title should be valid and unique for the destination document.

REL

The `REL` attribute gives the relationship(s) described by the hypertext link from the anchor to the target. 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. The `REL` attribute is only used when the `HREF` attribute is present.

REV

The `REV` attribute is the same as the `REL` attribute, 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.

URN

If present, the `URN` attribute specifies a uniform resource name (URN) for a target document. The format of URNs is under discussion (1994) by various working groups of the Internet Engineering Task Force.

METHODS

The `METHODS` attributes of anchors and links provide information about the functions that 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 HTML user agent may chose a different rendering as a function of the methods allowed; for example, something that is searchable may get a different icon.

The value of the `METHODS` attribute is a comma separated list of HTTP methods supported by the object for public use.

The following attributes are all allowed within the `<A ...>` element

- HREF
- NAME
- TITLE
- REL
- REV
- URN
- METHODS

List Elements

Quick Reference

HTML supports several types of lists, all of which may be nested. If used they should be present in the body of a HTML document.

<DL> ... </DL>

Attributes See Also Quick Reference

A definition list is a list of terms and corresponding definitions. Definition lists are typically formatted with the term flush-left and the definition, formatted paragraph style, indented after the term.

Example of use:

```
<DL>
<DT>Term<DD>This is the definition of the first term.
<DT>Term<DD>This is the definition of the second term.
</DL>
```

If the `<DT>` term does not fit in the `<DT>` column (one third of the display area), it may be extended across the page with the `<DD>` section moved to the next line, or it may be wrapped onto successive lines of the left hand column.

Single occurrences of a `<DT>` element without a subsequent `<DD>` element are allowed, and have the same significance as if the `<DD>` element had been present with no text.

The opening list element must be `<DL>` and must be immediately followed by the first term (`<DT>`).

The definition list type can take the **COMPACT** attribute, which suggests that a compact rendering be used, because the list items are small and/or the entire list is large.

Unless you provide the `COMPACT` attribute, the HTML user agent may leave white space between successive `<DT>, <DD>` pairs.The `COMPACT` attribute may also reduce the width of the left-hand (`<DT>`) column.

If using the `COMPACT` attribute, the opening list element must be `<DL COMPACT>`, which must be immediately followed by the first `<DT>` element:

```
<DL COMPACT>
<DT>Term<DD>This is the first definition in compact format.
<DT>Term<DD>This is the second definition in compact format.
</DL>
```

Term

This is the definition of the first term.

Term

This is the definition of the second term.

Term This is the first definition in compact format.
Term This is the second definition in compact format.

The following are all allowed
within the <DL> element.

<DT> : Definition list Term

<DD> : Definition list definition.

COMPACT

<DIR> : Directory list
<MENU> : Menu list
<OL> : Ordered list
<UL> : Unordered list

<DIR> ... </DIR>

See Also Quick Reference

A Directory List element is used to present a list of items containing up to 20 characters each. Items in a directory list may be arranged in columns, typically 24 characters wide. If the HTML user agent can optimize the column width as function of the widths of individual elements, so much the better.

A directory list must begin with the `<DIR>` element which is immediately followed by a `<LI>` (list item) element:

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

- A-H
- I-M
- M-R
- S-Z

<DL> : Definition list
<MENU> : Menu list
<OL> : Ordered list
<UL> : Unordered list

<MENU> ... </MENU>

See Also Quick Reference

A menu list is a list of items with typically one line per item. The menu list style is more compact than the style of an unordered list.

A menu list must begin with a `<MENU>` element which is immediately followed by a `<LI>` (list item) element:

```
<MENU>
<LI>First item in the list.
<LI>Second item in the list.
<LI>Third item in the list.
</MENU>
```

- First item in the list.
- Second item in the list.
- Third item in the list.

<DL> : Definition list
<DIR> : Directory list
<OL> : Ordered list
<UL> : Unordered list

<OL> ... </OL>

See Also Quick Reference

The Ordered List element is used to present a numbered list of items, sorted by sequence or order of importance.

An ordered list must begin with the `<OL>` element which is immediately followed by a `<LI>` (list item) element:

```
<OL>
<LI>Click the Web button to open the Open the URL window.
<LI>Enter the URL number in the text field of the Open URL
window. The Web document you specified is displayed.
<LI>Click highlighted text to move from one link to another.
</OL>
```

The Ordered List element can take the `COMPACT` attribute, which suggests that a compact rendering be used.

NOTE : The `<OL>` element has been Netscape enhanced.

1. Click the Web button to open the Open the URL window.
2. Enter the URL number in the text field of the Open URL window. The Web document you specified is displayed.
3. Click highlighted text to move from one link to another.

<DL> : Definition list
<DIR> : Directory list
<MENU> : Menu list
<UL> : Unordered list
Netscape enhancement to <OL>

<UL> ... </UL>

See Also Quick Reference

The Unordered List element is used to present a list of items which is typically separated by white space and/or marked by bullets.

An unordered list must begin with the `<UL>` element which is immediately followed by a `<LI>` (list item) element:

```
<UL>
<LI>First list item
<LI>Second list item
<LI>Third list item
</UL>
```

The Unordered List element can take the `COMPACT` attribute, which suggests that a compact rendering be used.

NOTE : The `<UL>` element has been Netscape enhanced

- First list item
- Second list item
- Third list item

<DL> : Definition list
<DIR> : Directory list
<MENU> : Menu list
<OL> : Ordered list
Netscape enhancement to <UL>

Information Type and Character Formatting Elements

Quick Reference

The following information type and character formatting elements are supported in the HTML 2.0 specification

Information type elements:
(***NOTE :*** Different information type elements may be rendered in the same way)

<CITE> ... </CITE> - Citation
<CODE> ... </CODE> - An example of Code
<EM> ... </EM> - Emphasis
<KBD> ... </KBD> - User typed text
<SAMP> ... </SAMP> - A sequence of literal characters
<STRONG> ... </STRONG> - Strong typographic emphasis
<VAR> ... </VAR> - Indicates a variable name

Character formatting elements

<B> ... </B> - Boldface type
<I> ... </I> - Italics
<TT> ... </TT> - TypeType (or Teletype)

Character-level elements are used to specify either the logical meaning or the physical appearance of marked text without causing a paragraph break. Like most other elements, character-level elements include both opening and closing elements. Only the characters between the elements are affected:

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

Character-level elements may be ignored by minimal HTML applications.

Character-level elements are interpreted from left to right as they appear in the flow of text. Level 1 HTML user agents must render highlighted text distinctly from plain text. Additionally, `<EM>` content must be rendered as distinct from `<STRONG>` content, and `<B>` content must rendered as distinct from `<I>` content.

Character-level elements may be nested within the content of other character-level elements; however, HTML user agents are not required to render nested character- level elements distinctly from non-nested elements:

```
plain <B>bold <I>italic</I></B>
```

may be rendered the same as

```
plain <B>bold </B><I>italic</I>
```

Note that typical renderings for information type elements vary between applications. If a specific rendering is necessary, for example, when referring to a specific text attribute as in "The italic parts are mandatory", a formating element can be used to ensure that the intended rendering is used where possible.

<CITE> ... </CITE>

Quick Reference

The Citation element specifies a citation; typically rendered as italics.

e.g. : `This sentence, containing a <CITE>citation reference</CITE> would look like:`

This sentence, containing a *citation reference* would look like:

<CODE> ... </CODE>

Quick Reference

The Code element indicates an example of code; typically rendered as monospaced . Do not confuse with the Preformatted Text element.

e.g. : `This sentence contains an <CODE>example of code</CODE>.  It would look like :`

This sentence contains an `example of code`. It would look like :

<EM> ... </EM>

Quick Reference

The Emphasis element indicates typographic emphasis, typically rendered as italics.

e.g. : `The <EM>Emphasis</EM> element typically renders as Italics.`

would render :

The *Emphasis* element typically render as Italics.

<KBD> ... </KBD>

Quick Reference

The Keyboard element indicates text typed by a user; typically rendered as monospaced . It might commonly be used in an instruction manual.

e.g. : `The text inside the <KBD>KBD element, would typically</KBD> render as monospaced font.`

would render as :

The text inside the `KBD element would typically` render as monospaced font.

<SAMP> ... </SAMP>

Quick Reference

The Sample element indicates a sequence of literal characters; typically rendered as monospaced.

e.g. : `A sequence of <SAMP>literal characters</SAMP> commonly renders as monospaced font.`

would render as :

A sequence of `literal characters` commonly renders as monospaced font.

<STRONG> ... </STRONG>

Quick Reference

The Strong element indicates strong typographic emphasis, typically rendered in bold.

e.g. : `The instructions <STRONG>must be read</STRONG> before continuing.`

would be rendered as :

The instructions **must be read** before continuing.

<VAR> ... </VAR>

Quick Reference

The Variable element indicates a variable name; typically rendered as italic.

e.g. : `When coding, <VAR>LeftIndent()</VAR> must be a variable`

would render as :

When coding *LeftIndent()* must be a variable.

<B> ... </B>

Quick Reference

The Bold element specifies that the text should be rendered in boldface, where available. Otherwise, alternative mapping is allowed.

e.g. : `The instructions <B>must be read</B> before continuing.`

would be rendered as :

The instructions **must be read** before continuing.

<I> ... </I>

Quick Reference

The Italic element specifies that the text should be rendered in italic font, where available. Otherwise, alternative mapping is allowed.

e.g. : `Anything between the <I>I elements</I> should be italics.`

would render as :

Anything between the *I elements* should be italics.

<TT> ... </TT>

Quick Reference

The Teletype element specifies that the text should be rendered in fixed-width typewriter font.

e.g : `Text between the <TT> typetype elements</TT> should be rendered in fixed width typewriter font.`

would render as :

Text between the `typetype elements` should be rendered in fixed width typewriter font.

<IMG...> In-line images

Attributes See Also Quick Reference

The Image element is used to incorporate in-line graphics (typically icons or small graphics) into an HTML document. This element cannot be used for embedding other HTML text.

HTML user agents that cannot render in-line images ignore the Image element unless it contains the ALT attribute. Note that some HTML user agents can render linked graphics but not in-line graphics. If a graphic is essential, you may want to create a link to it rather than to put it in-line. If the graphic is not essential, then the Image element is appropriate.

The Image element, which is empty (no closing element), has these attributes:

ALIGN

The `ALIGN` attribute accepts the values `TOP` or `MIDDLE` or `BOTTOM`, which specifies if the following line of text is aligned with the top, middle, or bottom of the graphic.

ALT

Optional text as an alternative to the graphic for rendering in non-graphical environments. Alternate text should be provided whenever the graphic is not rendered. Alternate text is mandatory for Level 0 documents. Example of use:

```
<IMG SRC="triangle.gif" ALT="Warning:"> Be sure to read these
instructions.
```

ISMAP

The `ISMAP` (is map) attribute identifies an image as an image map. Image maps are graphics in which certain regions are mapped to URLs. By clicking on different regions, different resources can be accessed from the same graphic. Example of use:

```
<A HREF="http://machine/htbin/imagemap/sample">
<IMG SRC="sample.gif" ISMAP>
</A>
```

NOTE : To be able to employ image maps in HTML documents, the HTTP server which will be controlling document access must have the correct cgi-bin software installed to control image map behaviour.

SRC

The value of the `SRC` attribute is the URL of the document to be embedded; only images can be embedded, not HTML text. Its syntax is the same as that of the `HREF` attribute of the <A> element. `SRC` is mandatory. Image elements are allowed within anchors.

Example of use:

```
<IMG SRC ="triangle.gif">Be sure to read these instructions.
```

NOTE : The `<IMG>` element has received possibly the largest Netscape enhancement.

The following attributes are allowed within the `<IMG ...>` element

- ALIGN
- ALT
- ISMAP
- SRC

Netscape enhancement to <IMG ...>


This text is aligned at the "top" of the picture


This text is aligned at the "middle" of the picture


This text is aligned at the "bottom" of the picture

Forms

Quick Reference

The inclusion of the Form elements, allowing user input/feedback on HTML documents, was the major difference between the HTML specification 2.0 and its predecessors.

They are created by placing input fields within paragraphs, preformatted/literal text and lists. This gives considerable flexibility in designing the layout of forms.

The following elements are used to create forms :

<FORM> ... </FORM> - A form within a document
<INPUT ...> ... </INPUT> - One input field
<OPTION> - One option within a Select element.
<SELECT> ... <SELECT> - A selection from a finite set of options
<TEXTAREA ...> ... </TEXTAREA> - A multi-line input field

Each variable field is defined by an `INPUT`, `TEXTAREA`, or `OPTION` element and must have a `NAME` attribute to identify its value in the data returned when the form is submitted.

Example of use (a questionnaire form):

```
<H3>Sample Questionnaire</H3>
<P>Please fill out this questionnaire:
<FORM METHOD="POST" ACTION="http://www.hal.com/sample">
<P>Your name: <INPUT NAME="name" size="48">
<P>Male <INPUT NAME="gender" TYPE=RADIO VALUE="male">
<P>Female <INPUT NAME="gender" TYPE=RADIO VALUE="female">
<P>Number in family: <INPUT NAME="family" TYPE=text>
<P>Cities in which you maintain a residence:
<UL>
<LI>Kent <INPUT NAME="city" TYPE=checkbox VALUE="kent">
<LI>Miami <INPUT NAME="city" TYPE=checkbox VALUE="miami">
<LI>Other <TEXTAREA NAME="other" cols=48 rows=4></textarea>
</UL>
Nickname: <INPUT NAME="nickname" SIZE="42">
<P>Thank you for responding to this questionnaire.
<P><INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FORM>
```

In the example above, the `<P>` and `<UL>` elements have been used to lay out the text and input fields. The HTML user agent is responsible for handling which field will currently get keyboard input.

Many platforms have existing conventions for forms, for example, using Tab and Shift keys to move the keyboard focus forwards and backwards between fields, and using the Enter key to submit the form. In the example, the `SUBMIT` and `RESET` buttons are specified explicitly with special purpose

fields. The SUBMIT button is used to e-mail the form or send its contents to the server as specified by the ACTION attribute, while RESET resets the fields to their initial values. When the form consists of a single text field, it may be appropriate to leave such buttons out and rely on the Enter key.

The Input element is used for a large variety of types of input fields. To let users enter more than one line of text, use the Textarea element.

Form-based file upload in HTML

There is currently another IETF Internet Draft (draft-ietf-html-fileupload-01.txt) that describes the suggested implementation of form based upload in HTML forms. This suggestion is currently not supported by any browser and makes no suggestion as to whether it should be included inHTML 2.0, 2.1 or 3.0. Hence it will not be covered in any detail.

Essentially, it is suggested to add a FILE option to the TYPE attribute of the INPUT element, allow an ACCEPT attribute for the INPUT element (which is a list of media types or type patterns allowed for the input) and to allow the ENCTYPE of a from to be multipart/form-data.

Such additions to the <FORM> element could prove invaluable for example, for companies providing technical support, or service providers, requesting data files.

Sample Questionnaire

Please fill out this questionnaire:

Your name:

Male ○

Female ○

Number in family:

Cities in which you maintain a residence:

- Kent ☐
- Miami ☐
- Other

Nickname:

Thank you for responding to this questionnaire.

<FORM> ... </FORM>

Attributes See Also Quick Reference

The Form element is used to delimit a data input form. There can be several forms in a single document, but the Form element can't be nested.

The **ACTION** attribute is a URL specifying the location to which the contents of the form is submitted to elicit a response. If the ACTION attribute is missing, the URL of the document itself is assumed. The way data is submitted varies with the access protocol of the URL, and with the values of the **METHOD** and **ENCTYPE** attributes.

In general:

- the METHOD attribute selects variations in the protocol.
- the ENCTYPE attribute specifies the format of the submitted data in case the protocol does not impose a format itself.

The Level 2 specification defines and requires support for the HTTP access protocol only.

When the ACTION attribute is set to an HTTP URL, the METHOD attribute must be set to an HTTP method as defined by the HTTP method specification in the IETF draft HTTP standard. The default METHOD is GET, although for many applications, the POST method may be preferred. With the post method, the ENCTYPE attribute is a MIME type specifying the format of the posted data; by default, is application/x-www-form-urlencoded.

Under any protocol, the submitted contents of the form logically consist of name/value pairs. The names are usually equal to the NAME attributes of the various interactive elements in the form.

> ***NOTE:*** The names are not guaranteed to be unique keys, nor are the names of form elements required to be distinct. The values encode the user's input to the corresponding interactive elements. Elements capable of displaying a textual or numerical value will return a name/value pair even when they receive no explicit user input.

The following are all attributes of the `<FORM>` element.

- `ACTION`
- `METHOD`
- `ENCTYPE`

All are detailed on this page.

Other `<FORM>` elements :

<INPUT> Element
<OPTION> Element
<SELECT> Element
<TEXTAREA> Element

Forms - <INPUT>

Attributes See Also Quick Reference

The Input element represents a field whose contents may be edited by the user.

Attributes of the Input element:

ALIGN

Vertical alignment of the image. For use only with `TYPE=IMAGE` in HTML level 2. The possible values are exactly the same as for the `ALIGN` attribute of the image element.

CHECKED

Indicates that a checkbox or radio button is selected. Unselected checkboxes and radio buttons do not return name/value pairs when the form is submitted.

MAXLENGTH

Indicates the maximum number of characters that can be entered into a text field. This can be greater than specified by the `SIZE` attribute, in which case the field will scroll appropriately. The default number of characters is unlimited.

NAME

Symbolic name used when transferring the form's contents. The `NAME` attribute is required for most input types and is normally used to provide a unique identifier for a field, or for a logically related group of fields.

SIZE

Specifies the size or precision of the field according to its type. For example, to specify a field with a visible width of 24 characters:

```
INPUT TYPE=text SIZE="24"
```

SRC

A URL or URN specifying an image. For use only with `TYPE=IMAGE` in HTML Level 2.

TYPE

Defines the type of data the field accepts. Defaults to free text. Several types of fields can be defined with the type attribute:

CHECKBOX : Used for simple Boolean attributes, or for attributes that can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same name. Each selected checkbox generates a separate name/value pair in the submitted data, even if this results in duplicate names. The default value for checkboxes is "on".

HIDDEN : No field is presented to the user, but the content of the field is sent with the submitted form. This value may be used to transmit state information about client/server interaction.

IMAGE : An image field upon which you can click with a pointing device, causing the form to be

immediately submitted. The coordinates of the selected point are measured in pixel units from the upper-left corner of the image, and are returned (along with the other contents of the form) in two name/value pairs. The x-coordinate is submitted under the name of the field with .x appended, and the y- coordinate is submitted under the name of the field with .y appended. Any `VALUE` attribute is ignored. The image itself is specified by the `SRC` attribute, exactly as for the Image element.

> ***NOTE:*** In a future version of the HTML specification, the `IMAGE` functionality may be folded into an enhanced `SUBMIT` field.

PASSWORD is the same as the `TEXT` attribute, except that text is not displayed as it is entered.

RADIO is used for attributes that accept a single value from a set of alternatives. Each radio button field in the group should be given the same name. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit `VALUE` attribute.

RESET is a button that when pressed resets the form's fields to their specified initial values. The label to be displayed on the button may be specified just as for the `SUBMIT` button.

SUBMIT is a button that when pressed submits the form. You can use the `VALUE` attribute to provide a non- editable label to be displayed on the button. The default label is application-specific. If a `SUBMIT` button is pressed in order to submit the form, and that button has a `NAME` attribute specified, then that button contributes a name/value pair to the submitted data. Otherwise, a `SUBMIT` button makes no contribution to the submitted data.

TEXT is used for a single line text entry fields. Use in conjunction with the `SIZE` and `MAXLENGTH` attributes. Use the Textarea element for text fields which can accept multiple lines.

VALUE

The initial displayed value of the field, if it displays a textual or numerical value; or the value to be returned when the field is selected, if it displays a Boolean value. This attribute is required for radio buttons.

The following are all allowed within the `<INPUT>` element.

```
ALIGN
CHECKED
MAXLENGTH
NAME
SIZE
SRC
TYPE :
   CHECKBOX; HIDDEN; IMAGE;
   PASSWORD; RADIO; RESET;
   SUBMIT; TEXT
VALUE
```

Other `<FORM>` elements :

<FORM> Element
<OPTION> Element
<SELECT> Element
<TEXTAREA> Element

Forms - <OPTION>

Attributes See Also Quick Reference

The Option element can only occur within a Select element. It represents one choice, and can take these attributes:

DISABLED

Proposed.

SELECTED

Indicates that this option is initially selected.

VALUE

When present indicates the value to be returned if this option is chosen. The returned value defaults to the contents of the Option element.

The contents of the Option element is presented to the user to represent the option. It is used as a returned value if the `VALUE` attribute is not present.

The following are all allowed within the `<OPTION>` element

- `DISABLED`
- `SELECTED`
- `VALUE`

All are detailed on this page .

Other `<FORM>` elements :

<FORM> Element
<INPUT> Element
<SELECT> Element
<TEXTAREA> Element

Forms - <SELECT ...> ... </SELECT>

Attributes See Also Quick Reference

The Select element allows the user to chose one of a set of alternatives described by textual labels. Every alternative is represented by the Option element.

Attributes are:

ERROR

Proposed.

MULTIPLE

The `MULTIPLE` attribute is needed when users are allowed to make several selections, e.g. `<SELECT MULTIPLE>`.

NAME

Specifies the name that will submitted as a name/value pair.

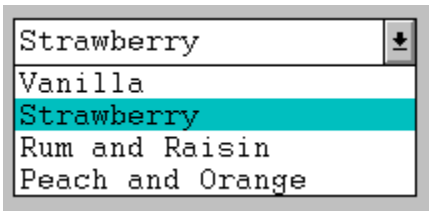
SIZE

Specifies the number of visible items. If this is greater than one, then the resulting form control will be a list.

The `SELECT` element is typically rendered as a pull down or pop-up list. For example:

```
<SELECT NAME="flavor">
<OPTION>Vanilla
<OPTION>Strawberry
<OPTION>Rum and Raisin
<OPTION>Peach and Orange
</SELECT>
```

If no option is initially marked as selected, then the first item listed is selected.


Strawberry
Vanilla
Strawberry
Rum and Raisin
Peach and Orange

The following are al allowed within the `<SELECT>` element

- `ERROR`
- `MULTIPLE`
- `NAME`
- `SIZE`

All are detailed on this page.

Other `<FORM>` elements :

<FORM> Element
<INPUT> Element
<OPTION> Element
<TEXTAREA> Element

Forms - <TEXTAREA> ... </TEXTAREA>

Attributes See Also Quick Reference

The Textarea element lets users enter more than one line of text. For example:

```
<TEXTAREA NAME="address" ROWS=64 COLS=6>
HaL Computer Systems
1315 Dell Avenue
Campbell, California 95008
</TEXTAREA>
```

The text up to the end element (`</TEXTAREA>`) is used to initialize the field's value. This end element is always required even if the field is initially blank. When submitting a form, lines in a `TEXTAREA` should be terminated using CR/LF.

In a typical rendering, the **`ROWS`** and **`COLS`** attributes determine the visible dimension of the field in characters. The field is rendered in a fixed-width font. HTML user agents should allow text to extend beyond these limits by scrolling as needed.

NOTE: In the initial design for forms, multi-line text fields were supported by the Input element with `TYPE=TEXT`. Unfortunately, this causes problems for fields with long text values. SGML's default (Reference Quantity Set) limits the length of attribute literals to only 240 characters. The HTML 2.0 SGML declaration increases the limit to 1024 characters.

The following are all allowed within the `<TEXTAREA>` element

- `NAME`
- `ROWS`
- `COLS`

All are detailed on this page.

Other `<FORM>` elements :

<FORM> Element
<INPUT> Element
<OPTION> Element
<SELECT> Element

Character Data

Quick Reference

The characters between HTML elements represent text. A HTML document (including elements and text) is encoded using the coded character set specified by the "charset" parameter of the "text/html" media type. For levels defined in this specification, the "charset" parameter is restricted to "US-ASCII" or "ISO-8859-1". ISO-8859-1 encodes a set of characters known as Latin Alphabet No. 1, or simply Latin-1. Latin-1 includes characters from most Western European languages, as well as a number of control characters. Latin-1 also includes a non-breaking space, a soft hyphen indicator, 93 graphical characters, 8 unassigned characters, and 25 control characters.

Because non-breaking space and soft hyphen indicator are not recognized and interpreted by all HTML user agents, their use is discouraged.

There are 58 character positions occupied by control characters. See Control Characters for details on the interpretation of control characters.

Because certain special characters are subject to interpretation and special processing, information providers and HTML user agent implementors should follow the guidelines in the Special Characters section.

In addition, HTML provides character entity references and numerical character references to facilitate the entry and interpretation of characters by name and by numerical position.

Because certain characters will be interpreted as markup, they must be represented by entity references as described in character and/or numerical references.

Special Characters

Quick Reference

Certain characters have special meaning in HTML documents. There are two printing characters which may be interpreted by an HTML application to have an effect of the format of the text:

Space

- Interpreted as a word space (place where a line can be broken) in all contexts except the Preformatted Text element.
- Interpreted as a nonbreaking space within the Preformatted Text element.

Hyphen

- Interpreted as a hyphen glyph in all contexts
- Interpreted as a potential word space by hyphenation engine

The following entity names are used in HTML, always prefixed by ampersand (&) and followed by a semicolon. 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 following table lists each of the supported characters specified in the Numeric and Special Graphic entity set, along with its name, syntax for use, and description. This list is derived from ISO Standard 8879:1986//ENTITIES Numeric and Special Graphic//EN however HTML does not provide support for the entire entity set. Only the entities listed below are supported.

Glyph	Name	Syntax	Description
<	lt	&lt	Less than sign
>	gt	&gt	Greater than sign
&	amp	&amp	Ampersand
"	quot	&quot	Double quote sign

Control Characters

Quick Reference

Control characters are non-printable characters that are typically used for communication and device control, as format effectors, and as information separators.

In SGML applications, the use of control characters is limited in order to maximize the chance of successful interchange over heterogenous networks and operating systems. In HTML, only three control characters are used: Horizontal Tab (HT, encoded as 9 decimal in US-ASCII and ISO-8859-1), Carriage Return, and Line Feed.

Horizontal Tab is interpreted as a word space in all contexts except preformatted text. Within preformatted text, the tab should be interpreted to shift the horizontal column position to the next position which is a multiple of 8 on the same line; that is, col := (col+8) mod 8.

Carriage Return and Line Feed are conventionally used to represent end of line. For Internet Media Types defined as "text/*", the sequence CR LF is used to represent an end of line. In practice, text/html documents are frequently represented and transmitted using an end of line convention that depends on the conventions of the source of the document; frequently, that representation consists of CR only, LF only, or CR LF combination. In HTML, end of line in any of its variations is interpreted as a word space in all contexts except preformatted text. Within preformatted text, HTML interpreting agents should expect to treat any of the three common representations of end-of-line as starting a new line.

Numeric Character References

See Also Quick Reference

In addition to any mechanism by which characters may be represented by the encoding of the HTML document, it is possible to explicitly reference the printing characters of the ISO-8859-1 character encoding using a numeric character reference.

Two reasons for using a numeric character reference:

- the keyboard does not provide a key for the character, such as on U.S. keyboards which do not provide European characters
- the character may be interpreted as SGML coding, such as the ampersand (&), double quotes ("), the lesser (<) and greater (>) characters

Numeric character references are represented in an HTML document as SGML entities whose name is number sign (#) followed by a numeral from 32-126 and 161-255. The HTML DTD includes a numeric character for each of the printing characters of the ISO-8859-1 encoding, so that one may reference them by number if it is inconvenient to enter them directly:

the ampersand (&), double quotes ("), lesser (<) and greater (>) characters

The following entity names are used in HTML, always prefixed by ampersand (&) and followed by a semicolon. This list, sorted numerically, is derived from ISO-8859-1 8-bit single-byte coded graphic character set:

Reference	Description
&#00- &#08	Unused
&#09	Horizontal tab
&#10	Line feed
&#11 - &#31	Unused
&#32	Space
&#33	Exclamation mark
&#34	Quotation mark
&#35	Number sign
&#36	Dollar sign
&#37	Percent sign
&#38	Ampersand
&#39	Apostrophe
&#40	Left parenthesis
&#41	Right parenthesis
&#42	Asterisk
&#43	Plus sign
&#44	Comma
&#45	Hyphen
&#46	Period (fullstop)
&#47	Solidus (slash)

&#48- &#57	Digits 0-9
&#58	Colon
&#59	Semi-colon
&#60	Less than
&#61	Equals aign
&#62	Greater than
&#63	Question mark
&#64	Commercial at
&#91	Left square bracket
&#92	Reverse solidus (backslash)
&#93	Right square bracket
&#94	Caret
&#95	Horizontal bar
&#96	Acute accent
&#97- &#122	Letters a-z
&#123	Left curly brace
&#124	Vertical bar
&#125	Right curly brace
&#126	Tilde
&#127 -	Unused
&#161	Inverted exclamation
&#162	Cent sign
&#163	Pound sterling
&#164	General currency sign
&#165	Yen sign
&#166	Broken vertical bar
&#167	Section sign
&#168	Umlaut (dieresis)
&#169	Copyright
&#170	Feminine ordinal
&#171	Left angle quote, guillemotleft
&#172	Not sign
&#173	Soft hyphen
&#174	Registered trademark
&#175	Macron accent
&#176	Degree sign
&#177	Plus or minus
&#178	Superscript two
&#179	Superscript three
&#180	Acute accent
&#181	Micro sign
&#182	Paragraph sign
&#183	Middle dot
&#184	Cedilla
&#185	Superscript one
&#186	Masculine ordinal
&#187	Right angle quote, guillemotright

&#188	Fraction one-fourth
&#189	Fraction one-half
&#190	Fraction three-fourths
&#191	Inverted question mark
&#192	Capital A, acute accent
&#193	Capital A, grave accent
&#194	Capital A, circumflex accent
&#195	Capital A, tilde
&#196	Capital A, ring
&#197	Capital A, dieresis or umlaut mark
&#198	Capital AE dipthong (ligature)
&#199	Capital C, cedilla
&#200	Capital E, acute accent
&#201	Capital E, grave accent
&#202	Capital E, circumflex accent
&#203	Capital E, dieresis or umlaut mark
&#204	Capital I, acute accent
&#205	Capital I, grave accent
&#206	Capital I, circumflex accent
&#207	Capital I, dieresis or umlaut mark
&#208	Capital Eth, Icelandic
&#209	Capital N, tilde
&#210	Capital O, acute accent
&#211	Capital O, grave accent
&#212	Capital O, circumflex accent
&#213	Capital O, tilde
&#214	Capital O, dieresis or umlaut mark
&#215	Multiply sign
&#216	Capital O, slash
&#217	Capital U, acute accent
&#218	Capital U, grave accent
&#219	Capital U, circumflex accent
&#220	Capital U, dieresis or umlaut mark
&#221	Capital Y, acute accent
&#222	Capital THORN, Icelandic
&#223	Small sharp s, German (sz ligature)
&#224	Small a, acute accent
&#225	Small a, grave accent
&#226	Small a, circumflex accent
&#227	Small a, tilde
&#228	Small a, dieresis or umlaut mark
&#229	Small a, ring
&#230	Small ae dipthong (ligature)
&#231	Small c, cedilla
&#232	Small e, acute accent
&#233	Small e, grave accent
&#234	Small e, circumflex accent

&#235	Small e, dieresis or umlaut mark
&#236	Small i, acute accent
&#237	Small i, grave accent
&#238	Small i, circumflex accent
&#239	Small i, dieresis or umlaut mark
&#240	Small eth, Icelandic
&#241	Small n, tilde
&#242	Small o, acute accent
&#243	Small o, grave accent
&#244	Small o, circumflex accent
&#245	Small o, tilde
&#246	Small o, dieresis or umlaut mark
&#247	Division sign
&#248	Small o, slash
&#249	Small u, acute accent
&#250	Small u, grave accent
&#251	Small u, circumflex accent
&#252	Small u, dieresis or umlaut mark
&#253	Small y, acute accent
&#254	Small thorn, Icelandic
&#255	Small y, dieresis or umlaut mark

Character Entity References

Character Entity References

See Also Quick Reference

Many of the Latin alphabet No. 1 set of printing characters may be represented within the text of an HTML document by a character entity.

Two reasons for using a character entity:

- the keyboard does not provide a key for the character, such as on U.S. keyboards which do not provide European characters
- the character may be interpreted as SGML coding, such as the ampersand (&), double quotes ("), the lesser (<) and greater (>) characters

A character entity is represented in an HTML document as an SGML entity whose name is defined in the HTML DTD. The HTML DTD includes a character entity for each of the SGML markup characters and for each of the printing characters in the upper half of Latin-1, so that one may reference them by name if it is inconvenient to enter them directly:

the ampersand (&), double quotes ("), lesser (<) and greater (>) characters

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

NOTE: To ensure that a string of characters is not interpreted as markup, represent all occurrences of <, >, and & by character or entity references.

NOTE: There are SGML features, CDATA and RCDATA, to allow most <, >, and & characters to be entered without the use of entity or character references. Because these features tend to be used and implemented inconsistently, and because they require 8-bit characters to represent non-ASCII characters, they are not used in this version of the HTML DTD. An earlier HTML specification included an Example element (<XMP>) whose syntax is not expressible in SGML. No markup was recognized inside of the Example element except the `</XMP>` end element. While HTML user agents are encouraged to support this idiom, its use is deprecated.

The following entity names are used in HTML, always prefixed by ampersand (&) and followed by a semicolon. The following table lists each of the characters specified in the Added Latin 1 entity set, along with its name, syntax for use, and description. This list is derived from ISO Standard 8879:1986//ENTITIES Added Latin 1//EN. HTML supports the entire entity set.

Name	Syntax	Description
Aacute	&Aacute	Capital A, acute accent
Agrave	&Agrave	Capital A, grave accent
Acirc	&Acirc	Capital A, circumflex accent
Atilde	&Atilde	Capital A, tilde
Aring	&Aring	Capital A, ring
Auml	&Auml	Capital A, dieresis or umlaut mark
AElig	&AElig	Capital AE dipthong (ligature)
Ccedil	&Ccedil	Capital C, cedilla

Eacute	&Eacute	Capital E, acute accent
Egrave	&Egrave	Capital E, grave accent
Ecirc	&Ecirc	Capital E, circumflex accent
Euml	&Euml	Capital E, dieresis or umlaut mark
Iacute	&Iacute	Capital I, acute accent
Igrave	&Igrave	Capital I, grave accent
Icirc	&Icirc	Capital I, circumflex accent
Iuml	&Iuml	Capital I, dieresis or umlaut mark
ETH	&ETH	Capital Eth, Icelandic
Ntilde	&Ntilde	Capital N, tilde
Oacute	&Oacute	Capital O, acute accent
Ograve	&Ograve	Capital O, grave accent
Ocirc	&Ocirc	Capital O, circumflex accent
Otilde	&Otilde	Capital O, tilde
Ouml	&Ouml	Capital O, dieresis or umlaut mark
Oslash	&Oslash	Capital O, slash
Uacute	&Uacute	Capital U, acute accent
Ugrave	&Ugrave	Capital U, grave accent
Ucirc	&Ucirc	Capital U, circumflex accent
Uuml	&Uuml	Capital U, dieresis or umlaut mark;
Yacute	&Yacute	Capital Y, acute accent
THORN	&THORN	Capital THORN, Icelandic
Szlig	&szlig	Small sharp s, German (sz ligature)
aacute	&aacute	Small a, acute accent
agrave	&agrave	Small a, grave accent
acirc	&acirc	Small a, circumflex accent
atilde	&atilde	Small a, tilde
aring	&aring	Small a, ring
auml	&auml	Small a, dieresis or umlaut mark
aelig	&aelig	Small ae dipthong (ligature
ccedil	&ccedil	Small c, cedilla
eacute	&eacute	Small e, acute accent
egrave	&egrave	Small e, grave accent
ecirc	&ecirc	Small e, circumflex accent
euml	&euml	Small e, dieresis or umlaut mark
iacute	&iacute	Small i, acute accent
igrave	&igrave	Small i, grave accent
icirc	&icirc	Small i, circumflex accent
iuml	&iuml	Small i, dieresis or umlaut mark
eth	&eth	Small eth, Icelandic
ntilde	&ntilde	Small n, tilde
oacute	&oacute	Small o, acute accent
ograve	&ograve	Small o, grave accent
ocirc	&ocirc	Small o, circumflex accent
otilde	&otilde	Small o, tilde

ouml	&ouml	Small o, dieresis or umlaut mark
oslash	&oslash	Small o, slash
uacute	&uacute	Small u, acute accent
ugrave	&ugrave	Small u, grave accent
ucirc	&ucirc	Small u, circumflex accent
uuml	&uuml	Small u, dieresis or umlaut mark
yacute	&yacute	Small y, acute accent
thorn	&thorn	Small thorn, Icelandic
yuml	&yuml	Small y, dieresis or umlaut mark
reg	&reg	Registered TradeMark
copy	&copy	Copyright
trade	&trade	TradeMark

NOTE : The last three character entities are only supported in recent versions of Mosaic and Netscape . They may not appear as planned in early versions of these, or different browsers.

Numerical Character References

Obsolete and Proposed features

Elements Quick Reference

Obsolete Features

This section describes elements that are no longer part of HTML. Client implementors should implement these obsolete elements for compatibility with previous versions of the HTML specification.

Comment

The Comment element is used to delimit unneeded text and comments. The Comment element has been introduced in some HTML applications but should be replaced by the SGML comment feature in new HTML user agents

Highlighted Phrase

The Highlighted Phrase element (`<HP>`) should be ignored if not implemented. This element has been replaced by more meaningful elements. (See here)

Example of use:

```
<HP1>first highlighted phrase</HP1>non highlighted text<HP2>second
highlighted phrase</HP2> etc.
```

Plain Text

The Plain Text element is used to terminate the HTML entity and to indicate that what follows is not SGML which does not require parsing. Instead, an old HTTP convention specified that what followed was an ASCII (MIME "text/plain") body. Its presence is an optimization. There is no closing element.

Example of use:

```
<PLAINTEXT>
0001 This is line one of a long listing
0002 file from <ANY@HOST.INC.COM> which is sent
```

Example and Listing

<XMP> ... </XMP> and <LISTING> ... </LISTING>

The Example element and Listing elements have been replaced by the Preformatted Text element. 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 elements is typically rendered in a monospaced font so that any formatting done by character spacing on successive lines will be maintained.

Between the opening and closing elements:

- The text may contain any ISO Latin-1 printable characters, expect for the end element opener. The Example and Listing elements have historically used specifications which do not conform to SGML. Specifically, the text may contain ISO Latin printable characters, including the element opener, as long it they does not contain the closing element in full.

- SGML does not support this form. HTML user agents may vary on how they interpret other elements within Example and Listing elements.

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

- The horizontal tab 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.

The Listing element is rendered so that at least 132 characters fit on a line. The Example element is rendered to that at least 80 characters fit on a line but is otherwise identical to the Listing element.

Proposed Features

This section describes proposed HTML elements and entities that are not currently supported under HTML Levels 0, 1, or 2, but may be supported in the future.

Defining Instance

```
<DFN> ... </DFN>
```

The Defining Instance element indicates the defining instance of a term. The typical rendering is bold or bold italic. This element is not widely supported.

Special Characters

To indicate special characters, HTML uses entity or numeric representations. Additional character presentations are proposed:

Character	Representation
Non-breaking space	
Soft-hyphen	&shy
Registered	&reg
Copyright	&copy

Strike

```
<STRIKE> ... </STRIKE>
```

The Strike element is proposed to indicate strikethrough, a font style in which a horizontal line appears through characters. This element is not widely supported.

Underline

```
<U> ... </U>
```

The Underline element is proposed to indicate that the text should be rendered as underlined. This proposed element is not supported by all HTML user agents.

Example of use:

```
The text <U>shown here</U> is rendered in the document as underlined.
```

The following elements are covered on this page :

- <!-- ...--> : Comments
- <HP> : Highlighted phrases
- <PLAINTEXT> : Text to be ignored by the parser
- <XMP> : Fixed width formatting
- <LISTING> : Fixed width formatting
- <DFN> : Defining Instance
- <STRIKE> : Strike through text
- <U> : Underlined text.

<!-- Comments -->

Quick Reference

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

For example:

```
<HEAD>
<TITLE Guide: Recommended Usage</TITLE>
<!-- Id: Text.html,v 1.6 1994/04/25 17:33:48 connolly Exp -->
</HEAD>
```

NOTE: Some historical HTML user agents incorrectly consider a > sign to terminate a comment.

Microsoft Extensions to HTML

Microsoft were slow to get on the Web, but they are taking strong steps to catch up with the companies that beat them to it, especially Netscape. Microsoft have their own web browser called, Internet Explorer and is available from,

http://www.microsoft.com

which supports a number of new tags Microsoft have created. Be careful with using their tags - like Netscape proprietary tags, these are unlikely to work on all browsers. We've listed for you all the additions and extensions to the HTML 2 specification that Microsoft have built into Internet Explorer.

See also

Netscape Extensions to HTML
Additions

Microsoft Addition: BGSOUND

Quick Reference

This tag lets you play background sounds or soundtracks while the reader is looking at your page.

SRC=URL
Specifies the Internet address of the sound file to play.

LOOP=n
LOOP=INFINITE
Specifies how many times a sound file will be played. If **LOOP=INFINITE** or **LOOP=-1**, the sound file will keep playing indefinitely.

For example:

```
<BGSOUND SRC=music.wav LOOP=5>
```

Microsoft Addition: MARQUEE

Quick Reference

The MARQUEE tag lets you create a scrolling text area. This is a useful space for advertizing or other information. There are a number of attributes that let you control the use of Marquees:

ALIGN
Specifies the location of text around the marquee. Can be either TOP, MIDDLE, or BOTTOM.

BEHAVIOR
Determines how the text will move within the marquee.

SCROLL	- the text will move in from one side, and disappear off the other
SLIDE	- the text will move in from one side, then stop when it touches the other
ALTERNATE	- the text will bounce back and forth within the marquee

BGCOLOR
Specifies the background color for the marquee. Instead of using the hexadecimal RGB triplet (e.g. #FFFFFF), you can use a Color Name.

DIRECTION
The direction that the text should scroll towards. It can be either LEFT or RIGHT

HEIGHT
The height of the Marquee, or as a percentage of screen height

HSPACE
The distance between the left and right margins of the marquee, and the surrounding text (in pixels)

LOOP=n
LOOP=INFINITE
Specifies how many times text will scroll across the marquee. If **LOOP=INFINITE** or **LOOP=-1**, the text will repeat indefinitely.

SCROLLAMOUNT
Specifies the number of pixels between each successive draw of the marquee text.

SCROLLDELAY
Specifies the number of milliseconds between each successive draw of the marquee text.

VSPACE
The distance between the top and bottom margins of the marquee, and the surrounding text (in pixels)

Netscape Extensions to HTML

Quick Reference

At the time of writing, the Netscape Navigator is the most widely-used browser on the Internet. However, they are facing increasingly tough competition, particularly from the large online services like CompuServe, America Online, Prodigy, and the Microsoft Network, who provide their users with custom browsing software. It used to be a rule of thumb in web design that 80% of people viewing your pages would do so with Netscape; however, this is no longer such a sure thing.

Netscape introduced many new HTML elements that aren´t defined in the *official* HTML specification and supported various extensions to elements that are defined in the specification as defined by the HTML working group.

Here, all extensions and additions to the HTML 2.0 specification are considered, as well as some of the extensions that are supported in the more recent versions of Netscape (1.1 beta 1 and above), that the Netscape authors are hoping will be implemented into HTML level 3.0 (or HTML+).

See also

Microsoft Extensions to HTML

Netscape Specific Extensions to existing Elements

Quick Reference

The following HTML 2.0 elements have all received extra attributes which are supported by Netscape , versions 1.0 and above.

NOTE : The new attributes that are supported by Netscape browsers may ***only*** be supported by Netscape browsers. Take care with document styling. If the user is not using Netscape , their view of the document may not be what you intended.

Netscape extensions to <ISINDEX>

See Also Quick Reference

To the `<ISINDEX>` element Netscape authors have added the `PROMPT` attribute. `<ISINDEX>` indicates that a document is a searchable index.

`PROMPT` has been added so that text, chosen by the author, can be placed before the text input field of the index. This allows any author chosen message to replace the default text of :

This is a searchable index. Enter search keywords

<ISINDEX> Element

Netscape extensions to <HR>

New Attributes See Also Quick Reference

The `<HR>` element specifies that a horizontal rule of some sort (The default being a shaded engraved line) be drawn across the page. To this element Netscape have added 4 new attributes which allow the document author to describe how the horizontal rule should look.

`<HR SIZE=number>`

The `SIZE` attributes lets the author give an indication of how thick they wish the horizontal rule to be.

`<HR WIDTH=number|percent>`

The default horizontal rule is always as wide as the page. With the `WIDTH` attribute, the author can specify an exact width in pixels, or a relative width measured in percent of document width.

`<HR ALIGN=left|right|center>`

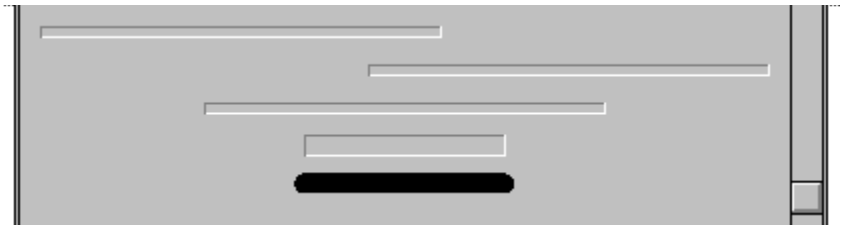
Now that horizontal rules do not have to be the width of the page it is necessary to allow the author to specify whether they should be pushed up against the left margin, the right margin, or centered in the page.

`<HR NOSHADE>`

Finally, for those times when a solid bar is required, the `NOSHADE` attribute lets the author specify that the horizontal rule should not be shaded at all.

The following <HR> render as shown

```
<HR SIZE=5 WIDTH=200 ALIGN=Left>
<HR SIZE=5 WIDTH=200 ALIGN=Right>
<HR SIZE=5 WIDTH=200 ALIGN=Center>
<HR SIZE=10 WIDTH=100 ALIGN=Center>
<HR SIZE=10 WIDTH=100 ALIGN=Center NOSHADE>
```



NOTE : The page side bars are shown to emphasise the line positioning on the page.

Netscape supports these additional attributes to the `<HR>` element

- `SIZE`
- `WIDTH`
- `ALIGN`
- `NOSHADE`

All are detailed on this page

<HR> Element

Netscape extensions to <UL>

See Also Quick Reference

The basic bulleted list has a default progression of bullet types that changes as you move through indented levels. From a solid disc, to a circle to a square. Netscape authors have added a **TYPE** attribute to the `<UL>` element so that no matter what the indent level the bullet type can be specified thus :

```
TYPE=disc
TYPE=circle
TYPE=square
```

NOTE : The types disc and circle appear the same when rendered.

`TYPE=disc/circle:`

- First list item
- Second list item
- Third list item

`TYPE=square:`

- First list item
- Second list item
- Third list item

<UL> Element

Netscape extensions to <OL>

See Also Quick Reference

The average ordered list counts 1, 2, 3, ... etc. Netscape authors have added the **TYPE** attribute to this element to allow authors to specify whether the list items should be marked with:

`(TYPE=A)` - capital letters. e.g. A, B, C ...
`(TYPE=a)` - small letters. e.g. a, b, c ...
`(TYPE=I)` - large roman numerals. e.g. I, II, III ...
`(TYPE=i)` - small roman numerals. e.g. i, ii, iii ...
`(TYPE=1)` - or the default numbers. e.g. 1, 2, 3 ...

For lists that wish to start at values other than 1 the new attribute **START** is available.

`START` is always specified in the default numbers, and will be converted based on `TYPE` before display. Thus `START=5` would display either an 'E', 'e', 'V', 'v', or '5' based on the `TYPE` attribute.

The following Ordered List, was rendered using `TYPE=a START=3`

c. Click the Web button to open the Open the URL window.
d. Enter the URL number in the text field of the Open URL window.
   The Web document you specified is displayed.
e. Click highlighted text to move from one link to another.

<OL> Element

Netscape extensions to <LI>

See Also Quick Reference

To give even more flexibility to lists, Netscape authors have added the **TYPE** attribute to the <LI> element as well. It takes the same values as either <UL> or <OL> depending on the type of list you are in, and it changes the list type for that item, and all subsequent items. For ordered lists we have also added the **VALUE** element so you can change the count, for that list item and all subsequent.

<OL> Ordered List Element
<UL> Unordered List Element

Netscape extensions to <IMG...>

New Attributes See Also Quick Reference

The `<IMG ...>` attribute is probably the most extended element.

```
<IMG ALIGN= left|right|top|texttop|middle|
absmiddle|baseline|bottom|absbottom>
```

The additions to your `ALIGN` options needs a lot of explanation. First, the values "left" and "right". Images with those alignments are an entirely new *floating* image type.

`ALIGN=left` image will float the image down and over to the left margin (into the next available space there), and subsequent text will wrap around the right hand side of that image.

`ALIGN=right` will align the image aligns with the right margin, and the text wraps around the left.

`ALIGN=top` aligns itself with the top of the tallest item in the line.

`ALIGN=texttop` aligns itself with the top of the tallest text in the line (this is usually but not always the same as `ALIGN=top`).

`ALIGN=middle` aligns the baseline of the current line with the middle of the image.

`ALIGN=absmiddle` aligns the middle of the current line with the middle of the image.

`ALIGN=baseline` aligns the bottom of the image with the baseline of the current line.

`ALIGN=bottom` aligns the bottom of the image with the baseline of the current line.

`ALIGN=absbottom` aligns the bottom of the image with the bottom of the current line.

```
<IMG WIDTH=value HEIGHT=value>
```

The `WIDTH` and `HEIGHT` attributes were added to `<IMG ...>` mainly to speed up display of the document. If the author specifies these, the viewer of their document will not have to wait for the image to be loaded over the network and its size calculated.

```
<IMG BORDER=value>
```

This lets the document author control the thickness of the border around an image displayed.

Warning: setting `BORDER=0` on images that are also part of anchors may confuse your users as they are used to a colored border indicating an image is an anchor.

```
<IMG VSPACE=value HSPACE=value>
```

For the *floating* images it is likely that the author does not want them pressing up against the text wrapped around the image. `VSPACE` controls the vertical space above and below the image, while `HSPACE` controls the horizontal space to the left and right of the image.

LOWSRC

Using the `LOWSRC` attribute, it is possible to use two images in the same space. The syntax is :

```
<IMG SRC="highres.gif" LOWSRC="lowres.jpg">
```

Browsers that do not recognize the LOWSRC attribute cleanly ignore it and simply load the image called "highres.gif".

The Netscape Navigator, on the other hand, will load the image called "lowres.jpg" on its first layout pass through the document. Then, when the document and all of its images are fully loaded, the Netscape Navigator will do a second pass through and load the image called "highres.gif" in place. This means that you can have a very low-resolution version of an image loaded initially; if the user stays on the page after the initial layout phase, a higher-resolution (and presumably bigger) version of the same image can "fade in" and replace it.

Both GIF (both normal and interlaced) and JPEG images can be freely interchanged using this method. You can also specify width and/or height values in the `IMG` element, and both the high-res and low-res versions of the image will be appropriately scaled to match.

If the images are of different sizes and a fixed height and width are not specified in the `IMG` element, the second image (the image specified by the `SRC` attribute) will be scaled to the dimensions of the first (`LOWSRC`) image. This is because by the time the Netscape Navigator knows the dimensions of the second image, the first image has already been displayed in the document at its normal dimensions.

This picture is aligned to the left of the page. All of this text will fall neatly to the side of the image.

This picture is aligned to the right of the page. All of this text will fall neatly to the side of the image. This image also has a BORDER of 2

Netscape supports these additional attributes to the `<IMG ...>` element

- New alignments
- `WIDTH`
- `HEIGHT`
- `BORDER`
- `VSPACE`
- `HSPACE`
- `LOWSRC`

<IMG ...> Element

Netscape extension to

See Also Quick Reference

With the addition of *floating* images, it was necessary to expand the `<BR>` element. Normal `<BR>` still just inserts a line break. A `CLEAR` attribute has been added to `<BR>`, so :

`CLEAR``=left` will break the line, and move vertically down until you have a clear left margin (no *floating* images).

`CLEAR=right` does the same for the right margin.

`CLEAR=all` moves down until both margins are clear of images.

NOTE : The screenshots on the Netscape extensions to <IMG> pages use `<BR CLEAR=left>` and `<BR CLEAR=right>` respectively.

 Line Break Element

Netscape Specific Additions to HTML 2.0

Quick Reference

The following elements are all new . They are not part of the *Official* HTML specification for level 2.0, but are supported by Netscape , versions 1.1 and above and may appear in the finalised HTML 3.0 specification.

NOTE : The new attributes that are supported by Netscape browsers may ***only*** be supported by Netscape browsers. Take care with document styling. If the user is not using Netscape , their view of the document may not be what you intended.

Netscape addition - <NOBR>

See Also Quick Reference

The <NOBR> element stands for **NO BR**eak. This means all the text between the start and end of the <NOBR> elements cannot have line breaks inserted. While <NOBR> is essential for those character sequences that don't want to be broken, please be careful; long text strings inside of <NOBR> elements can look rather odd. Especially if during viewing, the user adjust the page size by altering the window size.

Netscape addition - <WBR>

See Also Quick Reference

The <WBR> element stands for **W**ord **BR**eak. This is for the very rare case when a <NOBR> section requires an exact break. Also, it can be used any time the Netscape Navigator can be helped by telling it where a word is allowed to be broken. The <WBR> element does not force a line break (
 does that) it simply lets the Netscape Navigator know where a line break is allowed to be inserted if needed.

Netscape addition - <FONT SIZE ...>

See Also Quick Reference

Netscape 1.0 and above supports different sized fonts within HTML documents. This should be distinguished from Headings.

The new element is `<FONT SIZE=value>`. Valid values range from 1-7. The default FONT size is 3. The value given to size can optionally have a '+' or '-' character in front of it to specify that it is relative the the document baseFONT. The default baseFONT is 3, and can be changed with the <BASEFONT SIZE ...> element.

e.g.

`<FONT SIZE=4>` changes the font size to 4 `</FONT>`

`<FONT SIZE=+2>` changes the font size to `<BASEFONT SIZE ...>` + 2 `</FONT>`

Other Netscape additions to HTML 2.0
Setting the <BASEFONT> size

Netscape addition - <BASEFONT SIZE ...>

See Also Quick Reference

This changes the size of the BASEFONT that all relative <FONT SIZE ...> changes are based on. It defaults to 3, and has a valid range of 1-7.

e.g. `<BASEFONT SIZE=5>`

Other Netscape additions to HTML 2.0
Changing the <FONT SIZE>

Netscape addition - <CENTER>

See Also Quick Reference

All lines of text between the begin and end of the `<CENTER>` element are centered between the current left and right margins. A new element has been introduced rather than using the proposed <P ALIGN= CENTER> because using `<P ALIGN= CENTER >` breaks many existing browsers when the `<P>` element is used as a container. The `<P ALIGN= CENTER >` element is also less general and does not support all cases where centering may be desired.

```
<CENTER>All this text would be centered in the page</CENTER>
```

Other Netscape additions to HTML 2.0
HTML 3.0 - Paragraph alignment

Netscape addition - <BLINK>

See Also Quick Reference

Surrounding any text with this element will cause the selected text to *blink* on the viewing page. This can serve to add extra emphasis to selected text.

```
<BLINK>This text would blink on the page</BLINK>
```

Other Netscape additions to HTML 2.0

Netscape Addition - <EMBED>

Quick Reference

NOTE : This tag is **only** recognised by the Windows version of Netscape Navigator, versions 1.1 and above.

The `<EMBED>` element allows you to put documents directly into an HTML page.

The syntax is:

```
<EMBED SRC="images/embed.bmp">
```

The `<EMBED>` element will allow you to embed documents of any type. Your user only needs to have an application which can view the data installed correctly on their machine.

If a width and height are specified, the embedded object is scaled to fit the available space. For example this is the same bitmap as above, scaled:

```
<EMBED SRC="images/embed.bmp" WIDTH=250 HEIGHT=50>
```

Embedded objects can be activated by double clicking them in the Netscape window. The application that supports use of the embedded object will be launched, with the object present.

NOTE : Using the `<EMBED>` element, you should be sure that the user will have a suitable application available that is OLE compliant. Otherwise, the HTML document will not be displayed as hoped. Essentially this element produces the same results as embedding objects in Word for Windows - the object is displayed, and can be edited in a suitable application by double clicking on the object.

HTML 3.0

Quick Reference

When the HTML specification level 3.0 is finally decided upon, it is likely that one of the most notable improvement over level 2.0 will be the inclusion of standardised elements allowing the user to create tables. While this specification has not yet been finalised, it hasn´t stopped both Netscape and Mosaic authors inplementing table support for their WWW browser. While such support is present, it should be used with care. As fast as authors are implementing support, the specification is changing and when designing WWW pages, you cannot envisage what version of what browser the user is using.

Also, adding alignment attributes to both Headings and Paragraph elements is proposed in HTML 3.0. At present, Netscape supports the <CENTER> element, as opposed to the `<P ALIGN=...>` element, currently supported by Mosaic 2.0b4 and proposed for the *official* HTML 3.0 specification.

> ***NOTE :*** Elements detailed here may only be supported by the *most recent* HTML browsers. It cannot be guaranteed which version of any browser users will be using. This should be foremost in the minds of HTML authors when constructing HTML documents.

HTML 3.0 - Heading alignment

See Also Quick Reference

Included in the proposed HTML level 3.0 specification is the ability to align Headings

Basically, **ALIGN**=left|center|right attributes have been added to the <H1> to <H6> elements. e.g :

```
<H1 ALIGN=center>Hello, this is a heading</H1>
```

would align a heading of style 1 in the centre of the page.

NOTE : This element is currently only supported by Mosaic 2.0 beta 4.

Hello, this is a centred heading

Hello, this is a right aligned heading

Hello, this is a left aligned heading

<H> Heading elements

HTML 3.0 - Paragraph alignment

See Also Quick Reference

Included in the proposed HTML level 3.0 specification is the ability to align paragraphs Basically, **ALIGN**=left|center|right attributes have been added to the <P> element. e.g :

```
<P ALIGN=LEFT> ... </P>
```

All text with in the paragraph will be aligned to the left side of the page layout. This setting is equal to the default <P> element.

```
<P ALIGN=CENTER> ... </P>
```

All text within the paragraph will be aligned to the center of the page.

```
<P ALIGN=RIGHT> ... </P>
```

All text will be aligned to the right side of the page.

NOTE: To account for the commonly used yet non-standard <CENTER> element, Mosaic (2.0b4) will change the default ALIGN=LEFT attribute of all paragraph and header elements to ALIGN=CENTER until a </CENTER> element is read. Mosaic will also allow internally defined alignment attributes to take precedence over a wrapping CENTER element. Mosaic authors would like to encourage all HTML authors to conform to the HTML 3.0 way of centering HTML and no longer use the non-standard <CENTER> element.

All of this paragraph
is left
aligned

All of this
paragraph
is centered

All of this
paragraph
is right
aligned

<P> Paragraph element
<CENTER> element

Tables

Quick Reference

At present, the table HTML elements are :

<TABLE> ... </TABLE> - The Table delimiter.
<TR ...> ... </TR> - Used to specify number of rows in a table.
<TD ...> ... </TD> - Specifies table data cells.
<TH ...> ... </TH> - Table Header cell.
<CAPTION ...> ... </CAPTION> - Specifies the table Caption.

NOTE : Some of the attributes of these elements are at present not included in the sketchy specification for level 3.0 HTML and may only be supported by Netscape browsers (recent versions). Such attributes are marked with a *.
e.g. `*CELLSPACING=*`

If these details are too confusing, there is also a graphical guide to Tables.

Some considerations about Tables

Blank cells which contain no displayable elements are not given borders. If a bordered but empty cell is required, put either a blank line or a non-breaking space in the cell. (`<td> </td>` or `<td><br></td>`)

The proposed HTML 3.0 spec. allows you to abuse row and column spans to create tables whose cells must overlap. Until this specification is finalised,don't do this, it looks awful.

Eventuall, you may create a cell containing an image and it will possibly be rendered with the image off-centre. The reason for this could be due to the HTML being written like :

```
<TD>
      <IMG SRC="url">
</TD>
```

That extra whitespace inside the cell and around the image gets collapsed into single space characters. It is these spaces (whose baselines by default align with the bottom of the image) that make the cell look lopsided. Instead, use :1

```
<TD><IMG SRC="url">
</TD>
```

Also, please consider the user. If theyre not using a recent browser that supports table elements, then page design will be completely lost. At present, use tables with caution

<TABLE> ... </TABLE>

Attributes See Also Quick Reference

This is the main wrapper for all the other table elements, and other table elements will be ignored if they aren't wrapped inside of a <TABLE> ... <TABLE> element. By default tables have no borders, borders will be added if the BORDER attribute is specified. At the time of writing, the <TABLE> element has an implied linebreak both before and after it. This is expected to change, allowing as much control over placement of tables as is currently available for the placement of images. Aligning them to various positions in a line of text, as well as shifting them to the left or right margins and wrapping text around them.

The <TABLE> element has the following attributes :

BORDER

This attribute appears in the <TABLE> element. If present, borders are drawn around all table cells. If absent, there are no borders, but by default space is left for borders, so the same table with and without the BORDER attribute will have the same width.

BORDER=<value>

By allowing the BORDER attribute to take a value, the document author gains two things. First they gain the ability to emphasize some tables with respect to others, a table with a border of four containing a sub-table with a border of one looks much nicer than if they both share the same default border width. Second, by explicitly setting border to zero they regain that space originally reserved for borders between cells, allowing particularly compact tables.

***CELLSPACING**=<value>*

This is a new attribute for the <TABLE> element. By default Netscape uses a cell spacing of two. For those fussy about the look of their tables, this gives them a little more control. Like it sounds, cell spacing is the amount of space inserted between individual cells in a table.

***CELLPADDING**=<value>*

This is a new attribute for the <TABLE> element. By default Netscape uses a cell padding of one. Cell padding is the amount of space between the border of the cell and the contents of the cell. Setting a cell padding of zero on a table with borders might look bad because the edges of the text could touch the cell borders.

```
<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0>
```

gives the most compact table possible.

***WIDTH**=<value_or_percent>*

When this attribute appears in the <TABLE> element it is used to describe the desired width of this table, either as an absolute width in pixels, or a percentage of document width. Ordinarily complex heuristics are applied to tables and their cells to attempt to present a pleasing looking table. Setting the <WIDTH> attribute overrides those heuristics and instead effort is put into fitting the table into the desired width as specified. In some cases it might be impossible to fit all the table cells at the specified width, in which case Netscape will try and get as close as possible.

When this attribute appears on either the <TH> or <TD> element it is used to describe the desired width of the cell, either as an absolute width in pixels, or a percentage of table width. Ordinarily complex heuristics are applied to table cells to attempt to present a pleasing looking table.

Setting the <WIDTH> attribute overrides those heuristics for that cell and instead effort is put into fitting the cell into the desired width as specified. In some cases it might be impossible to fit all the table cells at the specified widths, in which case Netscape will try and get as close as possible.

The following are all allowed within the `<TABLE>` element

- BORDER
- CELLSPACING
- CELLPADDING
- WIDTH

<TR> Specifies number of rows in a table.
<TD> Data cell element.
<TH> Header element.
<CAPTION> Caption element.
Graphical examples of Tables

Table - <TR ...> ... </TR>

See Also Quick Reference

This stands for table row. The number of rows in a table is exactly specified by how many `<TR>` elements are contained within it, irregardless of cells that may attempt to use the <ROWSPAN> attribute to span into non-specified rows. `<TR>` can have both the <ALIGN> and <VALIGN> attributes, which if specified become the default alignments for all cells in this row.

<TABLE> - The Table element.
<TD> Table data cell element.
<TH> Table Header element.
<CAPTION> Caption element.
Graphical examples of Tables

Table - <TD ...> ... </TD>

See Also Quick Reference

This stands for table data, and specifies a standard table data cell. Table data cells must only appear within table rows. Each row need not have the same number of cells specified as short rows will be padded with blank cells on the right. A cell can contain any of the HTML elements normally present in the body of an HTML document. The default alignment of table data is `ALIGN=left` and `VALIGN=middle`. These alignments are overridden by any alignments specified in the containing `<TR>` element, and those alignments in turn are overridden by any ALIGN or VALIGN attributes explicitly specified on this cell. By default lines inside of table cells can be broken up to fit within the overall cell width. Specifying the `NOWRAP` attribute for a `<TD>` prevents linebreaking for that cell.

`<TD ...> ... </TD>` can also contain NOWRAP, COLSPAN and ROWSPAN attributes

<TABLE> - The Table element.
<TR> Specifies number of rows in a table.
<TH> Table Header element.
<CAPTION> Caption element.
Graphical examples of Tables

Table - <TH ...> ... </TH>

See Also Quick Reference

This stands for table header. Header cells are identical to data cells in all respects, with the exception that header cells are in a **bold** FONT, and have a default ALIGN=`center`.

`<TH ...> ... </TH>` can also contain VALIGN, NOWRAP, COLSPAN and ROWSPAN attributes

<TABLE> - The Table element.
<TR> Specifies number of rows in a table.
<TD> Table data cell element.
<CAPTION> Caption element.
Graphical examples of Tables

Table - <CAPTION ...> ... </CAPTION>

See Also Quick Reference

This represents the caption for a table. `<CAPTION>` elements should appear inside the `<TABLE>` but not inside table rows or cells. The caption accepts an alignment attribute that defaults to ALIGN`=top` but can be explicitly set to `ALIGN=bottom`. Like table cells, any document body HTML can appear in a caption. Captions are always horizontally centered with respect to the table, and the may have their lines broken to fit within the width of the table.

<TABLE> - The Table element.
<TR> Specifies number of rows in a table.
<TD> Table data cell element.
<TH> Table Header element.
Graphical examples of Tables

Table - ALIGN attribute

See Also Quick Reference

If appearing inside a <CAPTION> it controls whether the caption appears above or below the table, and can have the values **top** or **bottom**, defaulting to top.

If appearing inside a <TR>, <TH>, or <TD> it controls whether text inside the table cell(s) is aligned to the left side of the cell, the right side of the cell, or centered within the cell. Values are **left**, **center**, and **right**.

<TR> Table Row element
<TD> Table Data element
<TH> Table Header element
<CAPTION> Table caption element
Graphical examples of Tables

Table - VALIGN attribute

See Also Quick Reference

Appearing inside a <TR>, <TH>, or <TD> it controls whether text inside the table cell(s) is aligned to the top of the cell, the bottom of the cell, or vertically centered within the cell. It can also specify that all the cells in the row should be vertically aligned to the same baseline. Values are **top**, **middle**, **bottom** and **baseline**.

<TR> Table Row element
<TD> Table Data element
<TH> Table Header element
Graphical examples of Tables

Table - NOWRAP attribute

See Also Quick Reference

If this attribute appears in any table cell (<TH> or <TD>) it means the lines within this cell cannot be broken to fit the width of the cell. Be cautious in use of this attribute as it can result in excessively wide cells.

<TD> Table Data element
<TH> Table Header element
Graphical examples of Tables

Table - COLSPAN attribute

See Also Quick Reference

This attribute can appear in any table cell (<TH> or <TD>) and it specifies how many columns of the table this cell should span. The default `COLSPAN` for any cell is 1.

<TD> Table Data element
<TH> Table Header element
Graphical examples of Tables

Table - ROWSPAN attribute

See Also Quick Reference

This attribute can appear in any table cell (<TH> or <TD>) and it specifies how many rows of the table this cell should span. The default ROWSPAN for any cell is 1. A span that extends into rows that were never specified with a <TR> will be truncated.

<TR> Table Row element
<TD> Table Data element
<TH> Table Header element
Graphical examples of Tables

Graphical Examples of Tables

Quick Reference

NOTE : The screen-shots provided here are used as examples. They are © 1995, Netscape communications.

Basic Tables :

A Basic 3x2 table
Two demonstrations of ROWSPAN
Demonstration of COLSPAN
Demonstration of HEADERS, <TH ...>
Demonstration of COLSPAN plus <TH ...>
Demonstration of multiple headers plus COLPSPAN
Demonstration of side headers
Demonstration of side headers plus ROWSPAN
Sample table using all of the above
Clever uses of ROWSPAN/COLSPAN

Adjusting margins and borders :

A table without borders
A table with a BORDER of 10
CELLPADDING and CELLSPACING

Alignment, Captions and Sub-tables :

Demonstration of multiple lines in a table
ALIGN=left|right|center
VALIGN=top|bottom|middle
CAPTION=top|bottom
Nested tables

Table Widths and placing :

Different table widths
Centering a table
Table width and nested tables
HEIGHT

Basic 3x2 Table

Quick Reference

The following will render a Basic table having three columns and two rows.

```
<TABLE BORDER>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

A	B	C
D	E	F

Two demonstrations of ROWSPAN

Quick Reference

The following will render a table where item 2 spans two rows.

```
<TABLE BORDER>
      <TR>
            <TD>Item 1</TD>
            <TD ROWSPAN=2>Item 2</TD>
            <TD>Item 3</TD>
      </TR>
      <TR>
            <TD>Item 4</TD> <TD>Item 5</TD>
      </TR>
</TABLE>
```

The following will render a table where item 1 spans two rows.

```
<TABLE BORDER>
      <TR>
            <TD ROWSPAN=2>Item 1</TD>
            <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item 4</TD>
      </TR>
      <TR>
            <TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD>
      </TR>
</TABLE>
```

Item 1	Item 2	Item 3
Item 4		Item 5

Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7

Demonstration of COLSPAN

Quick Reference

The following will render a table where item 2 spans two columns

```
<TABLE BORDER>
      <TR>
            <TD>Item 1</TD>
            <TD COLSPAN=2>Item 2</TD>
      </TR>
      <TR>
            <TD>Item 3</TD> <TD>Item 4</TD> <TD>Item 5</TD>
      </TR>
</TABLE>
```

Item 1	Item 2	
Item 3	Item 4	Item 5

Demonstration of Headers <TH>

Quick Reference

The following will render a table with headers

```
<TABLE BORDER>
      <TR>
            <TH>Head1</TH> <TH>Head2</TH> <TH>Head3</TH>
      </TR>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

Head1	Head2	Head3
A	B	C
D	E	F

Demonstration of COLSPAN and <TH>

Quick Reference

The following will render a table with headers that span two columns.

```
<TABLE BORDER>
      <TR>
            <TH COLSPAN=2>Head1</TH>
            <TH COLSPAN=2>Head2</TH>
      </TR>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
      </TR>
      <TR>
            <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
      </TR>
</TABLE>
```

Head1		Head2	
A	B	C	D
E	F	G	H

Demonstration of multiple headers and COLSPAN

Quick Reference

The following will render a table with multiple headers, one set of which is spanning two columns

```
<TABLE BORDER>
      <TR>
            <TH COLSPAN=2>Head1</TH>
            <TH COLSPAN=2>Head2</TH>
      </TR>
      <TR>
            <TH>Head 3</TH> <TH>Head 4</TH>
            <TH>Head 5</TH> <TH>Head 6</TH>
      </TR>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD> <TD>D</TD>
      </TR>
      <TR>
            <TD>E</TD> <TD>F</TD> <TD>G</TD> <TD>H</TD>
      </TR>
</TABLE>
```

Head1		Head2	
Head 3	**Head 4**	**Head 5**	**Head 6**
A	B	C	D
E	F	G	H

Demonstration of Side Headers

Quick Reference

The following will render a table with the headers at the side.

```
<TABLE BORDER>
      <TR><TH>Head1</TH>
            <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD></TR>
      <TR><TH>Head2</TH>
            <TD>Item 4</TD> <TD>Item 5</TD> <TD>Item 6</TD></TR>
      <TR><TH>Head3</TH>
            <TD>Item 7</TD> <TD>Item 8</TD> <TD>Item 9</TD></TR>
</TABLE>
```

Head1	Item 1	Item 2	Item 3
Head2	Item 4	Item 5	Item 6
Head3	Item 7	Item 8	Item 9

Demonstration of side headers with ROWSPAN

Quick Reference

The following will render a table with side headers, one of which spans multiple rows.

```
<TABLE BORDER>
      <TR><TH ROWSPAN=2>Head1</TH>
            <TD>Item 1</TD> <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item
            4</TD>
      </TR>
            <TR><TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD>
            <TD>Item 8</TD>
      </TR>
      <TR><TH>Head2</TH>
            <TD>Item 9</TD> <TD>Item 10</TD> <TD>Item 3</TD> <TD>Item
            11</TD>
      </TR>
</TABLE>
```

Head1	Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7	Item 8
Head2	Item 9	Item 10	Item 3	Item 11

Sample table using all of the above

Quick Reference

The following will render a table using all of the above attributes.

```
<TABLE BORDER>
      <TR>  <TD><TH ROWSPAN=2></TH>
            <TH COLSPAN=2>Average</TH></TD>
      </TR>
      <TR>  <TD><TH>Height</TH><TH>Weight</TH></TD>
      </TR>
      <TR>  <TH ROWSPAN=2>Gender</TH>
            <TH>Males</TH><TD>1.9</TD><TD>0.003</TD>
      </TR>
      <TR>    <TH>Females</TH><TD>1.7</TD><TD>0.002</TD>
      </TR>
</TABLE>
```

		Average	
		Height	Weight
Gender	Males	1.9	0.003
	Females	1.7	0.002

Clever use of ROWSPAN/COLSPAN

Quick Reference

The following wil render a table that uses both ROWSPAN=2 and COLSPAN=2 on side and bottom cells.

```
<TABLE BORDER>
      <TR>
            <TD ALIGN=center ROWSPAN=2 COLSPAN=2>A</TD>
            <TD>1</TD>
            <TD>2</TD>
      </TR>
      <TR>
            <TD>3</TD>
            <TD>4</TD>
      </TR>
      <TR>
            <TD ALIGN=center ROWSPAN=2 COLSPAN=2>C</TD>
            <TD ALIGN=center ROWSPAN=2 COLSPAN=2>D</TD>
      </TR>
      <TR>
      </TR>
</TABLE>
```

<table>
<tr><td rowspan="2">A</td><td>1</td><td>2</td></tr>
<tr><td>3</td><td>4</td></tr>
<tr><td>C</td><td colspan="2">D</td></tr>
</table>

A Table with no borders

Quick Reference

The following will render a table with no borders.

```
<TABLE>
      <TR>  <TD>Item 1</TD> <TD ROWSPAN=2>Item 2</TD> <TD>Item 3</TD>
      </TR>
      <TR>  <TD>Item 4</TD> <TD>Item 5</TD>
      </TR>
</TABLE>
```

Item 1 Item 2 Item 3
Item 4 Item 5

A table with a border of 10

Quick Reference

The following will render a table with a border of 10.

```
<TABLE BORDER=10>
      <TR>  <TD>Item 1</TD> <TD> Item 2</TD>
      </TR>
      <TR>  <TD>Item 3</TD> <TD>Item 4</TD>
      </TR>
</TABLE>
```

Item 1	Item 2
Item 3	Item 4

CELLPADDING and CELLSPACING

Quick Reference

The following renders a table using `CELLPADDING`, but no `CELLSPACING`

```
<TABLE BORDER CELLPADDING=10 CELLSPACING=0>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

The following renders a table using `CELLSPACING` but no `CELLPADDING`

```
<TABLE BORDER CELLPADDING=0 CELLSPACING=10>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

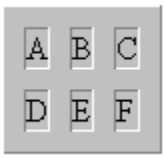
The following renders a table using `CELLPADDING` and `CELLSPACING`

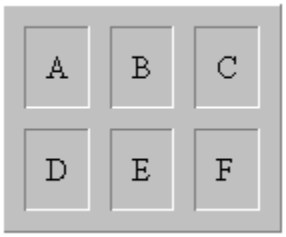
```
<TABLE BORDER CELLPADDING=10 CELLSPACING=10>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

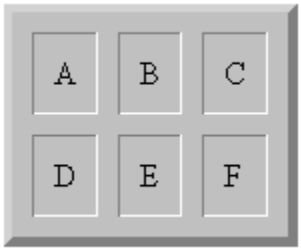
The following renders a table using `CELLSPACING`. `CELLPADDING` and specifying a `BORDER`.

```
<TABLE BORDER=5 CELLPADDING=10 CELLSPACING=10>
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
```

A	B	C
D	E	F


A B C
D E F


A
B
C
D
E
F


A
B
C
D
E
F

Multiple lines in a table

Quick Reference

The following will render a table with multiple lines of text in cells.

```
<TABLE BORDER>
      <TR>
            <TH>January</TH>
            <TH>February</TH>
            <TH>March</TH>
      </TR>
      <TR>
            <TD>This is cell 1</TD>
            <TD>Cell 2</TD>
            <TD>Another cell,<br> cell 3</TD>
      </TR>
      <TR>
            <TD>Cell 4</TD>
            <TD>and now this<br>is cell 5</TD>
            <TD>Cell 6</TD>
      </TR>
</TABLE>
```

January	February	March
This is cell 1	Cell 2	Another cell, cell 3
Cell 4	and now this is cell 5	Cell 6

ALIGN=left|right|center

Quick Reference

The following will render a table showing different possible alignments of text.

NOTE : this formatting can be applied to individual cells or whole rows.

```
<TABLE BORDER>
      <TR>
            <TH>January</TH>
            <TH>February</TH>
            <TH>March</TH>
      </TR>
      <TR ALIGN=center>
            <TD>all aligned center</TD>
            <TD>Cell 2</TD>
            <TD>Another cell,<br> cell 3</TD>
      </TR>
      <TR>
            <TD ALIGN=right>aligned right</TD>
            <TD ALIGN=center>aligned to center</TD>
            <TD>default,<br>aligned left</TD>
      </TR>
</TABLE>
```

January	February	March
all aligned center	Cell 2	Another cell, cell 3
aligned right	aligned to center	default, aligned left

VALIGN=top|bottom|middle

Quick Reference

The following will render a table showing different possible vertical text alignments possible within table cells.

NOTE : this formatting can be applied to individual cells or whole rows.

```
<TABLE BORDER>
      <TR>
            <TH>January</TH>
            <TH> <TH>March</TH>
                    </TR>
      <TR VALIGN=top>
            <TD>all aligned to top</TD>
            <TD>and now this<br>is cell 2</TD>
            <TD>Cell 3</TD>
      </TR>
      <TR>
            <TD VALIGN=top>aligned to the top</TD>
            <TD VALIGN=bottom>aligned to the bottom</TD>
            <TD>default alignment,<br>center</TD>
      </TR>
</TABLE>
```

January	February	March
all aligned to top	and now this is cell 2	Cell 3
aligned to the top	aligned to the bottom	default alignment, center

CAPTION=top|bottom

Quick Reference

The following will render a table with a caption at the top.

```
</TABLE BORDER>
<CAPTION ALIGN=top>A top CAPTION</CAPTION>
      <TR>
            <TH>January</TH>
            <TH>February</TH>
            <TH>March</TH>
      </TR>
      <TR>
            <TD>This is cell 1</TD>
            <TD>Cell 2</TD>
            <TD>Another cell,<br> cell 3</TD>
      </TR>
</TABLE>
```

The following will render a table with a caption at the bottom

```
<TABLE BORDER>
<CAPTION ALIGN=bottom>A bottom CAPTION</CAPTION>
      <TR>
            <TH>January</TH>
            <TH>February</TH>
            <TH>March</TH>
      </TR>
      <TR>
            <TD>This is cell 1</TD>
            <TD>Cell 2</TD>
            <TD>Another cell,<br> cell 3</TD>
      </TR>
</TABLE>
```

A top CAPTION

January	February	March
This is cell 1	Cell 2	Another cell, cell 3

January	February	March
This is cell 1	Cell 2	Another cell, cell 3

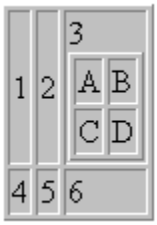
A bottom CAPTION

Nested Tables

Quick Reference

The following will render a table within a table. Table ABCD is inside table 123456.

```
<TABLE BORDER>
      <TR> <!-- ROW 1, TABLE 1 -->
            <TD>1</TD>
            <TD>2</TD>
            <TD>3
            <TABLE BORDER>
                  <TR> <!-- ROW 1, TABLE 2 -->
                        <TD>A</TD>
                        <TD>B</TD>
                  </TR>
                  <TR> <!-- ROW 2, TABLE 2 -->
                        <TD>C</TD>
                        <TD>D</TD>
                  </TR>
            </TABLE>
            </TD>
      </TR>
      <TR> <!-- ROW 2, TABLE 1 -->
            <TD>4</TD>
            <TD>5</TD>
            <TD>6</TD>
      </TR>
</TABLE>
```


1
2
3
A
B
C
D
4
5
6

Setting table width

Quick Reference

The following will render a table of width 50% (of page width).

```
<TABLE BORDER WIDTH="50%">
      <TR><TD>Width=50%</TD><TD>Width=50%</TD>
      </TR>
      <TR><TD>3</TD><TD>4</TD>
      </TR>
</TABLE>
```

Note that if the cells contain non-identical width data, it affects the cell width relative to the table width :

```
<TABLE BORDER WIDTH="50%">
      <TR><TD>Item width affects cell size</TD><TD>2</TD>
      </TR>
      <TR><TD>3</TD><TD>4</TD>
      </TR>
</TABLE>
```

NOTE : This table would appear aligned to the left of the page and half the width of the page

Width=50%	Width=50%
3	4

NOTE : This table would appear aligned to the left of the page and half the width of the page

Item width affects cell size	2
3	4

Centering a table

Quick Reference

The following would render a table in the center of the page.

```
<CENTER>
<TABLE BORDER WIDTH="50%">
      <TR>
            <TD>A</TD> <TD>B</TD> <TD>C</TD>
      </TR>
      <TR>
            <TD>D</TD> <TD>E</TD> <TD>F</TD>
      </TR>
</TABLE>
</CENTER>
```

NOTE : This table would be rendered aligned in the center of the viewing page.

A	B	C
D	E	F

Table width and nested tables

Quick Reference

The following will render two nested tables, using table width attribute to specify the size for the secondary table

```
<TABLE BORDER WIDTH="50%">
      <TR><TD>Item 1</TD><TD>Item 2</TD>
      </TR>
      <TR><TD>
            <TABLE BORDER WIDTH=100%>
                  <TR><TD>Item A</TD><TD>Item B</TD>
                  </TR>
            </TABLE>
            </TD>
            <TD>Item 4</TD>
      </TR>
</TABLE>
```

Item 1		Item 2
Item A	Item B	Item 4

Table Height

Quick Reference

The following will render a table of height 15% (relative to the viewing page)

```
<TABLE BORDER WIDTH="50%" HEIGHT="15%">
      <TR><TD>HEIGHT=15%</TD> <TD>Item 2</TD>
      </TR>
      <TR><TD>3</TD><TD>4</TD>
      </TR>
</TABLE>
```

NOTE : This table would be rendered at a height of 15% relative to the viewing page.

HEIGHT=15%	Item 2
3	4

Controlling the Document Background

Attributes See Also Quick Reference

Recent versions of the proposed HTML 3.0 spec. have added a **BACKGROUND** attribute to the BODY element. The purpose of this attribute is to specify a URL pointing to an image that is to be used as a background for the document. In Netscape , this background image is used to tile the full background of the document-viewing area. Thus specifying:

```
<BODY BACKGROUND="URL or path/filename.gif">
Document here
</BODY>
```

would cause whatever text, images, etc. appeared in that document to be placed on a background consisting of the (filename.gif) graphics file being tiled to cover the viewing area, much like bitmaps are used for Windows wallpaper.

NOTE : This is included in the HTML 3.0 specification, but at present is only supported by Netscape 1.1 and above. While Netscape would use the file as a background, other browsers wouldn´t.

The **BGCOLOR** attribute

This attribute to BODY is not currently in the proposed HTML 3.0 specification, but is supported by Netscape 1.1 and above and is being considered for inclusion in the HTML 3.0 spec. Essentially, it changes the color of the background without having to specify a separate image that requires another network access to load. The format that Netscape 1.1 understands is:

```
<BODY BGCOLOR="#rrggbb">
Document here
</BODY>
```

Where "#rrggbb" is a hexadecimal red-green-blue triplet used to specify the background color. See the Color Table for examples of colors together with their #rrggbb values.

Clearly, once the background colors/patterns have been changed, it will be necessary to also be able to control the foreground to establish the proper contrasts. The following attributes are also recognized as part of the BODY element by Netscape 1.1.

TEXT

This attribute is used to control the color of all the normal text in the document. This basically consists of all text that is not specially colored to indicate a link. The format of TEXT is the same as that of BGCOLOR.

```
<BODY TEXT="#rrggbb">
Document here
</BODY>
```

LINK, **VLINK**, and **ALINK** attributes

These attributes let you control the coloring of link text. VLINK stands for visited link, and ALINK

stands for active link. The default coloring of these is: `LINK=blue`, `VLINK=purple`, and `ALINK=red`. Again, the format for these attributes is the same as that for `BGCOLOR` and `TEXT`.

```
<BODY LINK="#rrggbb" VLINK="#rrggbb" ALINK="#rrggbb">
Document here
</BODY>
```

Coloring Considerations.

Since these color controls are all attributes of the `BODY` element, they can only be set once for the entire document. Document color cannot be changed partially through a document.

Setting a background image requires the fetching of an image file from a second HTTP connection, it will ***slow down*** the perceived speed of document loading. ***None*** of the document can be displayed until the image is loaded and decoded. Needless to say, keep background images small.

If the Auto Load Images option is turned off, background images will not be loaded. If the background image is not loaded for any reason, and a `BGCOLOR` was not also specified, then any of the foreground controlling attributes (`LINK`, `VLINK`, and `ALINK`) will be ignored. The idea behind this is that if the requested background image is unavailable, or not loaded, setting requested text colors on top of the default gray background may make thedocument unreadable.

The following attributes are included in the HTML 3.0 specification for controlling the background of the document.

BACKGROUND
TEXT
LINK
VLINK
ALINK
BGCOLOR is not included in the spec. as yet, but is supported by recent versions of the Netscape Navigator.

Background color chart

Background Colors - Example chart

Quick Reference

This chart shows a variety of possible colors, together with their #rrggbb hexadecimal triplet values. All could be used as valid HTML document backgrounds.

RGB TRIPLET COLOR CHART

E-Mail-ware by Doug Jacobson

FFFFFF	FFFFCC	FFFF99	FFFF66	FFFF33	FFFF00	FFCCFF	FFCCCC	
FF9933	FF9900	FF66FF	FF66CC	FF6699	FF6666	FF6633	FF6600	
FF0099	FF0066	FF0033	FF0000	CCFFFF	CCFFCC	CCFF99	CCFF66	
CC99FF	CC99CC	CC9999	CC9966	CC9933	CC9900	CC66FF	CC66CC	
CC3333	CC3300	CC00FF	CC00CC	CC0099	CC0066	CC0033	CC0000	
99CC99	99CC66	99CC33	99CC00	9999FF	9999CC	999999	999966	
9933FF	9933CC	993399	993366	993333	993300	9900FF	9900CC	
66FF33	66FF00	66CCFF	66CCCC	66CC99	66CC66	66CC33	66CC00	
666699	666666	666633	666600	6633FF	6633CC	663399	663366	
33FFFF	33FFCC	33FF99	33FF66	33FF33	33FF00	33CCFF	33CCCC	
339933	339900	3366FF	3366CC	336699	336666	336633	336600	
330099	330066	330033	330000	00FFFF	00FFCC	00FF99	00FF66	
0099FF	0099CC	009999	009966	009933	009900	0066FF	0066CC	
003333	003300	0000FF	0000CC	000099	000066	000033	EE0000	
110000	00EE00	00DD00	00BB00	00AA00	008800	007700	005500	
000077	000055	000044	000022	000011	EEEEEE	DDDDDD	BBBBBB	
FFCC99	FFCC66	FFCC33	FFCC00	FF99FF	FF99CC	FF9999	FF9966	
FF33FF	FF33CC	FF3399	FF3366	FF3333	FF3300	FF00FF	FF00CC	
CCFF33	CCFF00	CCCCFF	CCCCCC	CCCC99	CCCC66	CCCC33	CCCC00	
CC6699	CC6666	CC6633	CC6600	CC33FF	CC33CC	CC3399	CC3366	
99FFFF	99FFCC	99FF99	99FF66	99FF33	99FF00	99CCFF	99CCCC	
999933	999900	9966FF	9966CC	996699	996666	996633	996600	
990099	990066	990033	990000	66FFFF	66FFCC	66FF99	66FF66	
6699FF	6699CC	669999	669966	669933	669900	6666FF	6666CC	
663333	663300	6600FF	6600CC	660099	660066	660033	660000	
33CC99	33CC66	33CC33	33CC00	3399FF	3399CC	339999	339966	
3333FF	3333CC	333399	333399	333333	333300	3300FF	3300CC	
00FF33	00FF00	00CCFF	00CCFF	00CC99	00CC66	00CC33	00CC00	
006699	006666	006633	006633	0033FF	0033CC	003399	003366	
DD0000	BB0000	AA0000	AA0000	770000	550000	440000	220000	
004400	002200	001100	001100	0000DD	0000BB	0000AA	000088	
AAAAAA	888888	777777	555555	444444	222222	111111	000000	

If you find this chart useful, please send e-mail to jacobson@phoenix.phoenix.net

NOTE : The colors shown in this chart cannot be guaranteed. Due to resolution/color depth

differences of different Windows video drivers, colors may appear different on either the authors system or the end users system. This should be considered when using any color in documents.

Dynamic Documents

See Also Quick Reference

Currently, Netscape 1.1 and above supports a couple of different mechanisms for dynamic documents. These are documents that are updated on a periodic, or frequent basis)

These mechanisms are called "server push" and "client pull", and are based on existing standards (including the standard MIME multipart mechanism and the HTML 2.0 `META` element).

Server push

The server sends down a chunk of data; the browser display the data but leaves the connection open; whenever the server wants it sends more data and the browser displays it, leaving the connection open; at some later time the server sends down yet more data and the browser displays it; etc.

Client pull

The server sends down a chunk of data, including a directive (in the HTTP response or the document header) that says "reload this data in 5 seconds", or "go load this other URL in 10 seconds". After the specified amount of time has elapsed, the client does what it was told -- either reloading the current data or getting new data.

In server push, a HTTP connection is held open for an indefinite period of time (until the server knows it is done sending data to the client and sends a terminator, or until the client interrupts the connection). In client pull, HTTP connections are never held open; rather, the client is told when to open a new connection, and what data to fetch when it does so.

In server push, the magic is accomplished by using a variant of the MIME message format "multipart/mixed", which lets a single message (or HTTP response) contain many data items. In client pull, the magic is accomplished by an HTTP response header (or equivalent HTML element) that tells the client what to do after some specified time delay.

Server Push/Client Pull considerations.

Server push is generally more efficient than client pull, since a new connection doesn't need to be opened for each new piece of data. Since a connection is held open over time, even when no data is being transferred, the server must be willing to accept dedicated allocation of a TCP/IP port, which may be an issue for servers with a sharply limited number of TCP/IP ports.

Client pull is generally less efficient, since a new connection must be opened for each new piece of data. However, no connection is held open over time.

Note that in real world situations it is common for establishment of a new connection to take a significant amount of time -- i.e., one second or more. Given that this is the case, server push will probably be generally preferable for end-user performance reasons, particularly for information that is frequently updated.

Another consideration is that the server has comparatively more control in the server push situation than in the client pull situation. One example: there is one distinct open connection for each instance of server push in use, and the server can elect to (for example) shut down such a connection at any time (e.g., via a cron daemon) without requiring a whole lot of logic in the server. On the other hand, the same application using client pull will look like many independent connections to the server, and the server may need to have a considerable level of complexity in order to manage the situation

(e.g., associating client pull requests with particular end users to figure out who to stop sending new "Refresh" headers to).

An Important Note On Server Push And Shell Scripts: If a CGI program is written as a shell script, and the script implements some form of server push where the connection is expected to be open for a long time (e.g. an infinitely long stream of images representing live video), then the shell script normally will not notice when/if the user severs the connection on the client side (e.g., by pressing the "Stop" button) and will continue running. This is bad, as server resources will be thereafter consumed wastefully and uselessly. The easiest way to work around this shell script limitation is to implement such CGI programs using a language like Perl or C -- such programs will terminate properly when the user breaks the connection.

Server Push
Client Pull
<META> Element

Dynamic Documents - Server Push

See Also Quick Reference

Server push is the other dynamic document mechanism, complementing client pull.

In contrast to client pull, server push takes advantage of a connection that's held open over multiple responses, so the server can send down more data any time it wants. The obvious major advantage is that the server has total control over when and how often new data is sent down. Also, this method can be more efficient, since new HTTP connections don't have to be opened all the time. The downside is that the open connection consumes a resource on the server side while it's open (only when the server knows it wants this to happen, though). Also, server push has two other advantages: one is that a server push is easily interruptible (you can just hit "Stop" and interrupt the connection).The other advantage will be discussed later.

First, a short review: the MIME message format is used by HTTP to encapsulate data returned from a server in response to a request. Typically, an HTTP response consists of only a single piece of data. However, MIME has a standard facility for representing many pieces of data in a single message (or HTTP response). This facility uses a standard MIME type called "multipart/mixed"; a multipart/mixed message looks something like:

```
Content-type: multipart/mixed;boundary=ThisRandomString

--ThisRandomString
Content-type: text/plain

Data for the first object.

--ThisRandomString
Content-type: text/plain

Data for the second and last object.

--ThisRandomString--
```

The above message contains two data blocks, both of type "text/plain". The final two dashes after the last occurrence of "ThisRandomString" indicate that the message is over; there is no more data.

For server push we use a variant of "multipart/mixed" called "multipart/x-mixed-replace". The "x-" indicates this type is experimental. The "replace" indicates that each new data block will cause the previous data block to be replaced -- that is, new data will be displayed instead of (not in addition to) old data.

Here's an example of "multipart/x-mixed-replace" in action:

```
Content-type: multipart/x-mixed-replace;boundary=ThisRandomString

--ThisRandomString
Content-type: text/plain
```

```
Data for the first object.

--ThisRandomString
Content-type: text/plain

Data for the second and last object.

--ThisRandomString--
```

The key to the use of this technique is that the server does not push the whole "multipart/x-mixed-replace" message down all at once but rather sends down each successive data block whenever it sees fit. The HTTP connection stays open all the time, and the server pushes down new data blocks as rapidly or as infrequently as it wants, and in between data blocks the browser simply sits and waits for more data in the current window. The user can even go off and do other things in other windows; when the server has more data to send, it just pushes another data block down the pipe, and the appropriate window updates itself.

Here's exactly what happens:

Following in the tradition of the standard "multipart/mixed", "multipart/x-mixed-replace" messages are composed using a unique boundary line that separates each data object. Each data object has its own headers, allowing for an object-specific content type and other information to be specified.

The specific behavior of "multipart/x-mixed-replace" is that each new data object replaces the previous data object. The browser gets rid of the first data object and instead displays the second data object.

A "multipart/x-mixed-replace" message doesn't have to end! That is, the server can just keep the connection open forever and send down as many new data objects as it wants. The process will then terminate if the user is no longer displaying that data stream in a browser window or if the browser severs the connection (e.g. the user presses the "Stop" button). We expect this will be the typical way people will use server push.

The previous document will be cleared and the browser will begin displaying the next document when the "Content-type" header is found, or at the end of the headers otherwise, for a new data block.

The current data block (document) is considered finished when the next message boundary is found.

Together, the above two items mean that the server should push down the pipe: a set of headers (most likely including "Content-type"), the data itself, and a separator (message boundary). When the browser sees the separator, it knows to sit still and wait indefinitely for the next data block to arrive.

Putting it all together, here's a Unix shell script that will cause the browser to display a new listing of processes running on a server every 5 seconds:

```
#!/bin/sh
echo "HTTP/1.0 200"
echo "Content-type: multipart/x-mixed-replace;boundary=---
ThisRandomString---"
echo ""
echo "---ThisRandomString---"
while true
```

```
do
echo "Content-type: text/html"
echo ""
echo "h2Processes on this machine updated every 5 seconds/h2"
echo "time: "
date
echo "p"
echo "plaintext"
ps -el
echo "---ThisRandomString---"
sleep 5
done
```

Note that the boundary is sent to the browser before the sleep statement. This ensures that the browser will flush its buffers and display all the data that's been received up to that point to the user.

NCSA HTTPD users must not use any spaces in the content type, this includes the boundary argument. NCSA HTTPD will only accept a single string with no white space as a content type. If any spaces are in the line (besides the one right after the colon) any text after the white space will be truncated.

As an example, the following will work:

```
Content-type: multipart/x-mixed-replace;boundary=ThisRandomString
```

The following will not work:

```
Content-type: multipart/x-mixed-replace; boundary=ThisRandomString
```

The other advantage of server push is that it can be used for individual inlined images! A document that contains an image can be made to update by the server on a regular basis or at any time the server wants. Just have the `SRC` attribute of the `IMG` element point to an URL for which the server pushes a series of images.

If server push is used for an individual inlined image, the image will get replaced inside the document each time a new image is pushed -- the document itself won't get touched (assuming it isn't separately subject to server push) -- poor man's animation inlined into a static document.

Client Pull

Dynamic Documents - Client Pull

See Also Quick Reference

A simple use of client pull is to cause a document to be automatically reloaded on a regular basis. For example, consider the following document :

```
<META HTTP-EQUIV="Refresh" CONTENT=1>
<TITLE>Document ONE</TITLE>

<H1>This is Document ONE!</H1>

Here's some text. <P>
```

If loaded into a browser supporting Dynamic Documents (Netscape 1.1 and above), it would reload itself every second.

Simply put, the `META` element (a standard HTML 3.0 element, for simulating HTTP response headers in HTML documents) is telling the browser that it should pretend that the HTTP response when the document was loaded included the following header:

```
Refresh: 1
```

That HTTP header, in turn, tells the browser to reload (refresh) this document after 1 second has elapsed. If a 12 second delay was required, the following HTML directive would have been used:

```
META HTTP-EQUIV="Refresh" CONTENT=12
```

...which is equivalent to this HTTP response header:

```
Refresh: 12
```

Note: the `META` element should be used on the first line of a HTML document.

A couple of things to notice:

In this example, a new "Refresh" directive (via either the `META` element or the Refresh HTTP response header) is given as a part of every retrieval. This is an important point. Each individual "Refresh" directive is one-shot and non-repeating. The directive doesn't say "go get this page every 6 seconds from now until infinity"; it says "go get this page in 6 seconds".

If continuous reloading is required, the directive needs to be given on each retrieval. If the document only needs to be reloaded once, only give the directive on the first retrieval. Once given the directive, the browser will do the specified retrieval after the specified amount of time. The only way to cause it not to happen is to not have an open window that contains the document.

This also means that if an "infinite reload" situation is set up (as the example above does), the only way it can be interrupted is by pressing the "Back" button or otherwise going to a different URL in the current window (or, equivalently, by closing the current window).

So another thing to do, in addition to causing the current document to reload, is to cause another document to be reloaded in *n* seconds in place of the current document. This is easy. The HTTP response header will look like this:

```
Refresh: 12; URL=http://foo.bar/blatz.html
```

The corresponding `META` element would be:

```
<META HTTP-EQUIV="Refresh" CONTENT="12; URL=http://foo.bar/blatz.html">
```

Important note: the URL needs to be fully qualified (e.g. http://whatever/whatever). That is, don't use a relative URL.

Consider the following example documents, "doc2.html" and "doc3.html", each of which causes the other to load (so if one of them is loaded, the browser will flip back and forth between them indefinitely)

```
<META HTTP-EQUIV=REFRESH CONTENT="1; URL=http://machine/doc3.html">
<TITLE>Document TWO</TITLE>

<H1>This is Document TWO!</H1>

Here's some other text. <P>

<META HTTP-EQUIV=REFRESH CONTENT="1; URL=http://machine/doc2.html">
<TITLE>Document THREE</TITLE>

<H1>This is Document THREE!</H1>

Here's yet more text. <P>
```

On loading one of the documents; the browser will load the other in 1 second, then the first in another second, then the second again in another second, and so on forever.

How do you make it stop? The easiest way is to either close the window, or put a link in the document(s) that points to somewhere else. Remember, any retrieval of any document can cause the whole process to stop at any point in time if a fresh directive isn't issued -- the process only continues as long as each new document causes it to continue. Thus, the content creator has total control.

The interval can be 0 seconds! This will cause the browser to load the new data as soon as it possibly can (after the current data is fully displayed).

The data that is retrieved can be of any type: an image, an audio clip, whatever. One fun thing to envision is 0-second continuous updating of a live image (e.g. a camera feed), or a series of still images. Poor man's animation, kind of. Netscape Communications are considering mounting a camouflaged IndyCam on the prow of Jim Clark's boat and feeding live images to the world using this mechanism.

A "Refresh" header can be returned as part of any HTTP response, including a redirection. So

a single HTTP response can say "go get this URL now, and then go get this other URL in 10 seconds".

This means a continuous random URL generator can be made. Have a normal random URL generator (such as URouLette) that returns as part of its redirection response a "Refresh" directive that causes the browser to get another random URL from the random URL generator in 18 seconds.

See the impressive URouLette at :
http://kuhttp.cc.ukans.edu/cwis/organizations/kucia/uroulette/uroulette.html

Server Push
<META> Element

Quick Reference

A
B
C
D
E
F
G
H
I
J
K
L
M

N
O
P
Q
R
S
T
U
V
W
X
Y
Z

The following elements cover all HTML elements of the level 2.0 specification and those elements that will be in the HTML 3.0 specification, but which are supported by currently available HTML user agents.

<BODY>

<CAPTION>
<CENTER>
<CITE>
<CODE>

<DD>
<DFN>
<DIR>
<DL>
<DT>

<EM>
<EMBED>

<FONT SIZE= ...>
<FORM>

<H ALIGN= >.
<H1>
<H2>
<H3>
<H4>
<H5>
<H6>
<HEAD>
<HP>
<HR>
<HTML>

<I>
<IMG>
<INPUT>
<ISINDEX>

<KBD>

<LI>
<LINK>
<LISTING>

<MARQUEE>
<MENU>
<META>

<NEXTID>
<NOBR>

<OL>
<OPTION>

<P ALIGN=>
<P>
<PLAINTEXT>
<PRE>

<SAMP>
<SELECT>
<STRIKE>
<STRONG>

<TABLE>
<TD>
<TEXTAREA>
<TH>
<TITLE>
<TR>
<TT>

<U>
<UL>

<VAR>

<WBR>

<XMP>

SDI

SDI stands for the Software Development Interface. This is a standard way for programs to communicate with Web browsers.

Document Dialog

The Document dialog gives you an easy way to set document-wide attributes, like colors and background graphics.

The Document dialog is divided into three sections.

Document Information

Title

The name of the document. This will not be displayed in the document, but most browsers will display it on their caption bar. The document title is required.

Base URL Address

This provides a convenient way to record the URL of a document, in case it is read out of context (for example, if some downloads the document and reads it off-line). Relative Hypertext Links within the document will be based on this address. For example, if you use the base address of

```
<BASE HREF="http://www.sausage.com/index.html">
```

then a link like

```
<IMG SRC="gifs/sausage.gif>
```

would be translated as

```
<IMG SRC="http://www.sausage.com/index.html/gifs/sausage.gif>
```

This Document Is A Searchable Index

This generates the **ISINDEX** tag, indicating that this document is an index document. As well as reading it, the reader may search for keywords in it. The document can be queried with a keyword search by adding a question mark to the end of the document address, followed by a list of keywords separated by plus signs.

The ISINDEX tag would normally be generated automatically by the Web Server. If the server does not have a search engine, then this option will do nothing.

URL for Processing Queries

This provides the server with a URL to direct search queries to. This field is optional.

Text to Ask User for Keywords

Replaces the default message of "This is a searchable index. Enter search keywords:" with whatever message you like.

Graphics and Colors

Background Graphic

Specifies an image to use as a background for the document. This image will be tiled across the page, and nothing else will be visible until the browser has rendered it. For this reason, these images should be very small files, and should not be interlaced.

Banner (HTML 3)

The banner stays at the top of the page at all times - it does not scroll with the document. This is useful for logos, tool bars, and so on.

Base Font Size

The size to base all relative Font Size changes on. The default is 3. Base Font Size can range from 1 (the smallest) to 7 (the largest).

Colors

Our color sampler gives you an easy way to choose the colors for your document. Click on the item you want to change, then adjust its color with the RGB sliders, or click the RGB code button to choose the color graphically.

You can change the colors of the following elements:

- **Background** - the document background.
- **Foreground** - the document text.
- **Standard Link** - a hypertext link that the user has not yet followed.
- **Visited Link** - a hypertext link to a page the user has already been to.
- **Active Link Color** - a hypertext link that is currently loading.

Links and Meta

Links to Other Documents

This lets you define relationships between this document and other documents. See the help on the **LINK** element for a deeper discussion of these relationships.

Meta Information

Lets you define **META** information about the document. This is a useful way of providing the Web Server and Browser with information not contained in any other elements.

URL

URL stands for **U**niversal **R**esource **L**ocator. This is an Internet address to a HTML document, a file, a gopher, or any other Internet service.

Unknown topic created by Help-To-Source version 2.00

Missing topic created by Help-To-Source version 2.00

Missing topic created by Help-To-Source version 2.00

Missing topic created by Help-To-Source version 2.00

Button Bar

The Button Bar provides fast access to commonly-used HotDog functions. For more information about what a particular button does, hold the mouse pointer over it for a few seconds and a brief description of the button will appear on the screen.

If you don't want to see the Button bar, it can be hidden from the **View** menu.

See also

Elements Bar
Documents Bar

Format Text Dialog

There are a number of ways to format font and other character attributes in HotDog.

Choose Font from the **Format** menu. This will take to you the Font formatting dialog, where you can select the attributes you desire and change the font size.

Choose Bold, Italics, Underline, Blinking from the **Format** menu

Click one of the formatting buttons on the Elements Bar :

Bold

Italics

Underline

If you highlight text in your document before using one of the above options, the appropriate tags will be inserted at the start and end of the text, so that only the selected text is formatted.

If no text is highlighted, the start and end tags for the attributes will be inserted together.

Character Formatting Attributes

Font Size

Font sizes can be expressed either as a size relative to the base font size, or as an absolute size. Sizes range from 1 to 7; the default size is 3. For example, normal text would be rendered in the base font size (3):

HotDog is a great program

If you formatted this text with **FONT SIZE=+1**, it would be rendered as the base font size + 1, or size 4:

HotDog is a great program.

If you use FONT SIZE=7 (an absolute number), the font would be formatted as Size 7.

HotDog is a great program.

Bold

This formats the text in **Bold** type. The use of Bold is not recommended; you should use Strong instead.

Italics

Formats the text in *Italics*. The use of Italics is not recommended; you should use Emphasized instead.

Underline (HTML 3)

Underlines the text.

Emphasized

This provides some emphasis of the text. Emphasized should be used instead of Italics.

Strong

This provides strong emphasis of the text. Strong should be used instead of Bold.

Blinking (Netscape only)

Makes the text blink on and off when it is displayed. Some people object very strongly to blinking text on (mainly) aesthetic grounds, although it can provide useful emphasis.

Subscript (HTML 3)

Formats the text as a subscript..

Superscript (HTML 3)

Formats the text as a superscript.

Big (HTML 3)

Makes the text big.

Small (HTML 3)

Makes the text small.

See also

Big First Letter
Format Document

Special Items

HotDog lets you insert a number of special codes into your documents.

Date and Time

You can either insert the current date and time, or the publishing date and time. The latter will be updated whenever you Publish the document.

Choose the format you want to display the date and/or time in from the list provided.

File Name

The name of the current file.

HotDog Version

The version of HotDog used to create the file. This number is displayed in the About box.

Windows Version

The version of Windows used. This is returned by the operating system.

User

The person this version of HotDog is registered to. This is set when you register HotDog, and cannot be changed.

Company

The company this version of HotDog is registered to. This can be changed from the HOTDOG.INI file if you're a registered user.

Jump To A Document In This System

This creates a Hypertext Link between the current document and another document on your computer. You can test the link between these "local" documents without being connected to the Internet.

You can create a local hypertext link by choosing Jump To A Document On This System from the **Insert** menu. You can also create the links by dragging files from the HotDog or Windows File Manager.

See also

Hypertext Links
Jump to a Document on Another System
Jump Within This Document
Launch an Internet Service

Jump to a Document on Another System

This creates a hypertext link between the current document and (usually) a document on another computer. You can only test these "external" links when you are connected to the Internet.

You can create an external hypertext link by choosing Jump To A Document On Another System from the **Insert** menu. Fill in all the required details to create the URL for the external document.

External hypertext links can also be made to other Internet elements, such as gophers, FTP servers, and news groups. HotDog provides separate dialogs for news and mail links. This makes it easier for you to keep track of all the different types of URL, by giving news and mail addresses their own drop-down lists.

See also

Hypertext Links
Simple URL Dialog
Jump to a Document in This System
Jump Within This Document

Hypertext Links

Hypertext links let you define a section of text that can be clicked by the user. When they click this text, they will go to another document, download a file, listen to a sound, or perhaps go to an Internet service like a WAIS database.

Hypertext links are defined with the **<A>** Anchor element. Each anchor specifies the address of the document it goes to, with the **HREF** property. Anchors can be images or text, or a combination of the two. For example,

image only

About HotDog text only

About HotDog image and text

Documents on the same system are usually inserted as relative addresses. For example, let's say that your Internet account has the following directories:

public_html	for document files
public_html/graphics	for images

If all your documents are in public_html, you do not have to specify a URL for each one. Instead of

<A HREF="http://www.sausage.com/intro.html"> Sausage Software </A>

you can use

<A HREF="intro.html"> Sausage Software </A>

This works as long as there is a document called intro.html in the same directory as the current document.

Directories can be relative as well. For example, we could use a picture for the above hypertext link:

<A HREF="http://www.sausage.com/graphics/HotDog.gif"> Sausage Software </A>

or

<A HREF="graphics/HotDog.gif"> Sausage Software </A>

This works as long as the current directory has a sub-directory called graphics.

Jump Within This Document

This lets you create a Hypertext Link to a Target in the current document. Targets are most often used in long documents which are divided into several sections.

To create a link to a target in the current document, choose Jump Within This Document from the **Insert** menu or click the Internal button on the button bar. You can then choose the desired target from a list of all targets in the active document.

Launch an Internet Service

This lets you create a Hypertext Link to an Internet service. You might want to take the user to your favorite newsgroup, or let them send mail to you.

The dialog for this option shows a list of common Internet services. For some, like news and mail, you need to enter a specific newsgroup or mail address; for others, you will need to build the URL.

Tags Menu

Use this menu to insert raw HTML tags into your document. The Tags menu has the same effect as using the Tags List to select HTML tags.

If you have selected text in the current document, HotDog will insert beginning and ending tags around the text. This lets you easily apply tags like <CENTER> over multiple elements.

See also

HTML Overview

Interlaced Images

Interlaced images are a good way to display graphics for users with slow Internet connections (ie. modems).

An interlaced image will show a very low-resolution sample of the image initially, which will gradually become clearer. The browser can load the other information on the page while the rest of the image is loading.

To make a GIF interlaced, it has to be saved as an Interlaced GIF in the GIF/89 format. There are a number of shareware utilities available to do this, including LVIEW and POLYVIEW.

ALIGN

The Align attribute is most commonly used to position in-line images, but in HTML 3 and can be applied to most other elements.
There are a number of alignments supported:

left	Makes an image *float* against the left margin. Text will flow around this image.
right	Makes an image *float* against the right margin. Text will flow around this image.
top	Aligns the top of the image with the **top** of the tallest **item** in the line.
texttop	Aligns the top of the image with the **top** of the tallest **text** in the line.
middle	Aligns the middle of the image with the **baseline** of the current line.
absmiddle	Aligns the middle of the image with the **middle** of the current line.
baseline	Aligns the bottom of the image with the **baseline** of the current line.
bottom	The same as baseline.
absbottom	Aligns the bottom of the image with the **bottom** of the current line.

If you choose Alignment from the **Format** menu, HotDog will provide you with a dialog to easily see the effects of each option.

About HotDog

This screen displays version and copyright information about HotDog. It also gives you some information about your system resources.

Free Memory

This is the amount of RAM (Random Access Memory) available to the system. It includes Virtual Memory as set in the 386 Enhanced Section of Windows Control Panel, or the System section for Windows 95 and Windows NT. It is expressed in bytes (one Megabyte is a little over 1 million bytes).

Free Graphics

This is the percentage of free GDI resources. Windows 3.1 or Windows for Workgroups users should be careful if this gets very low (say, below 20%), as their system might crash. This figure is not as relevant for Windows NT and Windows 95 users.

News Dialog

Screen: Launch an Internet Service

This lets you create a hypertext link that will take the user to a UseNet newsgroup.

Name

The newsgroup name, for example alt.elvis.sighting. Click the drop-down arrow at the right of this box to see a list of all the newsgroups you've used before, in alphabetical order.

Description

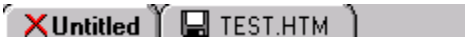
The text that the user will click on to follow the link. This defaults to the name of the newsgroup, but you can change it to anything you like.

Microsoft Color Names

Microsofts new tags feature names for commonly used colors, so you dont have to use hexadecimal RGB triplets (e.g. FFDDDD). The support color names are:

```
Black
Olive
Teal
Red
Blue
Maroon
Navy
Gray
Lime
Fuchsia
White
Green
Purple
Silver
Yellow
Aqua
```

Documents Bar



The Documents Bar provides a convenient way to switch between the documents you are currently editing, and to check at a glance if you have saved your work.

To switch to another document, click the tab containing its name. This has the same effect as choosing the document name from the Window list

Documents that have not changed since the last time they were saved are indicated with the icon next to their name. Unsaved documents are marked with a

If you don't want to see the Documents bar, it can be hidden from the **View** menu.

drag

to drag something, click on it with the left mouse button, and keep the button pressed down. If the item you clicked on can be dragged, a picture will appear when you move the mouse. Move this picture to another object and let go of the mouse button; this is called "dropping".

Saving/Starting Options

Auto-save every xx Minutes

Specify how often you want HotDog to automatically save your files for you. Files are saved into the AutoSave directory specified in File loactions, with the extension .HDB.

Show Handy Hints when HotDog Starts

By default, HotDog will display a handy hint every time it starts. This can be switched off by checking the **Get Rid Of These Things!** box on the Handy Hints screen, and can be switched on and off from the **Show Handy Hints** option.

Create Backup Files when saving

By default, HotDog creates a **.BAK** file before it overwrites any existing files when saving or publishing. This option will disable this behaviour.

Restore Last Session When HotDog Starts

If this option is checked, HotDog will remember the size and position of all open windows when it last closed. When HotDog next starts, these documents will be opened and arranged accordingly.

Note that whether or not this option is selected, HotDog will remember the positions of the Tags List and Special character lists.

Open New Document When HotDog Starts

By default, HotDog will always start with a blank document open. If you do not want this to happen then disable this option.

Editing Options

Default Font Name

This lets you control the font used for editing documents.

Default Font Size

This lets you control the size of the font used for editing documents.

Color tags after nn seconds idle (Pro version)

If this option is enabled, HotDog will display HTML tags in a different color from the rest of the document. Different tag types, for example Netscape or Microsoft tags, will be displayed in a different color from standard HTML tags.

HotDog waits *nn* seconds after you finish typing before going through your document to change the tag colors. If you have a slow computer, or you are working on large documents, you will probably want to set this to a fairly high number, e.g. 30, so that HotDog does not interfere with your work.

Convert Extended Characters while typing

This option is designed primarily for European users. If it is enabled, whenever you type an extended character like an umlaut or acute, HotDog will automatically translate it into the appropriate HTML code (for example, ë). If you don't want this to happen while you're typing, HotDog can also translate all extended characters when you publish your document.

HotDog might take longer to process keystrokes if you use this option.

Show Paragraph Marks (Pro version)

This option will display a character to mark every hard return in your document.

Show Spaces (Pro version)

This option will display a character to mark every space in your document.

Tags in lower case

If this option is enabled, any tags HotDog inserts will be in lower case. HotDog will not automatically convert tags you type into lower case.

Tab Indent (Inch)

Specifies the space (in inches) that HotDog will insert whenever you hit the Tab key.

Undo Depth (Pro version)

Tells HotDog how many previous actions to remember for the multi-level Undo function. HotDog can remember up to 99 actions; the default is 10. The higher this number is, the more memory HotDog will use.

Simple URL Dialog

This is the fastest way to create hypertext links.

URL

The Universal Resource Location, for example http://www.sausage.com. Click the drop-down arrow at the right of this box to see a list of all the URLs you've used before, in alphabetical order.

Description

The text that the user will click on to follow the link. This defaults to the URL, but you can change it to anything you like.

Microsoft Additions to Existing Elements

Quick Reference

The following elements are all new . They are not part of the *Official* HTML specification for level 2.0, but are supported by Microsofts Internet Explorer.

NOTE : The new attributes that are supported by Microsoft browsers may ***only*** be supported by Microsoft browsers. Take care with document styling. If the user is not using Internet Explorer , their view of the document may not be what you intended.

Mail Dialog

This lets you create a hypertext links that will let the user create an e-mail message to the specified address.

Mail Address

The e-mail address of the user, for example sales@sausage.com. Click the drop-down arrow at the right of this box to see a list of all the e-mail addresses you've used before, in alphabetical order.

Description

The text that the user will click on to follow the link. This defaults to the e-mail address, but you can change it to anything you like.

Big First Letter

This option provides a handy way to begin paragraphs with large letters (usually referred to as drop caps). For example:

> This option provides a...

To use it, select the text you want to begin with a large letter, then choose Big First Letter from the **Format** menu.Note that this formatting will only be visible to browsers that can render the <FONT> tag.

Hypertext Target Dialog

Hypertext Targets let you jump to specific locations within documents. You can jump to a target in the same document, or another document. Targets are most often used in long documents which are divided into several sections.

To create a hypertext target, choose Target from the **Insert** menu, or click the Target button on the Button Bar.

To create a link to a target in the current document, choose Jump Within This Document from the **Insert** menu or click the Internal button on the button bar. You can then choose the desired target from a list of all targets in the current document.

To link to a target in a different document, choose Jump to a Document in This System or Jump to a Document on Another System from the **Insert** menu to create the initial link. Specify the target name after the file name, with a #. For example,

<A HREF="document.htm"> links to a document on the same system.

If this document contained a target called "contents", you would use:

<A HREF="document.htm#contents">

Target Syntax

The current HTML 3 draft defines targets with the **ID** token. For example,

<P ID="contents"> will make a paragraph a target, with the name contents.

Previous versions of HTML used the **NAME** token in an **Anchor**. For example,

<A NAME="contents">

HotDog defaults to the first (HTML 3) method. If you prefer to use the older method, you can change the default target code from the General options section of the Options screen.

Introduction to HTML

Hyper**T**ext **M**arkup **L**anguage is a way of adding various attributes to plain text files which are published on the World Wide Web. HTML lets you mix graphics with text, change the appearance of text, and create hypertext documents which interact with the user.

HTML is based around the concept of "tags". A tag looks like this: <B>. Most HTML functions have an opening and closing tag - the tag applies to all text in between. For example, <B> is the tag for "bold". Any text between a <B> and a </B> will be displayed in bold type when the document is viewed by the appropriate browser. So <B>hello world</B> would be displayed as **hello world**.

HotDog will let you type these tags directly into your document, or to apply them using a combination of toolbars, dialog, menus, and pop-up lists. To put the text in the above example in italics, you could type hello world in your document, highlight it with the mouse, then press the Italics button on the **Elements bar**. HotDog would automatically translate this to <I>hello world</I>.

The HTML language has a number of different "flavors", or specifications. Most browsers today support the HTML 2 specification, although at the time of writing this has not yet been finalized by the appropriate Standards Committees. The Netscape Navigator implements its own extensions to HTML 2. These are not supported by all browsers; if you use them then not everyone will be able to read your pages.

HTML 3 is the next generation of the HTML language. At the time of writing, it is still a draft proposal and (if HTML 2 is anything to go by) a long way from being finalized. There are only a few browsers that fully support it, and these have all been developed purely as HTML 3 test tools. Don't be surprised if they're not as powerful or flexible as some of the commercial offerings.

HotDog supports both HTML 2+ and HTML 3, as well as other Netscape and Microsoft extensions to HTML.

One of the key strengths of HTML is that a document conforming to the HTML standard can be understood no matter what sort of software or computer the reader has. For example, the same page can be interpreted by someone using Netscape in Windows, someone using Lynx on UNIX, or even a blind person using special software.

There is much debate on the Internet about what constitutes "good" HTML. The original intention of HTML was to create a universal way of storing and viewing information. The subscribers to this theory see HTML as a content-based language - what's in the document is much more important than how it looks.

New features added to HTML, especially those supported by the Netscape browser, allow authors to create fancy graphical effects. This has spawned a whole new class of HTML "artists", for whom creating aesthetically pleasing pages is the main concern. Unfortunately, if you're using a text-based browser or one that doesn't support some of the special tricks involved in these pages, they won't display properly - or at all.

The first group maintain that standards are there to be followed, and deviations from the standards simply to make pages look pretty are unacceptable. The second group believe that the only way standards will advance is if they're broken, sort of like George Bernard Shaw's idea that "all progress

depends on the unreasonable man".

We've tried to stay out of this debate with HotDog. You can use HotDog to write "standard" HTML, or to implement Netscape and HTML 3 extensions; we're not going to stop you from either. In much the same way that a chainsaw can be used to chop wood, or as an instrument of destruction in B-grade movies, the choice to use HotDog's power for "good" or "evil" is yours and yours alone!

It is worth noting that the best HTML authors manage to create attractive and innovative Web sites that display well on all browsers. This obviously takes more work; it's up to you to decide if you're prepared to put this effort in for the benefit of all Internet users.

See also

HTML Tutorials
HTML Reference

Elements bar

"The Elements Bar has been removed from the Latest Version of HotDog."

You will now find the same or similar functionality on the menubar, **Toolbar**. The toolbar can be customized to contain any of HotDogs functions or Events. If you want to see more information on this then check out the tutorials or Toolbar section of this Help document.

The Toolbar provides fast access to commonly-used HotDog functions and tags. For more information about what a particular button on the Toolbar does, hold the mouse pointer over it for a few seconds and a brief description of the button will appear on the screen. If you don't want to see a Toolbar , it can be hidden by looking in the **View Toolbar** menu.

See also

HTML Tutorials

HTML Tutorials

The following lessons are available:

Creating an HTML Document
Adding Hypertext Links
Adding Pictures

See also

HTML Overview

Creating an HTML Document

This tutorial will take you through creating your first HTML document. We'll create a simple personal home page.

1. Start HotDog. You will see the default editing template, which looks like this:

```
<HTML>
<HEAD>
<TITLE> type_Document_Title_here </TITLE>
</HEAD>
<BODY>

</BODY>
</HTML>
```

2. The first step we recommend is saving your work. This will put the file name on the documents bar at the end of the screen, and make it easier for HotDog's autosave feature to recover files.

To save your file, choose **Save** from the File menu, or click the button on the Elements Bar.

Enter a name for the file. HotDog does not yet support long file names.

3. All HTML documents must have a title. You can type this directly into your document, where it says `type_Document_Title_here`, or you can specify it from the Format Document screen.

To give your document a title, choose **Document** from the Format menu

Enter the title in the box provided. The document title will appear in the caption bar of most browsers when your document is viewed; it will not otherwise be visible to users.

You can enter any text you like for the document title.

4. The content of your document must come after the <BODY> tag. Everything before the <BODY> is information that describes your document to Web Browser and Server software.

To enter information in your document that is visible to the user, position the cursor between the <BODY> opening tag and the </BODY> closing tag.

5. It often pays to give your document a heading that will be visible to the user. In most cases, this will be the same as or similar to the document title.

To create a heading for your document, just type the text you want for the heading. Then highlight it with the mouse, and click one of the buttons on the Elements Bar marked H1 through H6.

H1 is the largest size heading, which you would normally use at the start of a document. H6 is very small. Click the H1 button now.

6. We'll divide our document into two sections: Who Am I?, and Hobbies and Interests. Each of these sections will need its own heading. Let's do Who Am I first.

Type the text: Who Am I? into your document. As with step 5, highlight the text. This time, instead of using the H1 button, we'll use the next size down. Click the H2 button

7. Now we need to enter some information into the document. A paragraph about who you are and what you do is probably enough.

To create a Paragraph, click the ¶ button on the Elements Bar. This will insert a <P> tag.

Next type the paragraph. It will probably make things clearer for you if you get in the habit of putting a </P> closing tag at the end of each paragraph, but this is not currently required.

8. At this stage, you should have a document containing two headings, and a paragraph of text. Now is probably a good time to take a look at how this will actually be displayed on the World Wide Web.

To preview your work, click the Preview button on the Button Bar or choose **Preview Document** from the File menu.

If this is the first time you've used HotDog, you will need to tell it where to find your browser. If you don't have a browser, you will need to download one from our Web site.

Select the browser from the file dialog, then click OK. HotDog will start your browser and display a copy of your document. Notice the difference in size between the H1 and H2 text.

To return to your document, just minimize your browser. HotDog will interact with it, so you don't have to start a new copy of your browser every time.

9. Follow steps 6-8 again to create another paragraph for your Hobbies and Interests.

10. By now, you should have a document containing two paragraphs of text, each with its own heading. Let's provide a visual clue to the user to separate these two paragraphs further.

Position the cursor before the <H2> tag that starts the heading for the second paragraph.

Choose **Horizontal Line** from the Insert menu, or click the  button on the Elements Bar. This will insert an <HR> tag at the cursor position.

Preview your document again. There should be a recessed horizontal line dividing the two paragraphs.

11. Save your work again. You have created a basic HTML document!

Further Tutorials

Adding Hypertext Links
Adding Pictures

What's Sausage Software's WWW address!

- **US:** http://**www.sausage.com**
- **Europe** http://**www.fourthnet.co.uk/hotdog/index.html**
- **Australia:** http://**www.sausage.com.au**

Adding Hypertext Links

This Tutorial will take you through creating several different types of Hypertext Link in your document. You should be comfortable with the concepts explained in the Creating HTML Documents tutorial.

Hypertext links make your documents "interactive". When the user clicks on the link, something will happen. This tutorial will show you how to create links that:


Take the user to another document


Take the user to another point within this document


Let the user download a file


Let the user send mail to you


Take the user to a UseNet newsgroup

1. Start HotDog and open a document. If you've just completed the Creating HTML Documents tutorial, keep working in that document.

2. The first link we'll create is to an HTML document somewhere else on the Internet. The document we'll use is the Sausage Software home page at http://www.sausage.com

> To create a link to a document on another system, choose **Jump To A Document On Another System** from the Insert menu, or click the **External** button in the Button Bar.
>
> You will be taken to the **Build External Hypertext Link** dialog. Enter the document information here:
>
> Resource type is http
> Host Address is www.sausage.com
>
> Leave the other fields blank.
>
> As you typed the above information, you would have seen the contents of the URL box change. This is the full URL address for the hypertext link. When you choose OK, this URL will be added to the drop-down list for future reference.
>
> Move down to the Description box. This lets you specify the text that the user will click on to

follow the link. Type Sausage Software's home page here.

Now choose OK. HotDog will insert something like:

```
<A HREF="http://www.sausage.com">Sausage Software's home page</A>
```

into your document.

3. Preview the document to see the results of Step 2. Just click the **Preview** button in the Button Bar. If you're connected to the Internet, when you click the text Sausage Software's home page your browser will take you to this page.

4. Next, we'll create a hypertext link that will take the user to another place in the same document. This is a little more involved, because we have to create a destination for the link, as well as the link itself.

To create a destination target for a hypertext link, position the cursor at the point you want users to jump to, then click the **Target** button in the Button Bar, or choose **Hypertext Target** from the Insert Menu. You will be asked to enter a name for this link. This should be something that makes sense if you want to refer to the link from another document. Type the name and press Enter.

```
<A NAME="test target"></A>
```

Next, move to the place in your document where you want the users to click to go to the destination target.

Click the **Internal** button in the Button Bar, or choose **Jump Within This Document** from the Insert Menu.

You will see a list of all links in the current document. At this stage, there's probably only one, for the link you've just created. Click on this link to highlight it.

Type the text you want the user to click on in the Description box, then press Enter or click the OK button. HotDog should insert something like this:

```
<A HREF="#test target">Go to the test target</A>
```

Preview this again, to test the effects.

5. Now we'll create a link that will let the user download a file. There are a number of ways to do this, but we'll show you the way we think is easiest. As an example, we'll create a link to download the latest version of the HotDog Web Editor.

To create a link to a file, position the cursor at the point where you want to insert the link, then choose **Launch an Internet Service** from the Insert menu, or click the **Internet** button on the Button Bar.

You will see a list of common Internet services. Click the **FTP** button.

You will be taken to the **Build External Hypertext Link** dialog, just like in Step 2. Enter the document information here:

Resource Type is FTP
Host Name is sausage.clever.net
Path is pub/sausage
File Name is hdgsetup.exe

Enter the description for the link, then press Enter.

HotDog will insert something like this:

```
<A HREF="ftp://sausage.clever.net/pub/sausage/hdgsetup.exe">Download
the HotDog Web Editor</A>
```

6. The next type of link we'll create will let the reader send e-mail to you (or anyone else you like).

To create an e-mail link, position the cursor at the point where you want to insert the link, then choose **Launch an Internet Service** from the Insert menu, or click the **Internet** button on the Button Bar.

You will see a list of common Internet services. Click the **Mail** button.

Enter your e-mail address and the link description in the boxes provided, then click OK.

HotDog will insert something like this:

```
<A HREF="mailto:sales@sausage.com">Sausage Software</A>
```

7. Creating a link to a newsgroup is almost exactly the same as creating an e-mail link.

To create a link to a UseNet newsgroup, position the cursor at the point where you want to insert the link, then choose **Launch an Internet Service** from the Insert, or click the **Internet** button on the Button Bar.

You will see a list of common Internet services. Click the **News** button.

Enter the name of the newsgroup and the link description in the boxes provided, then click OK.

HotDog will insert something like this:

```
<A HREF="news:alt.elvis.sighting">Find out if anyone's seen the King!
</A>
```

8. The final type of link we'll create is to a document on your web site. Save the current document, then open a new one by choosing **New** from the File menu or clicking the button.

To create a hypertext link to another document on your web site, choose **Jump to a Document in this System** from the Insert menu.

You will see a standard Windows file dialog. Find the test document that you've just saved and click OK. HotDog will insert something like this:

```
<A HREF="test.htm"></A>
```

You will need to type the description of the link between the `>` and the `</A>`.

Note that this link uses a *relative* reference `"test.htm"`, rather than an absolute reference `HREF="http:/www.sausage.com/pub/sausage/test.htm"`. This link will always jump to test.htm, no matter what web site it's stored on (provided, of course, that a file called test.htm exists in the same directory).

Further Tutorials

Adding Pictures

Adding Pictures

This Tutorial will take you through adding images to your document. You should be comfortable with the concepts explained in the Creating HTML Documents and Adding Hypertext Links tutorials.

This Tutorial will assume that you have some .GIF or .JPG files in your system to use for images.

1. Start HotDog and open a new document.

2. The first type of picture we'll create is called an **In-Line Image**. This is a picture that displays inside your document, but doesn't do anything. If the reader clicks on it, nothing will happen.

> To create an in-line image, position the cursor at the point where you want to insert the image, then choose **Image** from the Insert menu or click the **Image** button on the Button Bar.
>
> You will see the Insert Image dialog. If you want to use an image that you already have in your system, click the button and choose the file.
>
> If you want to link to an image on another system, click the button. This will let you create a hypertext link to the image.
>
> Leave the **Document To Launch** field box blank, and click OK. HotDog will insert something like this:
>
> ```
> <IMG SRC="hr_brass.gif">
> ```
>
> Preview your document to see the effects of this.

3. As well as these "static" in-line images, you can create an image that is also a hypertext link. When the reader clicks on it, they follow the link in the same way as if they clicked on a text link.

> To create a hypertext image, position the cursor at the point where you want to insert the image, then choose **Image** from the Insert menu or click the **Image** button on the Button Bar.
>
> You will see the Insert Image dialog. If you want to use an image that you already have in your system, click the button and choose the file.
>
> If you want to link to an image on another system, click the button. This will let you create a hypertext link to the image.
>
> In the **Document To Launch** box, choose the document the reader will be taken to when they follow the link. As with the image, this can be a file on the current system or an external link. If you've completed the Adding Hypertext Links tutorial, click the drop-down arrow at the right of this box. Some of the links you created in that tutorial will be listed.
>
> When you enter something in the **Document To Launch** box, another box will appear for you to

describe the link. If you fill this in, the image will have some text next to it, which the reader can also click on to follow the link. If you leave it blank, then only the image will be displayed.

When you choose OK, HotDog will insert something like this:

```
<A HREF="hotdog.htm"><IMG SRC="hotdog.gif"> Sausage Software</A>
```

See also

HTML Tutorials