

An Introduction to HTML and XHTML

This is a brief introduction to HTML, along with some comments about the more modern formulation XHTML. The two are very similar, though XHTML has some more requirements than HTML. That is why we will usually describe how something works in HTML, and then add remarks about differences to XHTML when necessary

HTML (Hyper-Text Markup Language) and XHTML (Extensible HTML) are composed of **tags**, i.e. commands in angle-brackets (< >). HTML tags are case-insensitive, that is, it doesn't matter whether you type them in upper or lower case. However, in XHTML tags need to be lower case, so it is a good idea to get into the habit of always using lower case.

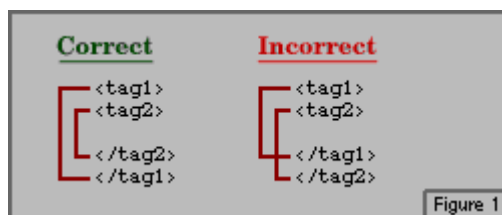
Tags typically occur in begin-end pairs. These pairs are in the form

```
<tag> ... </tag>
```

where the <tag> indicates the beginning of a tag-pair, and the </tag> indicates the end. (The three dots indicate an arbitrary amount of content between the tags.)

These pairs define **containers**. Any content within a container has the rules of that container applied to it. For example, the text within a "boldface container" would be boldfaced. Similarly, paragraphs are defined using a "paragraph container."

Thinking of tag-sets as containers will help in another way: it will help you remember that tags should always be balanced. In other words, you should keep containers nested within each other, just as you would have to do in the real world. HTML does not enforce this, but XHTML does.



Some commands do not consist of a begin and end tag, but just of a single tag. In HTML, this is just a begin tag:

```
<tag>
```

In XHTML, there is a special notation for single tags, with the "/" behind the command name:

```
<tag/>
```

Document Tags

The first tag in an HTML document is the “!DOCTYPE” tag. It looks like this (or similar):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
```

This says that the document is written in HTML, version 4.01, and that all requirements of HTML 4.01 are strictly adhered to. Don't worry if that line looks a bit complicated – just type it into your web page exactly like you see it above, and it will work for you. The DOCTYPE declaration needs to be written in upper case, just like you see it above – it is an exception to the general rule of writing tags in lower case.

If you want to use commands from previous versions of HTML, you can use the *transitional* variant of HTML 4.01 instead:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
```

In older web pages, you may note that the DOCTYPE declaration is missing. This is because it was not required in earlier versions of HTML. You can still write web pages without it and most browsers will display your page correctly. However, in order to write a valid HTML page, it is required nowadays.

After the DOCTYPE declaration come the “html” tags. These are the tags that tell a web browser where the HTML part in your document begins and ends.

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

Inside the “html” container, we have the “head” and the “body” container:

```
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

The “body” contains the actual content of your web page. The “head” contains all of the document's header information like the web document's

title and information about the document itself. This is an important point for search engines.

The container “title” is placed within the head structure. Between the title tags, you should have the title of your document. This will appear at the top of the browser's title bar, and also appears in the history list. Finally, the contents of the title container go into your bookmark file, if you create a bookmark to a page. Also, the head contains meta information about the document, most importantly the character encoding that is used. The encoding “ISO-8859-1” is used for English, French, Spanish, German and other western european languages. Here you see an example of a “head” container with a “title” element and a “meta” element denoting the encoding:

```
<head>
  <title>This is my very first HTML document</title>

  <meta http-equiv="content-type" content="text/html;
    charset=ISO-8859-1">

</head>
```

Again, you can just copy the “meta” element as you see it above into your document.

The “body” comes after the head structure. Between the body tags, you find all of the stuff that gets displayed in the browser window. All of the text, the graphics, and links, and so on - these things occur between the body tags. The strict variant of HTML 4.01 requires that any content inside the body is within a further set of tags (if you use the transitional variant of HTML 4.01, this is not necessary). For text, you can use “p” (the paragraph tag). A complete page would then look like this:

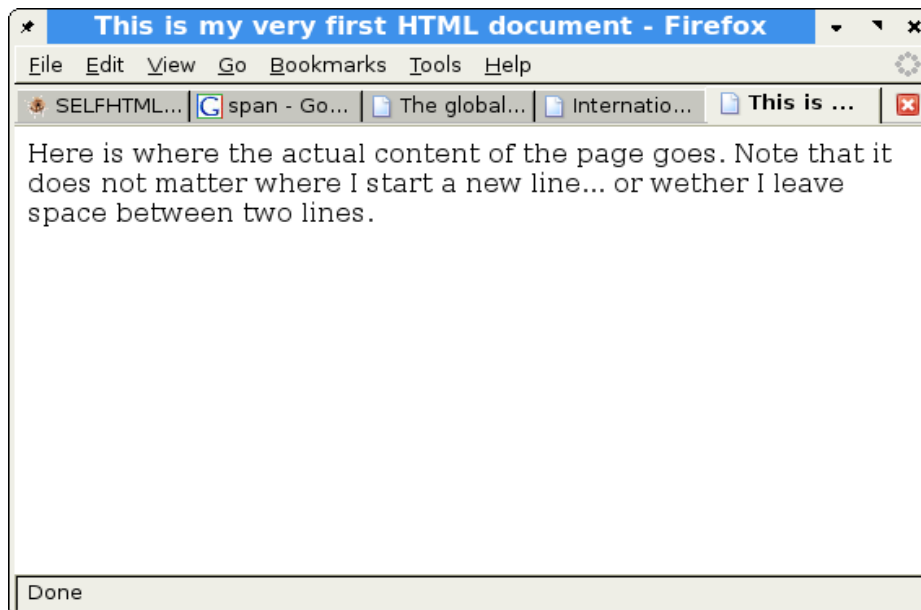
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>This is my very first HTML document</title>

    <meta http-equiv="content-type" content="text/html;
      charset=ISO-8859-1">

  </head>
  <body>
    <p> Here is where the actual content of the page goes.
    Note that it does not matter where I start a new line...

    or wether I leave space between two lines. </p>
  </body>
</html>
```

This will result in the following page being displayed in your browser:



If you want to leave yourself notes in an HTML document, but don't want those notes to show up in the browser window, you need to use the comment tag. To do that, you would do the following:

```
<!-- This is a comment. -->
```

Text Structures

As we have seen, a line break in your source file does not mean that there will be a line break on the web page displayed by your browser. So how do you structure the text on your page? For example, how do you get headings that are set apart from the text underneath? And how do you start a new paragraph that is set off by a blank line from your previous text?

Headings

Heading structures are most commonly used to set apart document or section titles.

There are six levels of headings, from heading 1 through heading 6. Heading 1 (h1) is "most important" and heading 6 (h6) is "least important." By default, browsers will display the six heading levels in the same font, with the point size decreasing as the importance of the heading decreases. Here are two examples:

```
<h1>Heading 1: The largest</h1>  
<h6>Heading 6: The smallest</h6>
```

Paragraphs and line breaks

To create an empty line between two blocks of text, you need to label those text blocks (or paragraphs) with the paragraph marker “p”. So, surround your paragraphs with `<p>` and `</p>` like we have done in our example page. Actually, in HTML you do not need the closing tag for “p”, you can create an empty line after a block by just writing `<p>`. However, it is good practice to always use the closing tag as well, and in XHTML, you have to – in XHTML it is not allowed to have an opening tag without a closing tag. Here is an example (note that you do not actually need to leave a blank line between the two paragraphs in your source code):

```
<p>This is the first paragraph. Isn't it a nice paragraph? It has lots of words in it. It's such a nice paragraph that I think it should never end.</p>
```

```
<p>This is the second paragraph. It's also a very nice paragraph but maybe not as nice as that first one was.</p>
```

If you just want to move down to the next line, without leaving a blank space, you would use a “br”, for line break. For example like this:

```
This is a line.<br>And here is the next line.
```

If using XHTML, again, you cannot have a single opening tag like `
`. It needs to be denoted as a single tag, hence you have to write `
` instead.

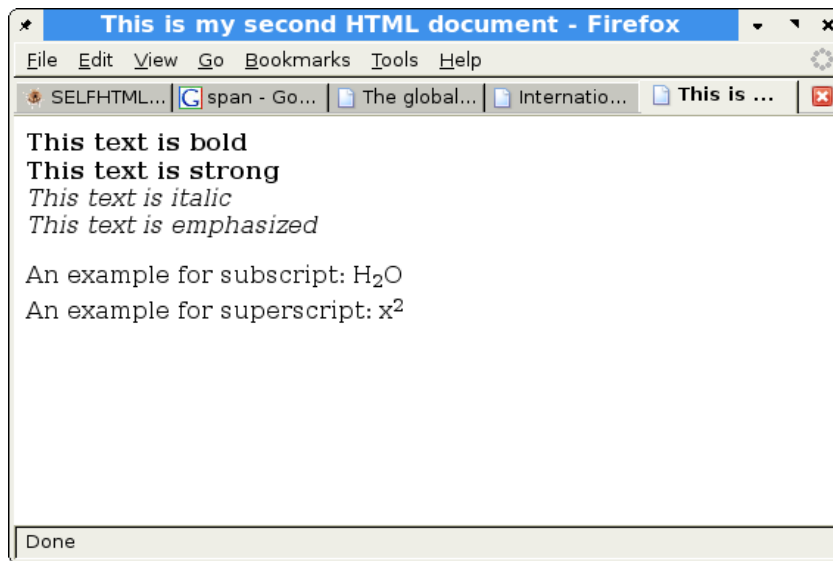
Logical and Physical Markup

These examples demonstrate how you can mark-up text in an HTML document when you want to describe its function and desired appearance.

```
<b>This text is bold</b> <br>
<strong> This text is strong </strong> <br>
<i>This text is italic</i> <br>
<em>This text is emphasized</em>

<p>
An example for subscript: H<sub>2</sub>O <br>
An example for superscript: x<sup>2</sup>
</p>
```

And this is what it will look like:



While “b” (for “bold”) and “i” (for “italics”) are *physical* markups (they prescribe the look of the text), the markups “strong” and “em” (for “emphasized”) are *logical* markups, i.e. they rather describe the idea behind the look. As you can see in the above example, the default behaviour of a browser is usually to display “strong” passages in bold, and “emphasized” passages in italics. However, this default can be overridden with the help of style sheets (more on that later). For example, the composer of a web site might decide that emphasized text should be displayed in bold, and strong text should be displayed in bold and in a larger font. Generally, it is better to use logical markup than physical markup, as this describes the *meaning* of a text passage rather than its look. You are more flexible that way and can re-use the same page later with different style settings.

Special characters

Certain characters, such as the angle-brackets (< >), the ampersand (&) and others are reserved by HTML to represent special things – for example, the left angle-bracket denotes the start of an HTML tag. If you actually want an angle-bracket to appear in your text, you need to use a special HTML command. In addition there are many ISO-Latin 1 characters that you may wish to include in a document, but which are not trivially available on a standard keyboard. HTML gives you the possibility of including these special characters through certain commands.

Commands for special characters consist of three parts:

- a leading ampersand character, (&),
- the name of the entity (in ascii characters)
- a terminating semicolon (;).

Examples:

<	&lt; (for “lower than”)
>	&gt; (for “greater than”)
&	&amp; (for “ampersand”)
ä	&auml; (for “a umlaut”)
ö, ü	&ouml; &uuml;
Ä, Ö, Ü	&Auml; &Ouml; &Uuml;
€	&euro;
©	&copy;

So, for example the German sentence

```
M&uuml;nchen ist eine sch&ouml;ne Stadt!
```

will display as: München ist eine schöne Stadt!

Formatting text with Attributes

Attributes can be used to change a tag's properties. For example, a paragraph's property might be its alignment (left, right or center). A text's property might be its font size or color.

To change these properties, tags can be extended by attributes:

```
<p style="text-align:center"> a centered paragraph </p>  
<p style="text-align:right"> a paragraph aligned right </p>
```

The paragraph tag “p” here has the attribute “style”, and the attribute is assigned a value, namely “text-align:center” (or “text-align:right”). The value of an attribute should always be in quotes (“”). HTML does not enforce this, but XHTML does.

This particular attribute “style” is our first example of CSS (Cascading Style Sheets) technology. Style sheets allow you to specify the presentation (look) of your text. We will learn more about CSS later.

In older HTML documents, you may also see the attribute “align”, for example: `<p align="center"> another centered paragraph </p>`. This still works in HTML, but is deprecated, i.e. you are not supposed to use this attribute anymore.

Changing fonts

You can change the size of text similarly by writing

```
<p style="font-size:250%">Pretty large text</p>  
<p style="font-size:50%">Pretty small text</p>
```

Here, the first line is displayed in a large font size, the second line in a small font size. If you don't want to change the the size of text for a whole paragraph, but just for a passage (a sequence of words) within a paragraph, for example, you can use the “span” tag:

```
<p> This text is in normal size. Now,  
<span style="font-size:250%">this is large</span> and now we go  
back to normal size. </p>
```

Again, you might come across a deprecated older tag for specifying text properties (size, face and color): the tag. For setting size, it was used like this: Here is a size 5 font, and you could set 7 different levels of sizes from 1 (smallest) to 7 (largest), where the default was 3. However, the recommended method for setting text properties nowadays is via styles, as shown before.

To change the font type, you can use “font-family”. Choose a different font face using any font you have installed. Be aware that if the user viewing the page doesn't have the font installed, it will not be displayed. Instead, the default font Times New Roman (which is a default browser setting) will be displayed. An option is to choose a few fonts that are similar in appearance:

```
<span style="font-family:Arial,Verdana,Helvetica">nice  
font...</span>
```

If you specify various fonts like this, the browser will try the first one first, and if that font is not installed, the second, and so on.

The old way was to use the element “font” with the attribute face: some text .

Colors

The attribute changing the color is color. It's values can be color names like red, green, blue etc.








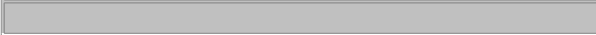

```
<span style="color:red">red text</span>
```

However, only 16 color names are supported by the W3C HTML 4.0 standard (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow). For all other colors you must specify how the color is made up from the three base colors red, green and blue (RGB). You need to specify for all three base colours an intensity between 0 (lowest) and 255 (highest), giving you the color that results from mixing these base colors with the specified intensities. For example, assigning 0 to red, 0 to green and 255 to blue gives you “100% blue”.

There are two ways of defining RGB values – firstly using hexadecimal notation, secondly using decimal numbers. When defining the RGB values of a color in hexadecimal notation, you write a “#” followed by six digits, of which two represent the red value, two the green value and two the blue

value. These are in hexadecimal notation, i.e. the lowest value 0 is written as “00”, the highest value 255 is written as “FF”, for example “#0000FF”. When using the decimal numbers, you write “rgb”, followed by a triplet of numbers, in parentheses and separated by commas, for example “rgb(0,0,255)”.

This table shows the result of combining red, green and blue in various ways:

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

```
<p style="color:#FF00FF">WOW! This is pink</p>
<p style="color:rgb(255,0,255)">This is pink, too!</p>
```

Images

Images can be used to make your web pages distinctive, and can greatly help to get your message across. The simple way to add an image is using the tag. Let's assume you have an image file called "tux.gif" in the same folder/directory as your HTML file. It is 200 pixels high by 150 pixels wide.

```

```

The src attribute names the image file. The width and height aren't strictly necessary but help to speed up the display of your web page. However, if your image has different height or width from what you specify, the browser will scale your image to the values you specified.

One more thing that you should do when using pictures is to display an “alternative” text. People who can't see the image (for example, because the image takes a long time to load) should see something instead, like a short textual description of the picture. You can add an alternative text as follows:

```

```

You can create images in a number of ways, for instance with a digital camera, by scanning a picture in, or by creating one with a painting or drawing program. Most browsers understand GIF and JPEG image formats, newer browsers also understand the PNG image format. To avoid long delays while the image is downloading from the network, you should avoid using large image files.

Generally speaking, JPEG is best for photographs and other smoothly varying images, while GIF and PNG are good for graphics art involving flat areas of color, lines and text. All three formats support options for progressive rendering where a crude version of the image is sent first and then progressively refined.

Hyperlinks

What makes the web so effective is the ability to define links from one page to another, and to follow links at the click of a button. A single click can take you right across the world!

Links are defined with the `<a>` tag. Let's create a link to the page defined in the file "peter.html":

```
This a link to <a href="peter.html">Peter's homepage</a>.
```

The text between the `<a>` and the `` is used as the caption for the link. It is common for the caption to be in blue underlined text. It is by the way a good idea to not have any blank spaces in the names of your HTML files, as this might create problems with some web servers. You can use an underscore, “_”, to separate words in your file names.

To link to a page on another web site you need to give the full web address (the URL). For instance, to link to “Google” you need to write:

```
A link to <a href="http://www.google.com">Google</a>.
```

If you want the user's browser to open a new window for the linked page, (that way the user finds back to your page as soon as he or she closes the new window), use the attribute `target`:

```
A link to <a href="http://www.google.com" target='_blank'>Google</a>.
```

Lists

HTML supports three kinds of lists. The first kind is a bulleted list, often called an unordered list. It uses the `` and `` tags. For instance,

```
<ul>
  <li>the first list item</li>

  <li>the second list item</li>

  <li>the third list item</li>
</ul>
```

results in

- the first list item
- the second list item
- the third list item

Note that you always need to end the list with the `` end tag, but that the `` is optional and can be left out (though that is not recommended). The second kind of list is a numbered list, often called an ordered list. It uses the `` and `` tags. For instance,

```
<ol>
  <li>the first list item</li>

  <li>the second list item</li>

  <li>the third list item</li>
</ol>
```

results in

1. the first list item
2. the second list item
3. the third list item

Tables

Tables are defined with the `<table>` tag. A table is divided into rows (with the `<tr>` tag for “table rows”), and each row is divided into data cells (with the `<td>` tag for “table data”). A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

If you do not specify the “border” attribute as in the example above, the table will be displayed without any borders.

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

results in the following:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing.

To avoid this, add a non-breaking space () to empty data cells, to make the borders visible:

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
```

```
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

XHTML

We have made comments about XHTML syntax throughout this tutorial. Now all that you need to know to create a valid XHTML is how the header differs from HTML. Here is a simple XHTML page:

```
<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <title>This is my very first XHTML document</title>
    <meta http-equiv="content-type" content="text/html;
      charset=ISO-8859-1"/>
  </head>
  <body>
    Here is where the actual content of the page goes.
    Note that it does not matter where I start a new line...

    or wether I leave space between two lines.
  </body>

</html>
```

Firstly, you will notice that there is an additional first line: `<?xml version="1.0" ?>`. Secondly, the DOCTYPE declaration looks different: the "html" after DOCTYPE is now spelled in lower case, and the rest of the command says that your document is written in (the transitional version of) XHTML 1.0. Thirdly, the "html" tag now has an attribute: `xmlns="http://www.w3.org/1999/xhtml"`. The rest of the document is the same as in our HTML example – note that in the "meta" tag, the content of the document is still classified as „text/html“.

Checking the correctness of your code

It is a good idea to check the correctness of your code (be it HTML or XHTML) before publishing it on a web site. To do this, you can use one of the *validator* programs that are freely available on the internet, for example from the W3 Consortium at <http://validator.w3.org/>.

Reference

You can find many tutorials for HTML on the web. A simple one, that contains slightly more than we have covered here, is found at:

<http://www.w3.org/MarkUp/Guide/>

and following the links from there to more advanced topics.

Overview HTML tags

Name	Description
A	anchor
ABBR	abbreviated form (e.g., WWW, HTTP, etc.)
ACRONYM	
ADDRESS	information on author
APPLET	Java applet
AREA	client-side image map area
B	bold text style
BASE	document base URI
BASEFONT	base font size
BDO	l18N BiDi over-ride
BIG	large text style
BLOCKQUOTE	long quotation
BODY	document body
BR	forced line break
BUTTON	push button
CAPTION	table caption
CENTER	shorthand for DIV align=center
CITE	citation
CODE	computer code fragment
COL	table column
COLGROUP	table column group
DD	definition description
DEL	deleted text
DFN	instance definition
DIR	directory list
DIV	generic language/style container
DL	definition list
DT	definition term
EM	emphasis
FIELDSET	form control group
FONT	local change to font
FORM	interactive form
FRAME	subwindow
FRAMESET	window subdivision
H1	heading
H2	heading
H3	heading
H4	heading
H5	heading
H6	heading
HEAD	document head
HR	horizontal rule
HTML	document root element
I	italic text style
IFRAME	inline subwindow

IMG	Embedded image
INPUT	form control
INS	inserted text
ISINDEX	single line prompt
KBD	text to be entered by the user
LABEL	form field label text
LEGEND	fieldset legend
LI	list item
LINK	a media-independent link
MAP	client-side image map
MENU	menu list
META	generic metainformation
NOFRAMES	alternate content container for non frame-based rendering
NOSCRIPT	alternate content container for non script-based rendering
OBJECT	generic embedded object
OL	ordered list
OPTGROUP	option group
OPTION	selectable choice
P	paragraph
PARAM	named property value
PRE	preformatted text
Q	short inline quotation
S	strike-through text style
SAMP	sample program output, scripts, etc.
SCRIPT	script statements
SELECT	option selector
SMALL	small text style
SPAN	generic language/style container
STRIKE	strike-through text
STRONG	strong emphasis
STYLE	style info
SUB	subscript
SUP	superscript
TABLE	
TBODY	table body
TD	table data cell
TEXTAREA	multi-line text field
TFOOT	table footer
TH	table header cell
THEAD	table header
TITLE	document title
TR	table row
TT	teletype or monospaced text style
U	underlined text style
UL	unordered list
VAR	instance of a variable or program argument