

Almost Everything Communications



# HTML Tags Glossary

**G. Tod Abbott**

A sample of the training materials available from Almost Everything Communications

## Useful HTML Tags

### General notes:

HTML is syntax sensitive. In what follows, I have tried to use straight quotes to show marks that should appear in the tag, and curly quotes for regular quotation marks. You should always use straight quotes (i.e. turn off “Smart Quotes”) in your HTML coding — curly quotes won’t work. In general, straight quotes are only needed for attribute definitions that contain spaces. It is not a bad habit, however, to use them with every such definition (as I have done here).

Most of the tags below are followed by attributes or modifiers to the tag. The syntax for including them is simply to place them inside the opening tag. Any number of attributes can appear in a single tag separated only by spaces. For Example:

```
<FONT FACE = "XXX" SIZE = "n" COLOR =
"#XXXXXX"> Text goes here
</FONT>
```

HTML ignores extra spaces and line returns. To add an extra space (more than one between characters or objects) use `&nbsp;`; which is the character entity for “non-breaking space.” Several of these can be lined up to add space on a line. To add an extra line or begin a new paragraph, use `<P></P>`. To skip to the next line (without skipping an extra one) use `<BR>`.

```
<A HREF = "FILENAME.HTM"></A>
```

Defines a link to another file. What appears between the tags is what serves as the link button to the new file. Can be modified with the addition of JavaScript event detectors and actions.

```
<A NAME = "INDEXNAME"></A>
```

An anchor name, that can be referenced by any link to or within the page. For instance, if we define the word “Xylophone” as an anchor name on a dictionary page:

```
<A NAME = "xylophone">Xylophone </A>
```

we can link directly to that part of the page (from either another page or from within the same page) with the link:

```
<A HREF = "DICTION.HTM#xylophone">
```

```
<BODY></BODY>
```

Where the meat of the web document is defined. The body follows the head, and encloses all of the tables, forms, images and everything else to be seen in the browser window. Can be modified by:

“BACKGROUND = “X”” where X is the location and name of an image file to be used as a background image on the page. “BGCOLOR = “#XXXXXX”” where the Xs define a color to serve as the background color defined via Hexadecimal RGB values. “LINK = “#XXXXXX”” where the Xs define the color of any text or image borders to be used as links on the page. “VLINK = “#XXXXXX”” where the Xs define the color of the text or image borders of links the user has already visited. “ALINK = “#XXXXXX”” where the Xs define the color of the text or image border of a link the user is in the process of activating

```
<BR>, <P></P>
```

Basic formatting tags. `<BR>` causes the browser to skip to the start of the next line, while `<P>` causes it to skip a whole line in addition to what the `<BR>` does.

### Character Entities

Some useful character entities:

```
&nbsp; non-breaking space
&#35; number symbol/pound sign (#)
&#36; dollar sign ($)
&#37; percent symbol (%)
&#38; ampersand (&)
&#40; open parenthesis
&#41; close parenthesis
&#42; asterisk (*)
&#43; plus sign (+)
&#45; minus sign/hyphen (-)
&#47; slash (/)
&#60; less than (<)
&#62; greater than (>)
&#61; equals (=)
```

You do not always need to use these, but it is not a bad idea.

```
<DIV></DIV>
```

Sets up a division to be formatted according to a style sheet. Between these tags could appear an image source tag, text, a table, or even a whole page formatted via regular HTML. Modified by: CLASS = “XXXX” identifies the style name that is to apply to the objects or text appearing in this division. ID = “XXXX” gives the division a name, so that it can be addressed and manipulated via scripting. STYLE = “XX:nn; XX:nn;” a place where you can modify any of the style definition attributes you wish. Most useful for entering particular positioning information.

## <FONT></FONT>

Can be used to define the font family, size, or color, of the text that appears between the tags. Used with: "SIZE = "n"" Where "n" is a number between 1 and 7 with 7 yielding the largest font, and 1 giving the smallest. The actual font sizes are determined by the browser preferences. The default font is always number 3. One can also specify font size via "+n," or "-n" where n is the number of places up or down the 1-7 scale you want to move. "COLOR = "#XXXXXX"" defines the color of the text that appears between the tags. "FACE = "X"" where x is the name of the font (or names of the fonts, separated by commas) to be used in displaying the text between the tags.

## <FORM></FORM>

Form elements must appear within a form defined by these tags. HTML formatting holds up, by and large, within these tags, though odd things can happen when trying to precisely place form elements. The form elements that can be used are:

- <INPUT > defines any of a number of input types:
  - T EXT produces a single-line text box
  - BUTTON produces a button
  - CHECKBOX produces a common checkbox
  - RADIO produces a set of checkboxes, only one of which can be selected
  - SUBMIT a button that activates the action defined in the opening form tag
  - RESET a button that resets all the fields in the form to their default values

<SELECT> produces a list of options, each defined by <OPTION> tags

<TEXTAREA> produces a multiple-line text area

See the FORMS example for more detailed information.

## <FRAMESET></FRAMESET>

Begins a frameset definition. Modified by: ROWS = "n,n,n" where the n's are the height in pixels or percentage (n%) of each row in the frameset. COLS = "n,n,n" same thing, except for column width. Note that you must use one or the other of these attributes when defining a frameset. Do not use both at once (for that you will need to nest framesets). Note also that for either of these, an asterisk in the definition will give that frame all the space available after the others are drawn (so it will scale with the window). FRAMEBORDER = "XXX" determines whether or not to show a border between each frame. "NO" turns off the borders (default has borders on). FRAMESPACING = "n" designates in pixels the space between each frame. BORDERCOLOR = "#XXXXXX" determines the color of the border between each frame. BORDER = "n" determines the width in pixels of the border between each frame.

## <FRAME SRC = "XXXXXX">

Determines which file or url to place inside a given frame (which frame it goes into is determined by the order of the Frame tags in relation to the order of the Frameset frame definitions). Is modified by: NAME = "XXXX" gives the frame a name so that links can be directed to it. SCROLLING = "XXX" determines whether or not to draw scrollbars in the frame. Options are YES, NO, or AUTO. Auto draws scroll bars only if the content will not fit in the frame as displayed. NORESIZE tells the browser not to allow the user to move the frame borders around to resize them. Necessary when formatting would not hold up with a differently-sized frame. MARGINWIDTH = "n" determines the amount of space between the sides of the frame and the content. Default is around 10 pixels. MARGINHEIGHT = "n" same thing, only with the top and bottom of the frame.

## <HEAD></HEAD>

The first section of a Web page. This is where the Title appears, as well as any metadata or Style Sheet definitions.

### Headings:

<H1></H1> <H2></H2> <H3></H3> <Hn></Hn>

Heading tags tell the browser to emphasize the text that falls between the tags. Numbers go from 1 to 6, with <H1> being the largest, boldest character, and <H6> being rather smaller and less obvious. Heading defaults are set by the browser preferences, but specific heading attributes can be set via style sheets.

## <HTML></HTML>

Tells the browser what kind of document it is dealing with. With a few exceptions, every document must begin with <HTML> and end with </HTML>.

## <IMG SRC = "IMAGE/XXXXX.GIF">

Identifies an image file to be displayed. Can be modified by: WIDTH = "n" where n is the number in pixels wide the image will be displayed. HEIGHT = "n" where n is the number in pixels high the image will be displayed. Note: be careful with these attributes. If the Width and Height attribute are not in the same ratio as the measurements of the image, the image will be distorted when it is displayed. These are very good for using a transparent GIF image as a spacer. "BORDER = "n"" where X is the width in pixels of the border to appear around the image. "BORDERCOLOR = "#XXXXXX"" where the Xs specify an RGB value in hexadecimal. "ALIGN = "X"" Where X is either "LEFT" or "RIGHT" which determines how the image fits in with the appearing around it. "HSPACE = "n"" where X defines the width in pixels of space between the image and what text or other object appears next to it. "VSPACE = "n"" where X

defines the height in pixels of the space between the image and what comes above or below it. "ALT = "X"" where X is a phrase or name describing the image that will appear in browsers that do not support images, have image loading turned off, or will appear in a "tool tips" window when the cursor passes over it (IE). "USEMAP = "#XXX"" identifies the image as an image map, and tells the browser which map to use in decoding the image.

**Lists:**

<OL></OL>, <UL></UL>, <LI>

<OL> begins an ordered list, that is a list with numbers automatically placed before each item. Ordered lists can be numbered with numerals (1), capital letters (A), lowercase letters (a), or upper (I) or lower (i) case Roman numerals. Choose between these with the TYPE attribute: <OL Type = A>. The START attribute allows you to control the number (or figure) at which the list will start: <OL Start = 3> — this is always a number, regardless of the numbering system of the list.

<UL> begins an unordered list, one with bullets before each item. A Disc, Circle or Square can be selected as the bullet shape. Again, use the TYPE attribute.

<LI> stands for "list item" and indicates the start of each new item for the list.

<MAP NAME = "XXXXXX"></MAP>

Indicates the beginning of an image map, and gives the map a name. Note that the name as it appears in this tag does not include the pound-sign that is needed in the USEMAP tag. The rest of the Image Map tags are:

<AREA SHAPE = "XXX" COORDS = "n,n,n,n"  
HREF = "XXX">

AREA SHAPE calls for either POLY, RECT, or CIRCLE depending on the type of shape you are defining. COORDS calls for the coordinates of the points that define the shape. For CIRCLE, these are the coordinates of the center of the circle, and then the diameter of the circle in pixels. For RECT, these are the coordinates of the top, left corner and the bottom, right corner. For POLY, these are the coordinates of each corner of the complex shape. Netscape supports the DEFAULT shape, which covers any area not covered by another shape. HREF calls for, of course, the URL of the file that you want to be linked to this part of the image.

JavaScript actions can also be placed within this tag (such as onMouseOver, onClick, etc.). In such a case, if you don't need it to link to another file, put "#" into the HREF (there must be an HREF in this tag).

<META NAME=... CONTENTS=...>

Appearing in the HEAD of an HTML document, meta tags specify information about a page, such as its author, its topic,

and a concise description to be displayed by search engines. Attributes determine what sort of information a given tag presents. Meta tags can also be used to redirect browsers to a different page.

<META NAME=KEYWORDS CONTENT="PCB,  
PAH, Ashtabula, Ohio">

This tag would flag the page for search engines looking for any of the terms contained in the content attribute.

Attributes include:

NAME, which specifies the type of information contained in the page as either AUTHOR, DESCRIPTION, or KEYWORDS;

CONTENT, which specifies the what value that type of information contains, such as the author's name, important keywords, or a concise description of the page contents;

HTTP-EQUIV, which, with the value "REFRESH" redirects the browser to a page specified in the CONTENT attribute of the tage after an amount of time also specified in the CONTENT attribute:

<META HTTP-EQUIV=REFRESH  
CONTENT="10;URL=http://www.epa.gov/  
text.html">

This tag would send the browser to the page text.html after a 10 second pause.

<NOFRAMES> </NOFRAMES>

Designates formatting for browsers that do not support frames. All (or at least most) content in the frames would be transferred (by the page author) to standard HTML for this section.

<SCRIPT TYPE = "JavaScript"></SCRIPT>

JavaScript scripts go between these tags. If the script is a function that does not write directly to the page, it should appear in the HEAD of the document. If it is intended to insert something into the page, it should appear in the BODY code precisely where you want the inserted object to appear.

Lines of JavaScript are always ended with a semicolon (;).

Some JavaScript code words to help you read scripts:

function xxxxx(yyy) {...}  
Usually appearing in the HEAD of documents, these are portions of code called by other portions of code. "xxxx" represents the name of the function, while "yyy" represents a variable that will accept a value passed into the function in the function call.

The actions of the function appear between brackets.

```
xxxxx(); or xxxxx(yyy);
```

A function call. A function call always includes parentheses, whether it is passing a value (such as the "yyy" above) to the function or not

```
var xxx;
```

Defines a variable. JavaScript is loosely typed, so in most cases, the type of variable being defined will not be specified, JavaScript assigning the appropriate type for whatever value is passed into the variable.

Variables that are defined within a function apply only to that function, while those defined outside of a function are global.

```
if (i >= 2) {...}
```

A conditional. This checks to see if the variable *i* has a value greater than or equal to two, and if it does, performs the code appearing between the brackets.

Other conditional operators are:

`<=` and `==` as well as `||` (or) and `&&` (and).

```
document.image[2].border
```

Statements like this are specifying windows, frames or elements of pages with the JavaScript document-object model. This statement identifies the border value of the second image that appears on the current page. It is usually possible to figure out what is being identified by simply reading the statements. Other windows, frames, layers, form elements or any other objects that have been named can be specified by name, such as:

```
document.introform.button1
```

where "introform" is the name of a form on the page, and "button1" has been assigned to a form element.

JavaScript can sense and respond to a number of events tied to particular objects. The most commonly used events are "onMouseOver," "onMouseOut," and "onClick." Here is a less than exhaustive list of JavaScript events:

**onClick:** object is clicked

**onMouseOver:** mouse cursor is over the object

**onMouseOut:** mouse cursor moves off the object

**onMouseDown:** mouse button is pressed over object

**onMouseUp:** mouse button is released over object

**onKeyPress:** keyboard key is pressed and released

**onKeyDown:** keyboard key is pressed down

**onKeyUp:** keyboard key is released

**onLoad:** object completes loading from the server

These work with form elements:

**onFocus:** form element is made active

**onBlur:** form element is made inactive

**onAbort:** page or function is canceled

**onReset:** reset button is activated

**onSubmit:** submit button is activated

**onChange:** element value is changed

**onError:** an error occurs in the code

**onSelect:** element is selected

Any JavaScript function can be called by any of these events.

```
<SPAN></SPAN>
```

Used like `<DIV>` to apply styles to text and objects, `<SPAN>` does not insert a line break, as does `<DIV>`. `<SPAN>` is consequently not used to create independent layers, but is useful for applying styles to small portions of text or table cells.

```
<STYLE TYPE = "text/css"></STYLE>
```

Encloses a Cascading Style Sheet style definition section.

There are other types of styles than "text/css" but they are not in general use at this time, such as "text/javascript" which is supported only by Netscape Navigator. This tag goes between the `<HEAD>` and `</HEAD>` tags. Note, it is important that the style sheet definition be enclosed by comment tags as in the example below.

A sample style definition

```
<STYLE TYPE = "text/css">
<!--
.quote {
font-size: 12px;
font-family: Arial, Courier;
color: #000000;
background-color: #CCCCCC;
position: absolute;
left: 511;
top: 60;
width: 275;
height: 66;
z-index: 3;
visibility: hidden;
}
-->
</STYLE>
```

Here ".quote" defines the name of the style (which would be called in the document as "quote"). Most of the attributes are self-explanatory. "position: absolute;" means that the object will be placed precisely at the coordinates set by the "left" and "top" attributes. "position: relative;" would place the object in relation to what comes before and after. Always specify font-size with pixels (px) rather than points (pn). Pixel sizes don't change, while different machines display point sizes differently.

"z-index: n;" determines the layer height of the object. Think of the page as a set of transparent pages set on top of each other. "z-index" determines which page it is on, and thus what it is above and what it is below. "visibility: hidden;" tells the browser to not display this object when the page

loads. This is only really useful for pages on which some sort of scripting will be making objects appear and disappear.

Styles are usually applied with `<DIV></DIV>` or `<SPAN></SPAN>` tags, though they can appear directly within `<IMG>` tags and even Table `<TD></TD>` tags. In each case, the style is applied with the `CLASS` attribute, followed by the name of the style, such as:

```
<DIV CLASS = titl></DIV>
```

Additions can be made to a particular application of a Style with the `STYLE` attribute:

```
<DIV CLASS = titl STYLE = "font-color:
#CCCCCC; font-weight: 800"></DIV>
```

Style sheets can be used to redefine standard HTML tags. For instance, you could define a standard style to be used for all paragraphs marked off with `<P>` tags, or create a custom level of emphasis for `<B>`. This would be useful for altering the format of a page, if you had used `<B>` and later decided italics would be more effective, you could just link all of the pages to a Style sheet that defines `<B>` as non-bold italic. This is also useful to override the default settings for the `<H1>` tags.

Note that each style attribute is named, followed by a colon, and then the value of the attribute is followed by a semicolon. Other than that the formatting is unimportant (do not be surprised to find everything entered on a single line, or without spaces). A given Style definition will probably only have a subset of these attributes. You only need to include the ones you want to specify. There are more attributes to choose from, but these are, I think, the most useful.

The “Cascading” in “Cascading Style Sheets” means that any character formatting you input after you have defined a style will take precedence over the Style sheet definition. For instance, even though you request Arial for the “quote” style, if you use the `<FONT FACE = "COURIER">` tag in text that is otherwise ascribed to the “quote” style, the text covered by that tag will use the different font. You can even change the definition of a style closer to its use in a tag.

```
<LINK REL = STYLESHEET TYPE = "text/css"
HREF = "XXXXXX">
```

Creates a link to an external style sheet document. This is handy for providing a single source for all of the styles used in a site—and thus for easily updating all the formatting throughout the site. Appears within the `HEAD` section of the page.

```
<TABLE></TABLE>
```

A table begins with the `<TABLE>` tag, and then is built of `<TR></TR>` and `<TD></TD>` tags. The Browser merely

reads across and down row, by row, building the table.

Specifying the width of a table or its cells will generally speed up drawing of the table. Note that character formatting does not carry over from cell to cell, so that you must enter a new `<FONT>` tag for every cell, if you want something other than the default font, size or color (or use a style designation in each Cell definition tag). Also, be sure and close any `<FONT>` tags you use—not doing so may crash Navigator. In fact, Browsers are in general quite picky about Table tags. Be sure to close every Cell, Row or Table, or else the table will not display properly, will not load, or the browser may crash.

`<TABLE>` is modified by:

`WIDTH = "n"` where n is a number in pixels, or percentage (n%) of available space. Percentages are especially useful, because the table will scale to fit the window or space available. `BORDER = "n"` where n is the width in pixels of the border you want to appear around each cell of the table (“0” yields a border-less table). `BORDERCOLOR = "#XXXXXX"` determines the color of the border around each cell. `CELLPADDING = "n"` determines the amount of space placed between the walls of each cell and its contents. `CELLSPACING = "n"` determines the amount of space between each cell (similar, but not identical to, the border width). `BACKGROUND = "XXXXXX"` where the Xs are the location and name of an image to be placed in the background of each cell (note, the image will begin anew in the top left corner of each cell.). `BGCOLOR = "#XXXXXX"` determines the background color for each cell of the table. `ALIGN = "XXXXXX"` determines the internal formatting of each cell of the table. Options are `CENTER`, `LEFT` or `RIGHT`

`<TR>` begins a new row of the table. `<TR>` is modified by a subset of the modifiers for `<TD>`, notably, `BGCOLOR`, `ALIGN`, `VALIGN`, and `CLASS` (see below).

`<TD>` begins a new cell of the table. `<TD>` is modified by:

`ALIGN = "XXXXXX"` options the same as TABLE align. This will override any alignment setting from the Table tag. `VALIGN = "XXXXXX"` sets the Vertical alignment of the cell contents. Options are `TOP`, `MIDDLE` or `BOTTOM`. `WIDTH = "n"` again either a number of pixels or percentage for the width of the cell. `HEIGHT = "n"` a number of pixels (not a percentage) for the height of a cell. `COLSPAN = "n"` tells the browser that this particular cell should be more than one column wide (determined by the number that replaces the n). `ROWSPAN = 'n'` tells the browser that this cell should be more than one row high. `BGCOLOR = "#XXXXXX"` sets the background color of the cell. `BACKGROUND = "XXXXXX"` locates and identifies an image file to be used as the background for this cell. `CLASS ID = "XXXXXX"` identifies a Style to be used on the contents of this cell—very useful.

`<!--XXXXXX-->`

A comment. This is a place to identify sections for your convenience or leave notes for others who may be working on the page. This can appear anywhere and will not be displayed by a browser. It is often necessary for CSS or JavaScript code to be “commented out” so that browsers that cannot interpret the code do not crash in the effort to do so.