

## HTML Overview

HTML (Hypertext Markup Language) is not a traditional programming language; rather it identifies the elements of a page that a browser displays on a computer screen. It is used to format information rather than perform actions in a designated order.

### General

Some general information you need to remember about HTML are:

- There are two types of tags: *containers* such as `<html>` and `</html>` and *separators* such as `<br>`. Container tags are switched on and switched off (as indicated by the `/`). Remember to close all container tags, as some browsers will accept tags that are not closed and some won't.
- Always include comments explaining your code. Your code must be clearly comprehensible to other coders for ease of maintenance. Therefore, it must include comments. HTML comments are enclosed within a `<!--` and a `-->`.
- HTML tags are not case sensitive: `<B>` means the same as `<b>`. Most tutorials use uppercase HTML tags in their examples but because we want to prepare ourselves for the next generations of HTML, we would be using lowercase tags throughout this school. The World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags.
- Excess white space (including spaces, new lines, tabs, carriage returns, etc.) in HTML is generally ignored. Therefore the use of extra white space in your code is encouraged in order to improve readability, without impacting on the result displayed.
- Use `<meta>` tags. The `<meta>` tag, contained within the `<head></head>` tag is not necessary, however it is good programming practice to use it. `meta` tags can be very useful for Web developers. They can be used to identify the creator of the page, what HTML specifications the page follows, and the keywords and description of the page.
  - Make sure to fill the *keyword* and *description* attributes. These allow the search engines to easily index your page using the keywords you specify and a description of the

site that you write. Use the keywords attribute to tell the search engines which keywords to use, like this:

```
<meta name="keywords" content="life, universe,
mankind, plants, relationships, the meaning of
life, science">
```

## HTML is Interpreted not Executed

If you make an error using traditional programming languages, it will either produce a fatal error or run the program incorrectly. However, errors using HTML are not fatal. The browser will make an effort to display the page, although it will probably not be displayed as intended; the browser will simply ignore the code it does not understand.

---

## Standard Tags

HTML documents' standard tags are as follows:

- `<html></html>`
- `<head></head>`
- `<title></title>`
- `<body></body>`

Standard tags and container tags often have an attribute assigned to them. This is a keyword that is placed after the tag name and before the closing right bracket of the tag. An attribute usually has a value associated with it that is specified after an equal (=) sign. If the value that is associated with the attribute is a sentence or a few words, then the value must be enclosed in double quotes ". However, it is good coding practice to enclose all attribute values in ". Further, a tag may have several attributes.

E.g. `<font face="Arial" size="6">`

Be aware of issues that can arise regarding user friendliness of the Graphical User Interface (GUI), i.e. the page you display needs to serve the purpose required. Examples include:

- The use of underlining text in an HTML document is not encouraged as hyperlinks are generally underlined. Therefore, the user may become confused if underlined text is not a link to elsewhere on the site or the web.
- Blinking text can be distracting for a user.
- Borders around an image are often used to indicate that it is a link. It is therefore discouraged to use a border unless the image is a

link. In order to make an image stand out the use of white space is an alternative.

- Background patterns can be distracting for a user.

Note: The spelling accepted in HTML is US English. E.g. 'color' is accepted instead of 'colour' and 'centre' is spelled 'center'.

---

## Header

All details encompassed in the `<head></head>` tags will not be displayed on the web page. Rather, this tag outlines the header section. The use of the `<title></title>` tags will label the page in the browser title bar. However, for the title to appear on the page it must be included within the `<body></body>` tags.

- The `head` section is also a technical feature that allows programmers to request just that section of a HTML page before deciding if they want to access the whole thing or not. It is a performance feature allowing a search to be performed more efficiently. As such, administrative details are contained in this section.

---

## Images

Be aware that when entering new dimensions for an image you need to keep the original height-width ratio in order to avoid distorting the image. E.g. If your image is 200 pixels by 100 pixels, the new dimension should remain 2:1.

Always include a description of every image using the `alt` attribute. `alt` is provided for browsers that have images turned off, or those that are unable to view images. The value of the `alt` image will appear on screen in place of the image.

Also, be aware that very large images will increase the time it takes for a web page to load. Images can be compressed without losing too much or any visible quality via photo-editing software such as Adobe Photoshop.

---

## Frames

Frames are optional. They divide Web pages into multiple, scrollable regions. This allows information to be presented in a flexible and useful fashion. Each region, or frame, has several features:

- It can contain a unique HTML page/document (individual URL), so it can load information independent of the other frames on the page;
- It can be given a `name`, allowing it to be targeted by other URLs, and;
- It can resize dynamically if the user changes the window's size (resizing can also be disabled, ensuring a constant frame size.)

When implementing columns in a frame, the use of `*` as the last width specifier will indicate to the browser to use up all the remaining horizontal width.

E.g. `<frameset cols = "150, *">`

Note: This feature is also useful in other contexts, such as tables.

It is good coding practice to use the `<noframes>` tag. This tag allows you to explain to the user that their browser does not support frames.

E.g.

```
<noframes>
  <body>
    <p>This page uses frames, but your browser doesn't
      support them</p>
  </body>
</noframes>
```

---

## Links

To link to other documents, you need to specify a URL. These come in two variations:

- Absolute: This is an entire pathname for documents not necessarily stored in the same directory structure.
- Relative: This is the path of the files stored in the local directory.

Note: Use of relative path names allows you to move all the files to a new server together without having to change the hyperlinks.

---

## Forms

World Wide Web “forms” are commonly used as the computer equivalent of paper forms. You will generally find a button or link at the end of every form usually labeled “Submit”. When you push this button, two things are sent to the server:

- The data typed into the form

- An `action`, which basically tells the server the name of the program that knows how to process that form's data

The server simply invokes that program, passes the form's data to it, and arranges for the output of that program to be sent back to the browser.

When using input attributes, always specify a size so the user has enough space to input the data.

*Note: Do not forget to close the `</form>` tag, as overlooking this will impact on further forms on the page.*

---

## Tables

Within the form, tables can be used to organize information.

There are three basic steps to defining a table.

1. Begin by defining the table and its attributes i.e. the appearance of the table, spacing, etc.
2. Follow this defining the row and its attributes e.g. the properties such as alignment.
3. Finally, define the columns and its attributes.

By defining the border attribute = "0" (used with the opening `<table>` tag), the border will be invisible when it is displayed on the browser.

Do not forget to close any of the area formatting tags inside the table. Failure to do so will distort the table.

Be aware that the `colspan` and `rowspan` attributes cover the areas of other cells. As such, the cells need to be merged. Otherwise, the formatting of the table will be distorted.

---

## Lists

There are three types of lists:

- Ordered
- Unordered
- Definition

It is important to indent each level of a nested list in order to ensure your code is easier to understand and debug.

E.g.  
`<ol>`  
    `<li>First element of List A</li>`

```
<li>Second element of List A</li>
  <!--This starts an unordered list-->
  <ul>
    <li>First element of List B</li>
    <li>Second element of List B</li>
  </ul>
</ol>
```

Note:

Indenting is relevant throughout HTML; it should also be implemented with tables etc.

It should be noted that browsers install a line of white space after every closed list.

---

## Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write: `&lt;` or `&#60;`

The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Note: The character entities are case sensitive.

## Non-breaking Space

The most common character entity in HTML is the non-breaking space.

Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the `&nbsp;` character entity.

### The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space	<code>&amp;nbsp;</code>	<code>&amp;#160;</code>
<	less than	<code>&amp;lt;</code>	<code>&amp;#60;</code>
>	greater than	<code>&amp;gt;</code>	<code>&amp;#62;</code>
&	ampersand	<code>&amp;amp;</code>	<code>&amp;#38;</code>
"	quotation mark	<code>&amp;quot;</code>	<code>&amp;#34;</code>
'	apostrophe	<code>&amp;apos;</code>	<code>&amp;#39;</code>

### Some Other Commonly Used Character Entities:

Result	Description	Entity Name	Entity Number
¢	cent	<code>&amp;cent;</code>	<code>&amp;#162;</code>
£	pound	<code>&amp;pound;</code>	<code>&amp;#163;</code>
¥	yen	<code>&amp;yen;</code>	<code>&amp;#165;</code>
§	section	<code>&amp;sect;</code>	<code>&amp;#167;</code>
©	copyright	<code>&amp;copy;</code>	<code>&amp;#169;</code>
®	registered trademark	<code>&amp;reg;</code>	<code>&amp;#174;</code>
×	multiplication	<code>&amp;times;</code>	<code>&amp;#215;</code>
÷	division	<code>&amp;divide;</code>	<code>&amp;#247;</code>

To see a full list of HTML character entities see [HTML Entities Reference](#).

---

## HTML Glossary

---

### <!--...-->

The comment tag is used to insert a comment in the source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

You can also store program-specific information inside comments. In this case they will not be visible for the user, but they are still available to the program. A good practice is to comment the text inside the script and style elements to prevent older browsers, that do not support scripting or styles, from showing it as plain text.. For example:

```
<html>
  <body>
    <!--This comment will not be displayed-->
    <p>This is a regular paragraph</p>
  </body>
</html>
```

---

### <!doctype...>

The `<!doctype>` declaration is the very first thing in your document, before the `<html>` tag. This tag tells the browser which HTML or XHTML specification the document uses.

```
<!doctype html public "-//w3c//dtd html 4.01
transitional//en">
```

However, it is worth noting that if the document does not contain this type declaration, a browser should infer it. The tag above identifies the document as conforming to the HTML 4.0 specifications set by the W3C. The transitional notation means that some formatting tags are embedded in the HTML instead of specified in a cascading style sheet (which is the strict standard). We use this in the school because presentation attributes and elements that W3C expects to move to a style sheet are followed as much as possible but we may use HTML's presentational features on occasion.

## <a...>...</a>

Anchor elements are defined by the `<a>` element. The `<a>` element accepts several attributes, but either the `name` or `href` attribute is required.

### href

If the `href` (Abbreviated from Hypertext REFerence) attribute is present, the text between the opening and closing anchor elements becomes a hypertext link. 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. Typically, hyperlinks specified using this element would be rendered in underlined blue text, unless the `link` attribute of the `<body>` element has been specified.

See

```
<a href="http://www.w3schools.com/html/default.asp/"> HTML School</A> for more information about the HTML Reference.
```

In this example, selecting the text "HTML School" takes the reader to a document located at

```
http://www.w3schools.com/html/default.asp/.
```

With the `href` attribute, the form `href="#identifier"` can refer to another anchor in the same document, or to a fragment of another document, that has been specified using the `name` attribute.

```
The section<a href="document.html#anchor">Anchors</A> provides details about links.
```

Selecting the link takes the reader to another anchor (that is, `<a name="anchor">` in a different document (`document.html`). If the anchor is in another document, the `href` attribute may be relative to the document's address or the specified base address, or can be a fully qualified URL.

### 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, which may be any arbitrary string but must be unique within the HTML document.

```
<a name="anchor">Anchors</a> gives information about...
```

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

```
<a href="document.html#anchor">
```

## target

With the advent of Frame page formatting, browser windows can now have names associated with them. Links in any window can refer to another window by name. When you click on the link, the document you asked for will appear in that named window. If the window is not already open, the browser will open and name a new window for you.

The syntax for the targeted window is:

```
<a href="download.html" target="reference">Download  
information</a>
```

This would load the document `download.html` in the frame that has been designated as having the name `reference`. If no frame has this name, then the browser will open a new browser window to display the document in. For more information, see `<frame>`.

---

## <b>...</b>

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.

---

## <basefont ...>

This attribute describes the default format of text that appears in the HTML page. Although this has been deprecated in HTML 4.01 strict standards in favor of cascading style sheets, it can still be used in embedded formats for our HTML page.

Note: A deprecated element or attribute is one that has been outdated. Deprecated elements may become obsolete in the future, but browsers should continue to support deprecated elements for backward compatibility.

### size

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.

```
<basefont size=5>
```

### face

This attribute allows changing of the face of the HTML document `<basefont>` exactly as it works for `<font face= ...>`.

### color

This allows the `<basefont>` color for the HTML document to be set (as such it is similar to the text attribute of the `<body>` element). Colors can either be set by using one of the reserved color names or as a hex `rrggbb` triplet value.

---

## `<body...>...</body>`

The body of an HTML document, as its name suggests, contains all the text and images that make up the page, together with all the HTML elements that provide the control and formatting of the page. The format is:

```
<body>
  The rest of the document included here
</body>
```

The `<body>...</body>` tags should be directly enclosed by the `<html>...</html>` tags.

The `<body>` and `</body>` tags themselves do not directly affect the look of the document when rendered, but they are required in order for the document to conform to the specification standard. Various attributes of the opening `<body>` tag can be used to set up various page-formatting settings.

The ability to specify background images and colors for HTML documents was first implemented by Netscape and has since been implemented by most other browsers. It should be noted that the following elements may not be supported by every browser.

## background

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 most browsers, this background image is used to tile the full background of the document-viewing area. Consider the following code:

```
<body background="imagename.gif">  
  Rest of the document goes here  
</body>
```

It would cause whatever text, images, and so on that appeared in the body of the document to be placed on a background consisting of the `imagename.gif` graphics file, being tiled to cover the viewing area (like bitmaps are used for Windows wallpaper). Most browsers that support this attribute allow the use of `.gif` and `.jpg` images for document backgrounds, whereas Internet Explorer supports those, plus Windows `.bmp` files.

## bgcolor

This allows the setting of the color of the background without having to specify a separate image that requires another network access to load. The format is:

```
<body bgcolor="#rrggbb">  
  Rest of document goes here  
</body>
```

where `#rrggbb` is a hexadecimal (base 16) red-green-blue triplet used to specify the background color.

## bgproperties

In Internet Explorer, you can watermark HTML documents by fixing a background image so that it doesn't scroll as a normal background image does. To give a page with a background image a watermarked background, add `bgproperties=fixed` to the `<body>` element as follows:

```
<body background="filename.gif" bgproperties=fixed>
```

## link, vlink, and alink

These link attributes allow you to control the color of link text. `vlink` stands for visited link, and `alink` stands for active link (this sets the color that the link text will be for the time that it is clicked on). Generally, the default colors of these attributes are `link=blue (#0000ff)`, `vlink=purple (#800080)`, and `alink=red (#ff0000)`. The format for these attributes is the same as that for `bgcolor` and `text`:

```
<body link="#rrggbb" vlink="#rrggbb" alink="#rrggbb">  
  Rest of document goes here  
</body>
```

## text

The `text` attribute can be 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">  
  Rest of document goes here  
</body>
```

---

## <br>

The line break inserts new line at the given point. The amount of line space used is dependent on the particular browser, but is generally the same as it would use when wrapping a paragraph of text over multiple lines.

Note: Use the `<br>` tag to enter blank lines, not to separate paragraphs.

---

## Character Entity References

Many of the Latin-1 set of printing characters may be represented within the text of an HTML document by a character entity

Character	Numeric Entity	Hex Value	Character Entity	Description
"	&#34;	22	&quot;	Quotation mark
&	&#38;	26	&amp;	Ampersand
<	&#60;	3C	&lt;	Less than

Character	Numeric Entity	Hex Value	Character Entity	Description
>	&#62;	3E	&gt;	Greater than
©	&#169;	A9	&copy;	Copyright
®	&#174;	AE	&reg;	Registered trademark
À	&#192;	C0	&Agrave;	Capital A, grave accent
Á	&#193;	C1	&Acute;	Capital A, acute accent
Â	&#194;	C2	&Acirc;	Capital A, circumflex accent
Ã	&#195;	C3	&Atilde;	Capital A, tilde
Ä	&#196;	C4	&Auml;	Capital A, dieresis or umlaut mark
Å	&#197;	C5	&Aring;	Capital A, ring
Æ	&#198;	C6	&AElig;	Capital AE diphthong (ligature)
Ç	&#199;	C7	&Ccedil;	Capital C, cedilla
È	&#200;	C8	&Egrave;	Capital E, grave accent
É	&#201;	C9	&Eacute;	Capital E, acute accent
Ê	&#202;	CA	&Ecirc;	Capital E, circumflex accent
Ë	&#203;	CB	&Euml;	Capital E, dieresis or umlaut mark
Ì	&#204;	cc	&Igrave;	Capital I, grave accent
Í	&#205;	CD	&Iacute;	Capital I, acute accent
Î	&#206;	CE	&Icirc;	Capital I, circumflex accent
Ï	&#207;	CF	&Iuml;	Capital I, dieresis or umlaut mark
Ñ	&#209;	D1	&Ntilde;	Capital N, tilde
Ò	&#210;	D2	&Ograve;	Capital O, grave accent
Ó	&#211;	D3	&Oacute;	Capital O, acute accent
Ô	&#212;	D4	&Ocirc;	Capital O, circumflex accent
Õ	&#213;	D5	&Otilde;	Capital O, tilde
Ö	&#214;	D6	&Ouml;	Capital O, dieresis or umlaut mark
Ø	&#216;	D8	&Oslash;	Capital O, slash
Ù	&#217;	D9	&Ugrave;	Capital U, grave accent
Ú	&#218;	DA	&Uacute;	Capital U, acute accent

Character	Numeric Entity	Hex Value	Character Entity	Description
Û	&#219;	DB	&Ucirc;	Capital U, circumflex accent
Ü	&#220;	DC	&Uuml;	Capital U, dieresis or umlaut mark
à	&#224;	E0	&agrave;	Small a, grave accent
á	&#225;	E1	&acute;	Small a, acute accent
â	&#226;	E2	&acirc;	Small a, circumflex accent
ã	&#227;	E3	&atilde;	Small a, tilde
ä	&#228;	E4	&aauml;	Small a, dieresis or umlaut mark
å	&#229;	E5	&aring;	Small a, ring
æ	&#230;	E6	&aelig;	Small ae diphthong (ligature)
ç	&#231;	E7	&ccedil;	Small c, cedilla
è	&#232;	E8	&egrave;	Small e, grave accent
é	&#233;	E9	&eacute;	Small e, acute accent
ê	&#234;	EA	&ecirc;	Small e, circumflex accent
ë	&#235;	EB	&euml;	Small e, dieresis or umlaut mark
ì	&#236;	EC	&igrave;	Small i, grave accent
í	&#237;	ED	&iacute;	Small i, acute accent
î	&#238;	EE	&icirc;	Small i, circumflex accent
ï	&#239;	EF	&iuml;	Small i, dieresis or umlaut mark
ñ	&#241;	f1	&ntilde;	Small n, tilde
ò	&#242;	f2	&ograve;	Small o, grave accent
ó	&#243;	f3	&oacute;	Small o, acute accent
ô	&#244;	f4	&ocirc;	Small o, circumflex accent
õ	&#245;	f5	&otilde;	Small o, tilde
ö	&#246;	f6	&ouml;	Small o, dieresis or umlaut mark
ø	&#248;	f8	&oslash;	Small o, slash
ù	&#249;	f9	&ugrave;	Small u, grave accent
ú	&#250;	FA	&uacute;	Small u, acute accent
û	&#251;	FB	&ucirc;	Small u, circumflex accent

---

Character	Numeric Entity	Hex Value	Character Entity	Description
ü	&#252;	FC	&uuml;	Small u, dieresis or umlaut mark
ÿ	&#255;	FF	&yuml;	Small y, dieresis or umlaut mark

---

## <center>

All lines of text between the begin and end of the `<center>` element are centered between the current left and right margins. This element was introduced by the Netscape authors because they claimed that using `<p align=center>` "broke" existing browsers when the `<p>` element was used as a container (that is, with a closing `</p>` element).

The element is used as shown below and any block of text (including any other HTML elements) can be enclosed between the centering elements.

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

Although this has been deprecated in HTML 4.01 strict standards in favor of cascading style sheets, it can still be used in embedded formats for our HTML page.

---

## <div>...</div>

The `<div>` tag defines a division/section in a document. Browsers usually place a line break before and after the div element. For example:

```
This is some text  
<div style="color:#FF0000;">  
  <h4>This is a header in a div section</h4>  
  <p>This is a paragraph in a div section</p>  
</div>
```

Note: Use the `<div>` tag to group block-elements to format them with styles.

---

## <font ...>

The `<font>` tag specifies the font face, font size, and font color of text. Although this has been deprecated in HTML 4.01 strict standards in favor

of cascading style sheets, it can still be used in embedded formats for our HTML page.

### size

The 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 to the document `<basefont>`.

```
<font size=4>changes the font size to 4</font>  
<font size=+2>changes the font size to BASEFONT SIZE ... +  
2</font>
```

### color = #rrggbb or color = color

The color attribute sets the color for the text that will appear on the screen. `#rrggbb` is a hexadecimal color denoting an RGB color value. Alternately, the color can be set to one of the available predefined colors. These color names can be used for the `bgcolor`, `text`, `link`, `alink`, and `vlink` attributes of the `<body>` tag as well.

```
<font color="#ff0000">this text is red.</font>
```

or

```
<font color="red">this text is also red.</font>
```

### face=name [,name] [,name]

The `face` attribute sets the typeface that will be used to display the text on the screen. The typeface displayed must already be installed on the user's computer. Substitute typefaces can be specified in case the chosen type face is not installed on the user's computer. If no exact font match can be found, the text will be displayed in the default type that the browser uses for displaying normal text.

```
<font face="Courier New, Comic Sans MS">  
This text will be displayed in either Courier New, or Comic  
Sans MS, depending on which fonts are installed on the  
browsers system. It will use the default 'normal' font if  
neither is installed.  
</font>
```

---

`<head>...</head>`

The `<head>` element of an HTML document is used to provide information about the document. It requires the `<title>` element between `<head>` and `</head>` tags:

```
<head>
  <title>Introduction to HTML</title>
</head>
```

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

The following elements are related to the `<head>` element. Although they don't directly affect the look of the document when rendered, you can use them to provide important information to the browser.

To do so, you employ the following elements, all of which should be included within the `<html>...</html>` tags.

<code>&lt;base&gt;</code>	Allows the base address of HTML document to be specified
<code>&lt;isindex&gt;</code>	Allows keyword searching of the document
<code>&lt;link&gt;</code>	Indicates relationships between documents
<code>&lt;meta&gt;</code>	Specifies document information usable by server/clients
<code>&lt;nextid&gt;</code>	Creates unique document identifiers
<code>&lt;style&gt;</code>	Specifies styles within the document when used by browsers that support use of style sheets
<code>&lt;title&gt;</code>	Specifies the title of the document

---

## `<hr>`

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

```
<hr>
<font face="Arial">May, 05, 2005, One Fine Day</font>
```

The `<hr>` element specifies that a horizontal rule of some sort (the default being a shaded engraved line) be drawn across the page. The following optional attributes have been deprecated in HTML 4.01 strict

standards in favor of cascading style sheets, it can still be used in embedded formats for our HTML page:

`<hr align=left | right | center>`

Because horizontal rules do not have to be the width of the page, it is necessary to allow the alignment of the rule to be specified. Using the above values, rules can be set to display centered, left-aligned, or right-aligned.

`<hr color=name | #rrggb>`

Internet Explorer allows the specifying of the hard rule color. Accepted values are any of the Internet Explorer supported color names, or any acceptable `rrggb` hex triplet.

`<hr noshade>`

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.

`<hr size=number>`

The `size` attribute lets the author give an indication of how thick they wish the horizontal rule to be. The number value specifies how thick the rule will be, in pixels.

`<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 the browser display window.

---

`<html>...</html>`

The `<html>` element identifies the document as containing HTML elements. It serves to surround all of the remaining text, including all other elements. Browsers use the presence of this element at the start of an HTML document to ensure that the document is actually HTML. The document should be constructed as follows:

```
<!doctype html public "-//w3c//dtd html 4.01
transitional//en">
<html>
```

The rest of the document should be placed here.

`</html>`

---

## `<hx>...</hx>`

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 first level heading heading</h1>
Here is some normal paragraph text
<h2>This is a second level heading</h2>
Here is some more normal paragraph text.
```

The rendering of headings is determined by the browser, but typical renderings are:

<code>&lt;h1&gt;...&lt;/h1&gt;</code>	Bold, very large font, centered. One or two blank lines above and below.
<code>&lt;h2&gt;...&lt;/h2&gt;</code>	Bold, large font, flush-left. One or two blank lines above and below.
<code>&lt;h3&gt;...&lt;/h3&gt;</code>	Italic, large font, slightly indented from the left margin. One or two blank lines above and below.
<code>&lt;h4&gt;...&lt;/h4&gt;</code>	Bold, normal font, indented more than <code>H3</code> . One blank line above and below.
<code>&lt;h5&gt;...&lt;/h5&gt;</code>	Italic, normal font, indented as <code>H4</code> . One blank line above.
<code>&lt;h6&gt;...&lt;/h6&gt;</code>	Bold, indented same as normal text, more than <code>H5</code> . One blank line above.

Although heading levels can be skipped (for example, from `h1` to `h3`), this practice is not recommended as skipping heading levels may produce unpredictable results when generating other representations from HTML.

---

## <i>...</i>

The italics element specifies that the text should be rendered in italics, where available. Otherwise, alternative mapping is allowed. E.g.

The instructions `<i>must be read</i>` before continuing.

would be rendered as:

The instructions *must be read* before continuing.

---

## <img...>

The `<img>` element is probably the second most important markup element (behind the Anchor element) as it handles all embedded graphical content in HTML documents.

The Image element is used to incorporate inline graphics (typically icons or small graphics) into an HTML document. This element cannot be used for embedding other HTML text. Browsers that cannot render inline images ignore the Image element unless it contains the `alt` attribute.

The Image element attributes are listed below. Although the "align", "border", "hspace", and "vspace" attributes of the image element were deprecated in HTML 4.01. strict standards in favor of cascading style sheets, it can still be used in embedded formats for our HTML page

### align

The `align` attribute accepts the values `left`, `right`, `top`, `texttop`, `middle`, `absmiddle`, `baseline`, `bottom`, and `absbottom`, which specifies the alignment of the image and that of the following line of text.

These attribute values to the `align` option require some explanation. First, the values "left" and "right". Images with those alignments are a *floating* image type.

- `align=left` will align the image on the left-hand edge of the browser display window and subsequent text will wrap around the right-hand side of that image.
- `align=right` will align the image on the right-hand edge of the browser display window and subsequent text will wrap around the left-hand side of that image.

- `align=top` allows any text following the image to align itself with the top of the tallest item in the line. (That is, the top of the image.)
- `align=texttop` allows any text following the image to align 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.

## alt

This attribute allows the setting of text as an alternative to the graphic for rendering in non-graphical environments, or when the user has deactivated the auto-loading of images. Alternate text should be provided by the browser whenever the graphic is not rendered.

```
 Be sure to read  
these instructions.
```

## border=value

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

It is useful if the image is to be a hyperlink, in that the `border` can be set to 0 to avoid the display of the standard blue hypertext link border.

## lowsrc

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

```

```

Browsers that do not recognize the `lowsrc` attribute cleanly ignore it and simply load the image specified by the `src` attribute.

Browsers that support this attribute, however, will load the image called `lowquality.gif` on their first layout pass through the document. When the rest of the document has been completely loaded and formatted on the page, the browser will then redraw the page and load the image specified by the standard `src` attribute. This allows the author to specify a low-resolution (or smaller file size version of the main image-perhaps a grayscale version) image to be displayed initially while the document is loading, which is later replaced by the higher quality version.

Any graphic file format that the browser supports can be used interchangeably within the `lowsrc` and `src` attributes. You can also specify width and height values in the `img` element, and both the high-resolution and low-resolution versions of the image will be appropriately scaled to match. However, if no width and height values have been set, the values used for the `lowsrc` image (that is, the dimensions of that image) will be used to re-scale the `src` image. This is to minimize page format disruption that would be caused by the browser trying to load two different sized images into the same page space.

```
Mosaic, from the National Center for Supercomputing Applications represents the original graphical browser which Netscape development was based on.
```

```
<br clear="all">
<hr>
```

```
Netscape, from Netscape Communications, after initial development from Mosaic, stormed away and became more or less the de facto Web browser.
```

```
<br clear="all">
<hr>
```

```
 Internet Explorer, from Microsoft, exhibits Microsoft's serious intentions to enter the Web browser market and compete head-to-head with Netscape.
```

```
<br clear="all">
<hr>
```

## src

The value of the `src` attribute is the URL of the image to be displayed. Its syntax is the same as that of the `href` attribute of the `<a>` element. `src` is the only mandatory attribute of the `<img>` element. Image elements are allowed within anchors.

```
<img src = "warning.gif">Be sure to read these instructions.
```

The `src` attribute can accept fully qualified, or partial, relative URL's, or even just image names (providing the image is located in the same directory as the HTML document).

## vspace=value hspace=value

For the *floating* images (that is, those displayed with an `align=left|right` attribute) it is likely that the author does not want the text wrapped around the image to be pressed up against 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. Value should be a pixel value.

## width=value height=value

The `width` and `height` attributes allow the browser to determine the text layout surrounding images before the entire image has been downloaded, which can significantly speed up display of the document text. If the author specifies these, the viewer of the document will not have to wait for the image to be loaded over the network and its size to be calculated. Internet Explorer uses image placement mechanisms, so that if the display of in-line images has been turned off, the space that the images would occupy in the page is marked as if the image were there (with any `alt` text being displayed in the place holder). This allows authors to be sure that the text layout on the page will be as desired, even if the user is not displaying the images.

---

## <link...>

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 would typically be used to provide pointers to related indexes, or glossaries. Links can also be used to indicate a static

tree structure in which the document was authored by pointing to a parent, next, and previous document, for example.

The `<link>` element represents one of the primary style sheet inclusion mechanism elements. It can be used to specify the location of the style sheet that is to be used for the document. For example:

```
<html>
  <head>
    <title>This document uses a style sheet</title>
    <link rel="stylesheet" type="text/css"
          href=http://www.stylesheets.com/sheets/formal.css
          title="formal">
  </head>
  <body>
    Rest of the document goes here
  </body>
</html>
```

In the preceding HTML fragment, the `<link>` element points to the file "formal.css" at the given URL. It tells the browser that:

- the file addressed is a style sheet, by explicitly giving the "text/css" MIME type;
- the file's RELationship to the HTML document is that it is a "stylesheet";
- the stylesheet's title is "formal."

---

## `<meta...>`

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 specialized document meta-information.

### **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 whether or not the DTD includes anything about it.

## 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.

Some meta tag examples are:

```
<meta http-equiv="expires" content="Sat, 06 Jan 1990
00:00:01 GMT">
<meta http-equiv="from" content="nick@htmlib.com">
<meta http-equiv="reply-to"
content="stephen@htmlib.com"
<meta http-equiv="keywords"
content="training, sample, HTML">
```

The `<meta>` element is particularly useful for constructing Dynamic documents via the Client Pull mechanism. This uses the following syntax:

```
<meta http-equiv="refresh" content="x">
```

which causes the document to be re-loaded in `x` seconds. This can be useful to provide automatic redirection of browsers. For instance, if the element was

```
<meta http-equiv="refresh" content="2;
url=http://some.site.com/otherfile.html">
```

then the `refresh` directive would cause the file at `http://some.site.com/otherfile.html` to be loaded after 2 seconds. Although this generally works if the URL specified is partial, you should use a fully qualified URL to ensure its proper functioning.

---

## `<p>...</p>`

The paragraph element indicates a paragraph of text. No specification has ever attempted to define exactly the indentation of paragraph blocks and this may be a function of other elements, style sheets, and so on.

---

Typically, paragraphs should be surrounded by a vertical space of between one and one and a half lines. With some browsers, the first line in a paragraph may be indented.

```
<h1>The Paragraph element</h1>
```

```
<p>The paragraph element is used to denote paragraph blocks.</p>
```

```
<P>This would be the second paragraph.</p>
```

---

## <sub>...</sub>

The `<sub>` element specifies that the enclosed text should be displayed as a subscript, and, if practical, using a smaller font (compared with normal text).

This is the main text, with `<sub>this bit</sub>` being subscript.

This is the main text, with `this bit` being subscript.

The exact appearance of the subscript text will change depending on any `<font size=...>` and `<basefont size=...>` settings, if specified.

---

## <sup>...</sup>

The `<sup>` element specifies that the enclosed text should be displayed as a superscript, and, if practical, using a smaller font (compared with normal text).

This is the main text, with `<sup>this bit</sup>` being superscript.

This is the main text, with `this bit` being superscript.

The exact appearance of the superscript text will change depending on any `<font size=...>` and `<basefont size=...>` settings, if specified.

---

## <title>...</title>

Every HTML document must have a `<title>` element. As its name suggests, it is used to specify the title of the document in question. Normally, browsers will render the text contained within the `<title>...</title>` elements in the title bar of the browser window.

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

This is the only element that is required within the `<head>` element.

```
<head>  
  <title>Welcome to the HTML Reference</title>  
</head>
```

---

## `<u>...</u>`

The `<u>...</u>` elements state that the enclosed text should be rendered, if practical, underlined.

```
The <u>main point</u> of the exercise...
```

would be rendered as:

The main point of the exercise...

---

## Forms

Forms provide for the inclusion of objects like text boxes, choice lists, and so on and have proved invaluable for recent HTML applications, particularly search engines, database query entries, and the like.

It should be noted that while these HTML elements can be used to easily define the presentation of the form to the user, the real value behind any form is in what it does with the information that is entered. For a form to do anything more than send a straight text dump of the form data (including control characters) to an e-mail address, the form data will need to be passed to some kind of client-side or server-based program for processing.

The following elements are used to create forms:

<code>&lt;form&gt;...&lt;/form&gt;</code>	A form within a document
<code>&lt;input ...&gt;...&lt;/input&gt;</code>	One input field
<code>&lt;option&gt;</code>	One option within a Select element
<code>&lt;select&gt;...&lt;/select&gt;</code>	A selection from a finite set of options
<code>&lt;textarea ...&gt;...&lt;/textarea&gt;</code>	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.

A very simple form for eliciting user response would be:

```

<h1 align="center">Comment Form</h1>
<form method="post"
action="http://www.htmlib.com/formscript.cgi">
  <center>
    Your name: <input name="name" size="20">
    Your e-mail address: <input name="email" size="20">

    <p>I think the html reference is:
    <select name="choice">
      <option>outstanding
      <option>very good
      <option>good
      <option>average
      <option>below average
      <option>awful
    </select>

    <p>If you have any further comments, please enter them
    here:<br>

    <textarea name="comments" rows="10" cols="40"
      wrap="virtual">
    </textarea>

    <p><input type="submit"> <input type="reset">

  </center>
</form>

```

Different platforms will have different native systems for navigating within the input fields of a form. (For example, Windows users can use the Tab key to move from one field to the next through the order that the fields appear within the form.) Different browsers may also display different text on any buttons included in the form.

---

## <form>...</form>

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 not be nested, i.e. a form can't contain another form.

```
<form action="form_action.asp" method="get|post"
enctype="mime type">
```

The `action` attribute is a URL specifying the location to which the contents of the form data fields are submitted to elicit a response. This could be simply a direction to an e-mail address, but generally, would be used to point towards some kind of server-based script/application that handles the forwarding of form data. If the `action` attribute is missing, the URL of the document itself is assumed.

Generally, the `method` attribute specifies a method of accessing the URL specified in the `action` attribute. Generally, the method will be either `get` or `post`. The `get` method is ideal for form submission where the use of the form data does not require external processing. For example, with database searches, there is no lasting effect caused by the query of the form (that is, the query runs its search through the database and reports the results). However, when the form is used to provide information, for example, that updates a database, then the `post` method should be used, with the `action` attribute pointing to a script that executes the form data processing.

---

## <input>

The `<input>` element represents a field whose contents may be edited or activated by the user. Attributes of the `<input>` element are listed below:

## align

To be used with the `type=image` setting, this attribute specifies the alignment of the image. It takes the same values as the `align` in the `<img>` element.

## checked

To be used with a `type=checkbox` or `type=radio` setting, this indicates that the checkbox or radio button is selected.

## maxlength

To be used with `type=text` setting, this 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

This attribute represents the name that will be used for the data 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

To be used with the `type=image`, this attribute represents a URL specifying the desired image.

## type

Defines the type of data the field accepts. Defaults to free text. Several types of fields can be defined with the type attribute: `button`, `checkbox`, `image`, `file`, `hidden`, `password`, `radio`, `reset`, `submit`, and `text`.

## button

This can be used to embed buttons directly into HTML documents that add functionality when used in conjunction with a programming script. The `name` attribute is used to give the button a unique name, which can be used to set its function in the script. The `value` attribute specifies the text that is displayed on the button in the document.

## checkbox

Used for simple Boolean attributes (where a field will be chosen, or not) 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." This field type requires the `name` and `value` attributes; `checked` is an optional attribute.

## file

The `file` option to the `type` attribute of the `input` element allows an `accept` attribute for the `input` element. This allows the inclusion of files with form information, which could prove invaluable, for example, for companies providing technical support, or service providers, requesting data files.

## hidden

With this input type, 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. The `name` attribute is required. The image itself is specified by the `src` attribute, exactly as for the Image element.

## password

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

## radio

`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` and `name` attribute. `checked` is an optional attribute and can be used to specify which options are selected for initial form display.

## reset

`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

`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 browser-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

`text` is used for single-line text-entry fields. It should be used in conjunction with the `size` and `maxlength` attributes to set the maximum amount of text that can be entered. For textual input that requires multiple lines, use the `<textarea>` element for text fields which can accept multiple lines. Explicit `value` and `name` attributes are also required.

## value

When used with `type= ...` attributes, this attribute sets the initial displayed value of the field if it displays a textual or numerical value. If the `type= ...` attribute is one which only allows Boolean values (that is, chosen or not chosen) then this specifies the value to be returned when the field is selected.

## <select ...>...</select>

The `<select>` element allows the user to choose one of a set of alternatives described by textual labels. Every alternative is represented by the `<option>` element.

Attributes used with the `<select>` are listed in the below:

### multiple

The `multiple` attribute is needed when users are allowed to make several selections, for example, `<select multiple>`.

### name

Specifies the name that will be submitted as part of 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="choice">
  <option>outstanding
  <option>very good
  <option>good
  <option>average
  <option>below average
  <option>awful
</select>
```

---

## <option>

The `<option>` element can only occur within a `<select>` element. It represents one choice, and can take these attributes:

### 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.

---

## `<textarea...>...</textarea>`

The `textarea` element lets users enter more than one line of text.

Any text included up to the end element is used to initialize the field's value. This end element is always required even if the field is initially blank.

### rows/columns = number

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. Browsers should allow text to extend beyond these limits by scrolling as needed.

### wrap

To specify how to handle word-wrapping display in text input areas in forms use the following `wrap` options:

<code>&lt;textarea wrap=off&gt;</code>	The default setting. Wrapping doesn't happen. Lines are sent exactly as typed.
<code>&lt;textarea wrap=virtual&gt;</code>	The display word-wraps, but long lines are sent as one line without new-lines.
<code>&lt;textarea wrap=physical&gt;</code>	The display word-wraps, and the text is transmitted at all wrap points.

---

## Frames

Frames allow the browser display window to be subdivided into separate sections. Each section can be updated or have new documents loaded

into it separately from the remaining frame sections. As such, a frame-based layout can be especially useful for HTML applications where some information is required across a whole range of pages (such as a table of contents or title graphics)

Frames are generated by three elements: `<frameset>`, `<frame>` elements, and `<iframe>`.

## Frame Document

A frame document has a basic structure very much like a normal HTML document, except the `body` container is replaced by a `frameset` container which describes the sub-HTML documents, or Frames, that will make up the page.

```
<html>
  <head>
  </head>
  <frameset>
  </frameset>
</html>
```

No HTML that would normally be included within the `<body>` section of an HTML document should be included within the `<frameset>` ... `</frameset>` elements.

---

## `<frameset>`

This is the main container for a frame. It has two attributes `rows` and `cols`. The `<frameset>` element has a matching end element, and within the `frameset` you can only have other nested `<frameset>`, `<frame>`, or `<noframes>` elements.

### `rows="row_height_value_list"`

This takes a list of values, separated by comma marks. They can represent either absolute pixel, percentage, or relative scaling values. The total set by the values given in the `ROWS` attribute should not exceed 100% (as the total rows are extended across the whole available browser display window).

If any of the values are single numerical values, then these are considered to be absolute pixel values. It is not recommended to fix a frameset by using a complete set of pixel values, because browsers use a variety of different screen resolutions when viewing documents, and the

layout may become distorted. Percentage values can be given for this attribute. If the total percentage values given exceed 100% then all values will be scaled down by the browser so that the total is 100%. The remaining value option is to use a \* character. This tells the browser that the frame is a relative size frame and should be displayed accordingly. Numerical values can be used with the \* character, to scale the relative frame sections within the browser window.

To specify a three-part, vertical framed layout where the first section uses 20 percent of the display window, the second uses 100 pixels, and the third section uses the remaining screen, use:

```
<frameset rows="20%, 100, *>
```

To split the layout into two vertical frames, the first using a quarter of the display window, the second using three-quarters of the window, use:

```
<frameset rows="25%, 75%>
```

This would be exactly the same as using `<frameset rows="*, 3*">`.

**cols="column\_width\_list"**

The `cols` attribute takes as its value a comma separated list of values that is of the exact same syntax as the list described above for the `rows` attribute.

The `<frameset>` element can be nested. In this way, frame sections can be set up where the display window can be split into either horizontal or vertical sections, with any of these being further sub-divided by nested `<frameset>` elements.

---

## **<frame>**

This element defines a single frame in a frameset. It has eight possible attributes: `src`, `name`, `marginwidth`, `marginheight`, `scrolling`, `noresize`, `frameborder` and `framespacing`. The `<frame>` element is not a container, so it has no matching end tag.

### **src="url"**

This attribute is used to specify the HTML document that will be used as the display in the particular frame section of the frameset.

### **name="frame\_name"**

The name attribute is used to assign a name to a frame so it can be targeted by links in other documents, by using `<a href="_url_" target="frame_name">`. (These would usually be from other documents in the same frameset.) The `name` attribute is optional; by default all windows are unnamed.

Names must begin with an alphanumeric character. Several reserved names have been defined, which start with an underscore.

These are currently:

<code>_blank</code>	Always load this link into a new, unnamed window.
<code>_self</code>	Always load this link over the document that originated the link.
<code>_parent</code>	Always load this link over the parent frame. (becomes self if the frame has no parent, or is the parent frame).
<code>_top</code>	Always load this link at the top level (becomes self if the frame is the top frame).

### **marginwidth="value"**

This accepts an absolute pixel value and forces indentation from the left- and right-hand side of the frame pane according to the number of pixels. It cannot be set to a value less than 1 as this would cause the contents of the frame to be displayed right up against the left-hand margin. By default, the browser will choose its own `marginwidth` when trying to produce the best possible display.

### **marginheight="value"**

This is analogous to the `marginwidth` attribute, but it controls the top and bottom margins.

### **scrolling="yes | no | auto"**

This attribute can be used to control the appearance of any scrollbars that may appear as a result of the frame contents being too much to display in the set pane. Using "no" may be dangerous, because the HTML author cannot know the resolution/display window size of the client browser and so information may not be displayable.

### **noresize**

By default, all frames specified in a framed document can be resized by the client. Setting this flag (it requires no value) prevents the frame from being resized.

### **frameborder="yes | no"**

This allows control of the frame border display. With this attribute set to "no," the borders for the specific frame are not drawn.

---

## **<noframes>**

This element is provided for HTML authors who want to create alternative content for browsers that cannot display frames. This is especially useful if the author is making the very first document of the site a framed document. It should be noted that this element is not actually recognized by non-frame capable browsers. As with any HTML, if the browser does not recognize the element, it ignores it. Browsers that cannot display frames would ignore all the `<frameset>` and `<frame>` elements, but will display whatever is enclosed in the `<noframes> ... </noframes>` elements, which can be any HTML at all, because that is what it recognizes. On the other hand, frame-capable browsers will preferentially display what is set up by the frame elements, unless they provide any mechanism where the display of frames can be turned off, in which case they may display this alternative content.

---

## List Elements

HTML supports three types of lists, all of which may be nested. If used they should be present in the `<body>` of an HTML document.

<code>&lt;dl&gt;...&lt;/dl&gt;</code>	Definition list
<code>&lt;ol&gt;...&lt;/ol&gt;</code>	Ordered list
<code>&lt;ul&gt;...&lt;/ul&gt;</code>	Unordered list

### `<dl>...</dl>`

Definition lists are typically rendered by browsers, with the definition term `<dt>` flush left in the display window with the definition data `<dd>` rendered in a separate paragraph, indented after the definition term. Individual browsers may also render the definition data on a new line, below the definition term.

```
<dl>
  <dt>&lt;PRE&gt;
  <dd>Allows for the presentation of preformatted text.
  <dt>&lt;P&gt;
  <dd>This is used to define paragraph blocks.
</dl>
```

The layout of the definition list is at the discretion of individual browsers. However, generally, the `<dt>` column is allowed one-third of the display area. If the term contained in the `<dt>` definition exceeds this in length, 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>`.

### `<ol>...</ol>`

The Ordered List element is used to present a numbered list of items, sorted by sequence or order of importance and is typically rendered as a numbered list, but this is at the discretion of individual browsers.

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

```
<ol>
  <li>Click on the desired file to download.
  <li>In the presented dialog box, enter a name to save
  the file with.
  <li>Click 'OK' to download the file to your local drive.
</ol>
```

As mentioned above, the average ordered list counts 1, 2, 3, ... and so on. The `type` attribute allows authors to specify whether the list items should be marked with:

<code>(type=A)</code>	Capital letters. For example, A, B, C ...
<code>(type=a)</code>	Small letters. For example, a, b, c ...
<code>(type=I)</code>	Large Roman numerals. For example, I, II, III ...
<code>(type=i)</code>	Small Roman numerals. For example, i, ii, iii ...
<code>(type=1)</code>	The default numbers. For example, 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. For examples, changing the preceding example to:

```
<ol type=a start=3>
  <li>Click on the desired file to download.
  <li>In the presented dialog box, enter a name to save
  the file with.
  <li>Click 'OK' to download the file to your local drive.
</ol>
```

would present the list as using lower-case letters, starting at *c*.

To give even more flexibility to lists, the `type` attribute can be used with the `<li>` element. It takes the same values as `<ol>` and it changes the list type for that item and all subsequent items. For ordered lists, the `value` attribute is also allowed, which can be used to set the count for that list item and all subsequent items.

```
<ul>...</ul>
```

The Unordered List element is used to present a list of items which is typically separated by white space and/or marked by bullets, but this is at the discretion of individual browsers.

An unordered list must begin with the `<ul>` element, which is immediately followed by a `<li>` (list item) element: Unordered lists can be nested.

```
<ul>
  <li>First list item
  <li>Second list item
  <li>Third list item
</ul>
```

The basic bulleted list has a default progression of bullet types that changes—from a solid disc, to a circle, to a square—as you move through indented levels. The `type` attribute can be used in the `<ul>` element so that no matter what the indent level the bullet type can be specified thus:

```
type=disc
type=circle
type=square
```

To give even more flexibility to lists, the `type` attribute to the `<li>` element is also allowed. It takes the same values as `<ul>` and it changes the list type for that item and all subsequent items.

---

## Tables

The table HTML elements are

<code>&lt;table&gt;...&lt;/table&gt;</code>	The table delimiter
<code>&lt;tr ...&gt;...&lt;/tr&gt;</code>	Used to specify number of rows in a table
<code>&lt;td ...&gt;...&lt;/td&gt;</code>	Specifies table data cells
<code>&lt;th ...&gt;...&lt;/th&gt;</code>	Table header cell
<code>&lt;caption ...&gt;...&lt;/caption&gt;</code>	Specifies the table caption

---

`<table>...</table>`

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, if tables have no borders, borders will be added if the `border` attribute is specified.

The `<table>` element has the following attributes.

### **border**

This attribute can be used to both control and set the borders to be displayed for the table. If present, then a border will be drawn around all data cells. The exact thickness and display of this default border is at the discretion of individual browsers. If the attribute isn't present, then the border is not displayed, but the table is rendered in the same position as if there were a border (that is, allowing room for the border). It can also be given a value, that is, `border=<value>`, which specifies the thickness of the table border. The border value can be set to 0, which regains all the space that the browser has set aside for any borders (as in the case where no border has been set).

### **cellpadding=value**

The `cellpadding` is the amount of white space between the borders of the table cell and the actual cell data (whatever is to be displayed in the cell). It defaults to an effective value of 1. This example gives the most compact table possible:

```
<table border=0 cellspacing=0 cellpadding=0>
```

### **cellspacing=value**

The `cellspacing` is the amount of space inserted between individual table data cells. It defaults to an effective value of 2.

### **width=value\_or\_percent**

If used, this attribute can specify either the exact width of the table in pixels, or the width of the table as a percentage of the browser display window.

---

### **<caption ...>...</caption>**

This represents the caption for a table. `<caption>` elements should appear inside the `<table>` but not inside table rows or cells. Like table cells, any document body HTML can appear in a caption. Captions are,

---

by default, horizontally centered with respect to the table, and they may have their lines broken to fit within the width of the table.

---

### `<col>...</col>`

This element, which is Internet Explorer-specific, can be used to specify the text alignment for table columns. It accepts the following attributes.

**`align="center | justify | left | right"`**

This sets the text alignment within the column group. The default value is `"center"`.

**`span=value`**

This can be used to set the number of columns upon which the `ALIGN` attribute is to act.

---

### `<td ...>...</td>`

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.

`<td ...>...</td>` can accept the following attributes.

**`align="left | center | right"`**

This attribute controls whether text inside the table cell(s) is aligned to the left, right or center.

**`colspan="value"`**

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.

**`rowspan="value"`**

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.

**`valign="top | middle | bottom | baseline"`**

The `valign` attribute controls whether text inside the table cell(s) is aligned to the top, to the bottom, 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.

---

**`<th ...>...</th>`**

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 contain the following attributes:

**`align="left | center | right"`**

This attribute controls whether text inside the table cell(s) is aligned to the left, right or center of the cell.

**`colspan="value"`**

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.

**`rowspan="value"`**

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.

**`valign="top | middle | bottom | baseline"`**

The `valign` attribute controls whether text inside the table cell(s) is aligned to the top or bottom 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.

---

## <tr ...>...</tr>

This stands for table row. The number of rows in a table is exactly specified by how many <tr> elements are contained within it, regardless of cells that may attempt to use the `rowspan` attribute to span into non-specified rows.

The <tr> element can have the following attributes.

### **align="left | center | right"**

This controls whether text inside the table cell(s) is aligned to the left, right or center of the cell.

### **valign="top | middle | bottom | baseline"**

This attribute controls whether text inside the table cell(s) is aligned to the top, to the bottom, 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.