

INTRODUCTION TO HTML PROGRAMMING

Basically, a computer sees an "A" as simply an "A" - whether it is bold, italic, big or small.

To tell the browser that an "A" should be bold we need to put a markup in front of the A.

Such a markup is called a Tag.

All HTML tags are enclosed in < and >.

Example: a piece of text as it appears on the screen.

This is an example of **bold** text.

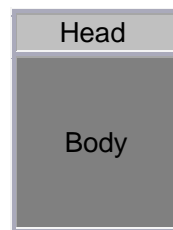
HTML: the HTML for the above example:

This is an example of bold text.

As you can see, the start tag indicates that whatever follows should be written in bold. The corresponding end tag indicates that the browser should stop writing text in bold.

PAGE STRUCTURE

All normal webpages consist of a head and a body.



- The head is used for text and tags that do not show directly on the page.
- The body is used for text and tags that are shown directly on the page.

Finally, all webpages have an <html> tag at the beginning and the end, telling the browser where the document starts and where it stops.

The most basic code - the code you will use for any page you make, is shown below:

```
<html>
<head>
<!-- This section is for the title and technical info of the page. -->
</head>
<body>
<!-- This section is for all that you want to show on the page. -->
</body>
</html>
```

HEAD SECTION

The head section of the webpage includes all the stuff that does not show directly on the resulting page.

The <title> and </title> tags encapsulate the title of your page. The title is what shows in the top of your browser window when the page is loaded. Right now it should say something like "Basics - Html Tutorial" on top of the window containing this text.

Another thing you will often see in the head section is [metatags](#). Metatags are used for, among other things, to improve the rankings in search engines.

Quite often the head section contains [javascript](#) which is a programming language for more complex HTML pages.

Finally, more and more pages contain codes for cascading style sheets (CSS). CSS is a rather new technique for optimizing the layout of major websites.

Since these aspects are way out of reach at this stage we will proceed with explaining the body section.

BODY SECTION

The body of the document contains all that can be seen when the user loads the page.

In the rest of this tutorial you can learn in detail about all the different aspects of HTML, including:

- Text
 - Formatting
 - Resizing
 - Layout
 - Listing
- Links
 - To local pages
 - To pages at other sites
 - To bookmarks
- Images
 - Inserting images (GIF and jpg)
 - Adding a link to an image
- Backgrounds
 - Colors
 - Images
 - Fixed Image
- Tables
- Frames
- Forms
- Metatags
- Hexadecimal Colors

The last page in this introduction will give you an overview of how to proceed with the HTML tutorial (and beyond).

HTML TEXT

To enter text on your pages - all you have to do is simply enter the text.

If you do not specify any attributes for text it will use the default size, font etc. of the visitor's browser.

Browsers can only show fonts available on the visitor's PC.

Therefore you are limited to using the fonts that are available on almost any computer. If you need to use a fancy font, you should use your graphics program to make an image with the text. This will assure that the visitor will see it - even if he doesn't have the fancy font you're using.

Since images take up much more space than plain text, thus increasing download time, you should use this option with care.

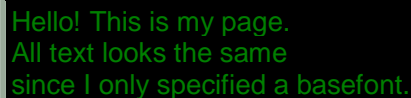
On the following pages you will learn how to customize the font settings - both on normal text and text that works as a link.

Furthermore, you will learn to control how the text aligns on your pages.

BASE FONT

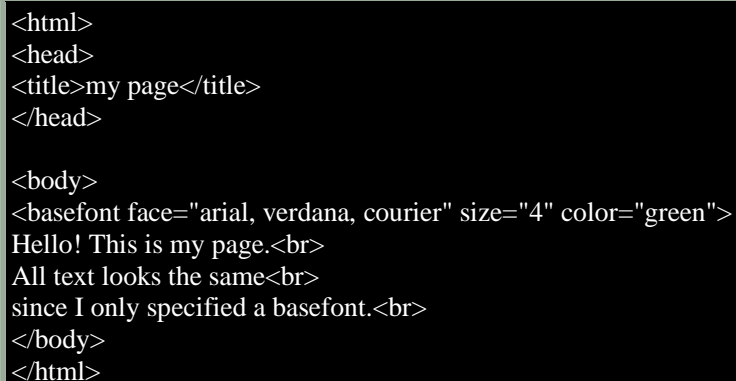
To specify the overall font for your page add the <basefont> tag at the beginning of the <body> section.

Example: The page as it looks in the browser.

A screenshot of a web browser window with a black background. The text inside is green and reads: "Hello! This is my page." followed by a line break, "All text looks the same" followed by a line break, and "since I only specified a basefont.".

Hello! This is my page.
All text looks the same
since I only specified a basefont.

HTML: The code to produce the above example.

A screenshot of a code editor showing HTML code. The code is as follows:

```
<html>
<head>
<title>my page</title>
</head>

<body>
<basefont face="arial, verdana, courier" size="4" color="green">
Hello! This is my page.<br>
All text looks the same<br>
since I only specified a basefont.<br>
</body>
</html>
```

The color attribute selects the desired color for your text. The face attribute selects the desired font.

Note:

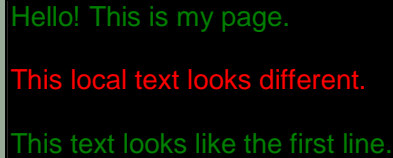
If you enter a list of fonts, like in the example, the browser will use the first font in the list available on the visitor's computer.

The size attribute specifies the desired size, between 1 (smallest) and 7 (biggest).

FONT

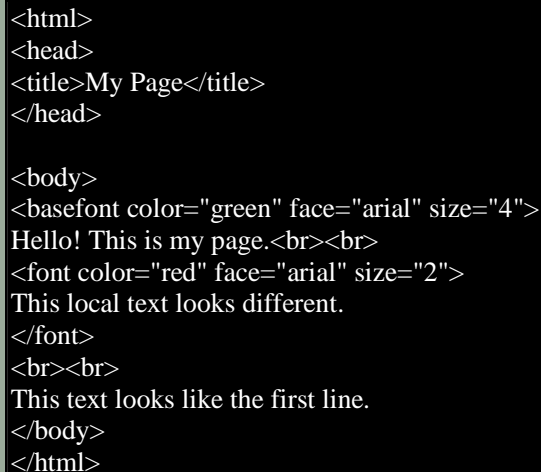
The tag will change the font.

Example: How the output looks in the browser.



Hello! This is my page.
This local text looks different.
This text looks like the first line.

HTML: The code to produce the above example.



```
<html>
<head>
<title>My Page</title>
</head>

<body>
<basefont color="green" face="arial" size="4">
Hello! This is my page.<br><br>
<font color="red" face="arial" size="2">
This local text looks different.
</font>
<br><br>
This text looks like the first line.
</body>
</html>
```

The color attribute selects the desired color for your text. The face attribute selects the desired font.

Note:

If you enter a list of fonts, like in the example, the browser will use the first font in the list available on the visitor's computer.

TEXT LINKS

The tags used to produce links are the <a> and .

The <a> tells where the link should start and the indicates where the link ends.

Everything between these two will work as a link.

The target of the link is added to the <a> tag using the href="http://www.whateverpage.com" setting.

The example below shows how to make the word here work as a link to yahoo.

Click [here](http://www.yahoo.com) to go to yahoo.

You simply:

- Specify the target in the ``.
- Then add the text that should work as a link.
- Finally add an `` tag to indicate where the link ends.

TEXT FORMAT

These are the tags for text formats:

<code>text</code>	writes text as bold
<code><i>text</i></code>	writes text in italics
<code><u>text</u></code>	writes underlined text
<code><sub>text</sub></code>	lowers text and makes it smaller
<code><sup>text</sup></code>	lifts text and makes it smaller
<code><blink>text</blink></code>	guess yourself! (Note: Netscape only.)
<code><strike>text</strike></code>	strikes a line through the text
<code><tt>text</tt></code>	writes text as on a classic typewriter
<code><pre>text</pre></code>	writes text exactly as it is, including spaces.
<code>text</code>	usually makes text italic
<code>text</code>	usually makes text bold

Note:

The `<blink>` tag is only supported by Netscape browsers, and should be avoided.

TEXT SIZE

These are the tags for changing the font size.

<code><big>text</big></code>	increase the size by one
<code><small>text</small></code>	decrease the size by one
<code><h1>text</h1></code>	writes text in biggest heading
<code><h6>text</h6></code>	writes text in smallest heading
<code>text</code>	writes text in smallest fontsize. (8 pt)
<code>text</code>	writes text in biggest fontsize (36 pt)

The <small> and <big> tags are special in that they can be repeated. If you want to increase the font size with a factor two, then you could do it like this:

```
bla bla bla <big><big>whatever</big></big> bla bla bla
```

Note:

While the font tag lets you specify font attributes in plain HTML, you really should look into the tutorial on CSS to learn how to get full, flexible and much more advanced control of your text.

TEXT LAYOUT

These tags will let you control the layout.

HTML	EXPLANATION
<code><p>text</p></code>	Adds a paragraph break after the text. (2 linebreaks).
<code><p align="left">text</p></code>	Left justify text in paragraph.
<code><p align="center">text</p></code>	Center text in paragraph.
<code><p align="right">text</p></code>	Right justify text in paragraph.
<code>text
</code>	Adds a single linebreak where the tag is.
<code><nobr>text</nobr></code>	Turns off automatic linebreaks - even if text is wider than the window.
<code>text<wbr></code>	Allows the browser to insert a linebreak at exactly this point - even if the text is within <nobr> tags.
<code><center>text</center></code>	Center text.
<code><div align="center">text</div></code>	Center text.
<code><div align="left">text</div></code>	Left justify text.
<code><div align="right">text</div></code>	Right justify text.

Example: the difference between layout tags:

RESULT	HTML
Hello world- a linebreak does not insert a linebreak in HTML	Hello world - a linebreak does not insert a linebreak in HTML
you will need	<code><p>you will need</p></code>
to insert	<code><p align="right">to insert</p></code>
special tags	<code><p align="left">special tags</p></code>
that will insert linebreaks where you want it!	that will insert linebreaks where you want it!

<p>Another method is to write a sentence, that is long enough to force a linebreak.</p> <p>This option can however be turned off with the nobr-tag, unless a wbr is used to force it!</p> <p>You can also center And turn the center off And on! Go left! Go right!</p>	<pre>
 Another method is of course to write a sentence, that is long enough to force a linebreak.

 <nobr>This option can however be turned off<wbr>with the nobr tag,<wbr>unless a wbr is used to force it!</nobr> <center>You can center</center> And turn the center off <div align="center">And on!</div> <div align="left">Go left!</div> <div align="right">Go Right!</div></pre>
---	--

Note in particular the difference between the `<p>` and the `<div>` tags. The `<div>` tag allows you to justify content without being forced to add a double linebreak.

Also, note that these alignment tags are not limited to text. They work on text, images, applets or whatever it is that you insert on the page.

BULLETED LISTS

To create a bulleted list you need to add a `` and a `` tag at the beginning and the end of the list.

Numbered lists have `` tags instead of `` tags.

To separate single list items use `` and `` tags.

There are special settings that you can use to customize the lists on your page.

BULLETED LISTS

This page shows how to make different kinds of bulleted lists.

You have the following bullet options:

- disc
- circle
- square

Look at these examples to see the detailed syntax.

HTML-CODE	EXPLANATION / EXAMPLE
	Makes a bulleted list using the default bullet type:
<pre> text text text </pre>	<ul style="list-style-type: none"> • text • text • text
	Starts a bulleted list using discs as bullets:
<pre><ul type="disc"></pre>	<ul style="list-style-type: none"> • This is one line • This is another line • And this is the final line
	Starts a bulleted list using circles as bullets:
<pre><ul type="circle"></pre>	<ul style="list-style-type: none"> ○ This is one line ○ This is another line ○ And this is the final line
	Starts a bulleted list using squares as bullets:
<pre><ul type="square"></pre>	<ul style="list-style-type: none"> ▪ This is one line ▪ This is another line ▪ And this is the final line

On the next page you can learn how to create and customize numbered lists....

NUMBERED LISTS

You have the following number options:

- Plain numbers
- Capital Letters
- Small Letters
- Capital Roman Numbers
- Small Roman Numbers

In addition to these options you can specify at which number the list should start. The default start value for numbered lists is at number one (or the letter A).

Look at these examples to see the detailed syntax.

HTML-CODE	EXPLANATION / EXAMPLE
<pre> text text text </pre>	<p>Makes a numbered list using the default number type:</p> <ol style="list-style-type: none"> 1. text 2. text 3. text
<pre><ol start="5"></pre>	<p>Starts a numbered list, first # being 5.</p> <ol style="list-style-type: none"> 5. This is one line 6. This is another line 7. And this is the final line
<pre><ol type="A"></pre>	<p>Starts a numbered list, using capital letters.</p> <ol style="list-style-type: none"> A. This is one line B. This is another line C. And this is the final line
<pre><ol type="a"></pre>	<p>Starts a numbered list, using small letters.</p> <ol style="list-style-type: none"> a. This is one line b. This is another line c. And this is the final line
<pre><ol type="I"></pre>	<p>Starts a numbered list, using capital roman numbers.</p> <ol style="list-style-type: none"> I. This is one line II. This is another line III. And this is the final line
<pre><ol type="i"></pre>	<p>Starts a numbered list, using small roman numbers.</p> <ol style="list-style-type: none"> i. This is one line ii. This is another line iii. And this is the final line
<pre><ol type="1"></pre>	<p>Starts a numbered list, using normal numbers.</p> <ol style="list-style-type: none"> 1. This is one line 2. This is another line

<pre><ol type="I" start="7"></pre>	<p>3. And this is the final line</p> <p>An example of how type and start can be combined.</p> <p>VII. This is one line</p> <p>VIII. This is another line</p> <p>IX. And this is the final line</p>
--	--

HTML IMAGES

This section will show how to add images to your pages.

We will start out with a presentation of the two main image types on webpages: jpg and gif.

After that, we will proceed with various ways to insert and customize images, with a special focus on the different alignments you can choose.

GIF & JPG

Computers store images in several different ways.

Some storage methods focus on compressing the size of the image as much as possible. A major problem with using images on websites is that images take much longer to load than text.

To reduce download times as much as possible two of the best image compressing formats used on the web are:

GIF	JPG
256 colors	Unlimited colors
Can handle transparent areas	Can't handle transparent areas
This format is not good at compressing photographs	Excellent for compressing photographs and complex images
In general, it is excellent for banners, buttons and clipart	In general, it is not good for banners, buttons and clipart.

This means that:

- Banners, buttons, dividers, clipart and other simple images usually work best as GIF's.
- Photographs and other complex images usually work best as JPG's.

If you want to use an image that is in a format other than JPG or GIF, you will need to load the image into a graphics program and save it as either JPG or GIF.

INSERTING IN HTML

The tag used to insert an image is called img.

Below you see an image called "rainbow.gif".



Here is the HTML code used to insert the image on this webpage:

```

```

If the image is stored in the same folder as the HTML page, you can leave out the domain reference (<http://www.echoecho.com/>) and simply insert the image with this code:

```

```

On the following pages we will discuss different ways to control how your image is inserted.

RESIZING

You can change the size of an image using the width and height attributes.

In general, it is not advisable to reduce image size using these settings, since the image will be transferred over the internet in its original size no matter what reduction is set for it. This will slow the loading of your webpage.

This means, that if you have an image that is bigger in size than you want it to be on your page, you should reduce the size in a graphics program, rather than reducing the size on the webpage using the width and height attributes.

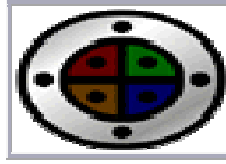
On the contrary, sometimes, it can be wise to enlarge images using this technique.

Below are two presentations of the exact same image - with different settings for width and height.



```

```



```

```

If you leave out the settings for width and height, the browser will automatically use the real size of the image.

However, you should always enter the settings for width and height, even when using the real size!

The reason is that if the settings are left out, the browser can't build the page until the image is loaded entirely.

This means, that the visitor cannot read text around the image while the image itself is loading - which in turn will give the visitor an impression of a slow loading page.

This becomes especially true if the image is inside a table.
In that case, the whole table will not be shown until the image is loaded entirely.

BORDER AROUND

You can add a border to the image using the border setting shown in the example below:

Note:

Netscape browsers will only show the border if the image is a link.



```

```

Adding a border to your image might help the visitor recognize that the image is a link. However, the net is filled with images that work as links and have no borders indicating it - so the average visitor is used to letting the mouse run over images to see if they are links.

Still - if you have an image that is often mistaken you might consider adding a border to it - although you should probably consider changing the image entirely - since if it does not indicate by itself that it is a link then it isn't serving it's purpose.

ALTERNATIVE TEXT

You can add an alternative text to an image using the alt setting shown in the example below:



```

```

You should always add alternative texts to your images, so the users can get an idea of what the image is about before it is loaded.

This becomes particularly important if the image is a link.

Few things are as annoying as knowing that you want to leave the current page - and at the same time being forced to wait for an image to load before being able to do so.

It is extremely tempting to use the browser's straightforward options to leave the entire site instead.

SPACING AROUND

You can easily add space over and under your images with the Vspace attribute.

In a similar way you can add space to the left and right of the image using the Hspace attribute.

Below is an example using these attributes:



```

```

As you see these settings allow you to add spacing around your image. Unfortunately, they also force you to add the same spacing to each side of the image (over and under - or left and right).

The workaround for this, if you only want spacing on one side of the image is to use a 1x1 pixel transparent gif image.

If, for example, you wanted a 10 pixel spacing to the left of your image you could use

the transparent image (pixel.gif) this way:



```

```

The 1x1 pixel transparent gif image is simply stretched to whatever size you want the spacing to have.

This 1x1 pixel "cowboy-trick" is probably one of the most widely used workarounds on the entire net.



The reasons are obvious: It works on all browsers and it gives you complete pixel precision in your design!

ALIGNMENT OF IMAGES

You can align images according to the text around it, using the following alignments:

- default aligns the image using the default settings of the Web browser. Same as baseline.
- left aligns the image in the left margin and wraps the text that follows the image.
- right aligns the image in the right margin and wraps the text that precedes the image.
- top aligns the top of the image with the surrounding text.
- texttop aligns the top of the image with the top of the tallest text in the line.
- middle aligns the middle of the image with the surrounding text.
- absmiddle aligns the image with the middle of the current line.
- baseline aligns the image with the baseline of the current line.
- bottom aligns the bottom of the image with the surrounding text.
- absbottom aligns the image with the bottom of the current line.
- center aligns the center of the image with the surrounding text.

In the table below you can see examples of the different vertical alignments you can make for an image.

The note  in the examples is only there to show how the circular sign  is affected by other images on the same line.


This means, that the alignments shown in the example are made to the circular sign and not the note.

HTML

EXAMPLE

```

```


	bla  bla  bla bla
<code></code>	bla  bla  bla bla
<code></code>	bla  bla  bla bla
<code></code>	bla  bla  bla bla
<code></code>	bla  bla  bla bla
<code></code>	bla  bla  bla bla
<code></code>	bla  bla  bla bla

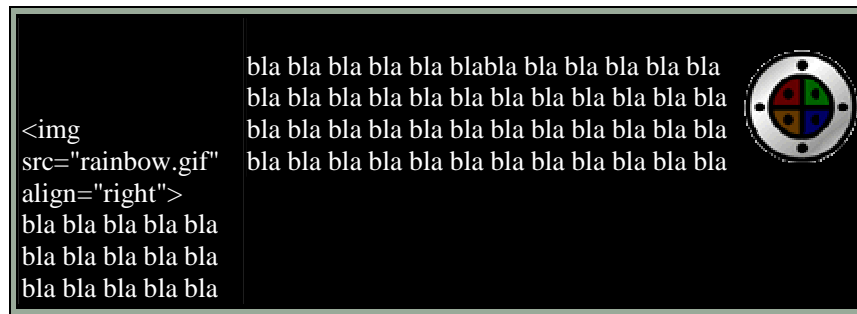
WRAP TEXT AROUND

In addition to the vertical alignments covered on the [previous page](#), images can also be aligned horizontally.

To do this, add `align="left"` or `align="right"` to the `` tag.

Consider these examples to see how it works:

HTML-CODE	EXAMPLE
<pre> bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla</pre>	 bla bla bla bla bla blabla bla



Another way to obtain the same effect would be to enter the image and text in an invisible table. Entering text in one column and the image in another would create a similar effect.

HTML LINKS

Links are the most fundamental part of the world wide web.
It is the links that tie it all together.

There are three different kinds of links you can have on your website:

- Links to anchors on the current page (Internal).
- Links to other pages within the current site (Local)
- Links to pages outside the current site (Global).

It is possible to make texts and images work as links.
With a little creativity other objects, such as pushbuttons or even [drop-down menus](#) can work as links as well.

This section will cover the usual links: Texts and Images.

HOW TO MAKE A LINK

The tags used to produce links are the `<a>` and ``.

The `<a>` tells where the link should start and the `` indicates where the link ends.

Everything between these two will work as a link.

The target of the link is added to the `<a>` tag using the `href="http://www.whateverpage.com"` setting.

The example below shows how to make the word here work as a link to yahoo.

Click <code>here</code> to go to yahoo.
--

You simply:

- Specify the target in the .
- Then add the text that should work as a link.
- Finally add an tag to indicate where the link ends.

COLORS ON TEXT LINKS

There are a few settings that can be useful for controlling the colors of text links. This page will teach you how to:

- Define colors for all links on the page.
- Define colors for individual links on the page.

Define colors for all links on the page

The general color of text links is specified in the <body> tag, like in the example below:

```
<body link="#C0C0C0" vlink="#808080" alink="#FF0000">
```

- link - standard link - to a page the visitor hasn't been to yet. (standard color is blue - #0000FF).
- vlink - visited link - to a page the visitor has been to before. (standard color is purple - #800080).
- alink - active link - the color of the link when the mouse is on it. (standard color is red - #FF0000).

Note

You can click [here](#) to learn more about the hexadecimal colorsystem that is used in HTML.

Define colors for individual links on the page

The method described above is for setting overall link colors for the page.

However, you might want one or more links to have different colors than the rest of the page.

There are two methods for doing this:

- Placing font tags between the <a href> and the tag.
This method will work on all browsers except MSIE 3.
- Using a style setting in the <a> tag.
This method works on MSIE3 and newer browsers.

The first technique would look like this:

```
Click <a href="http://www.yahoo.com"><font
```

```
color="FF00CC">here</font></a> to go to yahoo.
```

Note:

It is important that both the and the tags are between the <a href> and tags.

The second technique would look like this:

```
Click <a href="http://www.yahoo.com" style="color:
rgb(0,255,0)">here</a> to go to yahoo.
```

Note:

The RGB numbers indicate amounts of red, green, and blue using values between 0 and 255. You can read more about converting between RGB colors and hexadecimal colors [here](#).

Now, since neither of the two methods covers all browsers, we need to use both techniques at once.

This example will work on all browsers:

```
Click <a href="http://www.yahoo.com" style="color: rgb(0,255,0)"><font
color="FF00CC">here</font></a> to go to yahoo.
```

The last example is interesting. Not only because it will work on all browsers. But even more because it shows a general approach to making your pages browser safe.

Since browsers simply leave out information that is not understood, you can work around browser differences by simply adding different settings for multiple browsers.

LINK TARGETS

By default, links will open in the current window or frame.

You need to add a target if you want the link to open in another window or frame than the link itself is placed in.

To do this you simply add a target="" to the <a href>.

This example will open yahoo in a new window:

```
<a href="http://www.yahoo.com" target="_blank">
```

Predefined targets are:

- _blank loads the page into a new browser window.
- _self loads the page into the current window.
- _parent loads the page into the frame that is superior to the frame the hyperlink is in.
- _top cancels all frames, and loads in full browser window.

In addition to these, you can also enter the name of a frame window if your page is within a frameset.

NO UNDERLINE

By default, text links are underlined by the browser.

If your page is visited by MSIE3 or newer, you can turn off the underlining for an entire page by adding a style tag to the head section of the document.

Look at this example:

```
<html>

<head>
<title>This is my page</title>
<style type="text/css">
<!--
A{text-decoration:none}
-->
</style>
</head>

<body>
Welcome to my world!<br>
<a href="http://www.yahoo.com">This Link To Yahoo has no
underline</a>
</body>

</html>
```

Note:

The style setting will not cause an error if viewed on a browser that doesn't support it. The browser will simply skip the effect - the link will look like an ordinary underlined link - but no errors will occur.

ADVANCED TEXT LINKS

Instead of just turning off the underline on all links you could be more specific in defining the way you want your links to work.

In the example below underlining is turned off for all links.

The A:hover tells the browser that when the mouse is over a link the underline should appear.

The hover option only works on MSIE 4+.

(But it does not cause an error on Netscape if you include it - the effect just does not appear.).

```
<html>

<head>
```

```

<title>This is my page</title>
<style type="text/css">
<!--
A:link {text-decoration: none}
A:visited {text-decoration: none}
A:active {text-decoration: none}
A:hover {text-decoration: underline}
-->
</style>

</head>

<body>
Welcome to my world!<br>
<a href="http://www.yahoo.com">This Link To Yahoo has no
underline</a>
</body>

</html>

```

The methods described above will turn off the underline effect for links on the entire page.

If you want to turn off the effect for just a single link, add a style property to the <a href> tag:

```

<a href="http://www.yahoo.com" style="text-decoration: none">Go to
Yahoo</a>

```

IMAGE LINKS

If you want to make an image work as a link, the method is exactly the same as with texts.

You simply place the <a href> and the tags on each side of the image.

Below is the HTML code used to make the image work as a link to a page called myfile.htm:



```

<a href="myfile.htm"></a>

```

If you haven't entered a border setting you will see a small border around the image after turning it into a link. To turn off this border, simply add border="0" to the tag:

```
<a href="myfile.htm"></a>
```

Images that work as links can show a popup text when you place the mouse over it. This is done with the alt property in the tag.

For example:

```
<a href="myfile.htm"></a>
```



IMAGE MAPPING

It is possible to make one image link to several pages, depending on where the image is clicked.

This technique is called *imagemapping*.

You simply specify which areas of the image should link to where.

In the example below, if you position the mouse in the upper left corner it links to yahoo and in the lower right corner.... it links to hotbot.



```
  
<map name=example>  
<area shape=Rect Coords=0,0,29,29 Href="http://www.yahoo.com">  
<area shape=Rect Coords=30,30,59,59 Href="http://www.hotbot.com">  
</map>
```

In the above example we only used rectangular *imagemappings*. Possible shapes are:

- <area shape=rect coords= x1,y1, x2,y2 Href="http://www.domain.com">
- <area shape=circle coords= x1,y1, x2,y2 Href="http://www.domain.com">
- <area shape=polygon coords= x1,y1, x2,y2, ..., xn,yn Href="http://www.domain.com">

There are excellent tools out there to help you create *imagemaps*. No one calculates the coordinates by hand.

If you want to use *imagemaps* on your site you will need to get a program that will allow you to simply drag the mouse over the areas you want to work as links.

Most HTML editors have this as a built-in function.

LINK WITHIN A PAGE

Linking to anchors is very similar to normal links. Normal links always point to the top of a page. Anchors point to a place within a page.

A # in front of a link location specifies that the link is pointing to an anchor on a page. (Anchor meaning a specific place in the middle of your page).

To link to an anchor you need to:

- **Create a link pointing to the anchor**
- **Create the anchor itself.**

An anchor is created using the <a> tag.

If you want to create an anchor called chapter4, you simply add this line where you want the anchor to be:

```
<a name="chapter4"></a>
```

After doing this, you can make a link pointing to the anchor using the normal <a href> tag, like this:

```
Click <a href="#chapter4">here</a> to read chapter 4.
```

Note:

When linking to an anchor on a page you need to put a # in front of the anchor.

When you link to an anchor on the same page, simply enter

```
<a href="#YourAnchor">blabla</a>
```

When you link to anchors on external pages use this syntax:

```
<a href="http://www.yahoo.com#YahoosAnchor">blabla</a>
```

Anchors are generally used when you create pages with considerable amounts of text. You would typically make an index at the top of the page linking to the anchors that have been added to key places in the text that follows.

LINKS IN FRAMESETS

If a non-framebased HTML document contains a hyperlink that links to a page called analysis.htm then it appears in the HTML document somewhat like this:

```
Click here to see the <a href="analysis.htm">Analysis</a> of the project.
```

Now if the same link was in a frameset, (say in the frame window called menu) and we wanted it to link to a page that is loaded in the other frame window, (named main) then the HTML code would be:

```
Click here to see the <a href="analysis.htm" target="main">Analysis</a>
of the project
```

We simply added the desired frame window (main) as a target for the link.

The link will be opened in the main frame window instead of the menu frame window where the link itself is located.

LINK TO NEW WINDOW

If you want your link to open a page in a new window use the target="_blank" in the <a href> tag.

Targetting the link to "_blank" simply opens a new browser window that will load the linked page.

Linking to Yahoo the traditional way would require this link:

```
<a href="http://www.yahoo.com">Go to Yahoo</a>
```

If you add a target="_blank", the page will open in a new window:

```
<a href="http://www.yahoo.com" target="_blank">Go to Yahoo</a>
```

LINK TO EMAIL

Having a link that allows visitors to send email from your website can be a great addition to your site, making it easy for your visitors to send questions or comments.

There is a special link for this action.

Email links are done much the same as links to other pages, using the <a href> tag.

An email link would require the following code:

```
<a href="mailto:youremailaddress">Email Me</a>
```

This will result in the visitor's email program opening a new email with your address already in the To: field.

If you wish to have a specific subject in the email, you can add it to the html code using subject= setting :

```
<a href="mailto:email@echoecho.com?subject=SweetWords">
Send Email</a>
```

Suppose you want an email link for your visitors containing specific text in the body of

their message, simply add &body=:

```
<a href="mailto:email@echoecho.com?body=Please send me a copy of  
your new program!">Send Email</a>
```

Or combine all the options and allow your visitor to send email with the address, subject and text already entered.

```
<a href="mailto:email@echoecho.com?subject=SweetWords  
&body=Please send me a copy of your new program!">Email Me</a>
```

HTML BACKGROUND

When deciding whether you want to use a plain color or an image you should consider the fact that very few of the web's 100 most visited sites use background images.

More than 90 percent have a plain white background.

The few pages that actually do use images use very discrete and fast loading images for the purpose.

When picking the desired color - whether it be plain or an image - you should also consider the fact that some colors work with almost any other color - while there are colors that only work with a limited number of contrasts.

If you use green on a red background, it will look different than if you use the same green on a blue background. Without digging into deep theories about colors, we will make a note on the fact that white, gray and black colors tend to be balanced against other colors. That is, white, gray and black work with any of the colors in the rainbow.

This is probably the reason that white, black and gray are the most widely used background colors found on the net.

BACKGROUND COLOR

Adding a plain background color to your page is easy.

All you need to do is add a bgcolor property to the body tag:

```
<body bgcolor="#FF0000">
```

BACKGROUND IMAGE

If you want to add a background image instead of a plain color there are some considerations you should make before doing so:

- Is the background image discrete enough to not take away the focus from what's written on it?
- Will the background image work with the text colors and link colors I set up for the page?
- Will the background image work with the other images I want to put on the page?
- How long will the page take to load my background image? Is it simply too big?
- Will the background image work when it is copied to fill the entire page? In all screen resolutions?

After answering these questions, if you still want to add the background image you will need to specify in the <body> tag which image should be used for the background.

```
<body background="drkrainbow.gif">
```

Note:

If the image you're using is smaller than the screen, the image will be replicated until it fills the entire screen.

If, say you wanted a striped background for your page, you wouldn't have to make a huge image for it. Basically you could just make an image that is two pixels high and one pixel wide. When inserted on the page the two dots will be copied to fill the page - thus making what looks like a full screen striped image.

When you choose to use a background image for the page it is always a good idea to specify a background color as well.

```
<body background="drkrainbow.gif" bgcolor="#333333">
```

The reason is that until the background image is loaded, the background color will be shown.

If there is too much difference between the background color and the background image, it will look disturbing once the browser shifts from the background color to the image.

Therefore it is a good idea to specify a background color that matches the colors of the image as close as possible.

You may have noticed that background images scroll with the page when you use the scroll bar.

FIXED IMAGE

The background image will scroll when the user scrolls down the page, unless you have set it to be fixed:

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

By adding the bgproperties="fixed" you force the browser to let the background be

fixed even if the user is scrolling down the page.

Note: Fixed backgrounds are only supported by MSIE and do not work in Netscape browsers - instead they simply act as normal backgrounds.

HTML TABLES

Tables are used on websites for two major purposes:

- The obvious purpose of arranging information in a table
- The less obvious - but more widely used - purpose of creating a page layout with the use of hidden tables.

Using tables to divide the page into different sections is an extremely powerful tool. Almost all major sites on the web are using invisible tables to layout the pages.

The most important layout aspects that can be done with tables are:

- Dividing the page into separate sections.
An invisible table is excellent for this purpose.
- Creating menus.
Typically with one color for the header and another for the links following in the next lines.
- Adding interactive form fields.
Typically a gray area containing a search option.
- Creating fast loading headers for the page.
A colored table with a text on it loads like a bullet compared to even a small banner.
- Easy alignment of images that have been cut into smaller pieces.
- A simple way to allow text to be written in two or more columns next to each other.

The importance of using tables for these layout purposes can't be overrated. However there are a few things to keep in mind when doing so.

Most important is, that the content of a table is not shown until the entire table is loaded. If you have extremely long pages, you should divide it into two or more tables - allowing the user to start reading the upper content while the rest of the page is loading.

BASIC TABLES

Tables are defined with the <table> tag.

To insert a table on your page you simply add these tags where you want the table to occur:

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

The above table would be of no use since it has no rows and no columns.

ROWS:

To add rows to your table use the `<tr>` and `</tr>` tags.

Example:

```
<table>
<tr></tr>
<tr></tr>
</table>
```

It doesn't make sense to write the above lines in itself, cause you can't write content outside of table cells.

If you do write things outside of cells it will appear right above the table.

COLUMNS:

You can divide rows into columns with `<td>` and `</td>` tags:

Example:

```
<table>
<tr> <td>This is row one, left side.</td> <td>This is row one, right
side.</td> </tr>
<tr> <td>This is row two, left side.</td> <td>This is row two, right
side.</td> </tr>
</table>
```

Result:

This is row one, left side.	This is row one, right side.
This is row two, left side.	This is row two, right side.

This page has shown the core basics of tables. In addition to these, there are different options for customizing your tables.
The following pages will focus on the different settings for `<table>`, `<tr>` and `<td>` tags.

TABLE TAGS

The following properties can be added to the `<table>` tag:

Property	Description
<code>align=</code>	
<code>left</code>	left align table
<code>center</code>	center table
<code>right</code>	right align table

background=filename	image inserted behind the table
bgcolor=#rrggb	background color
border=n	border thickness
bordercolor=#rrggb	border color
bordercolordark=#rrggb	border shadow
cellpadding=n	distance between cell and content
cellspacing=n	space between cells
nowrap	protects against linebreaks, even though the content might be wider than the browser window.
frame=	
void,	removes all outer borders
above,	shows border on top of table
below,	shows border on bottom of table
lhs,	shows border on left side of table
rhs,	shows border on right side of table
hsides,	shows border on both horizontal sides
vsides,	shows border on both vertical sides
box	shows border on all sides of table
valign=	
top	aligns content to top of cells
bottom	aligns content to bottom of cells
width=	
n,n	minimum width of table in pixels
n,n%	minimum width in percentage of window size

Note:

Table properties are set for the entire table.

If certain properties are set for single cells, they will have higher priority than the settings for the table as a whole.

ROW/CELL TAGS

These settings can be added to both <tr> and <td> tags.

PROPERTY	DESCRIPTION
align=	
left	aligns content to the left of cells
right	aligns content to the right of cells
center	aligns content to the center of the cells
background=filename	sets a background image for the cells
bgcolor=#rrggb	sets a background color for the cells
bordercolor=#rrggb	sets color for the border of cells
bordercolordark=#rrggb	sets color for the border shadow of cells
valign=	
top	aligns to the top of cells

middle	aligns to the middle of the cells
bottom	aligns to the bottom of cells
width=	
n	specify a minimum width for the cells in pixels
n%	specify a minimum width for the cells in percent of the table width
height=	
n	minimum height of cells in pixels
n%	minimum height of cells in percentage of table height

These settings are only valid for <td> tags.

PROPERTY	DESCRIPTION
colspan=n	number of columns a cell should span
nowrap	protects against linebreaks, even though the content of a cell might be wider than the browser window
rowspan=n	number of rows a cell should span

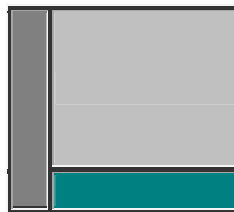
Note:

Settings for columns(<td> tag) have higher priority than settings for rows(<tr> tag).

Settings for cells (<tr> or <td> tags) have higher priority than settings for the table as a whole(<table> tag).

HTML FRAMES

Frames can divide the screen into separate windows.



Each of these windows can contain an HTML document.

A file that specifies how the screen is divided into frames is called a frameset.

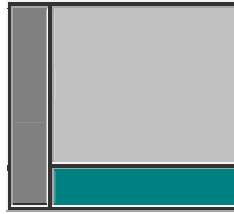
If you want to make a homepage that uses frames you should:

- make an HTML document with the frameset
- make the normal HTML documents that should be loaded into each of these frames.

When a frameset page is loaded, the browser automatically loads each of the pages associated with the frames.

BASIC EXAMPLE

A frameset is simply an HTML document that tells the browser how to divide the screen into split windows.



The HTML for the above frameset:

```
<html>
<head>
<title>My Frames Page</title>
</head>

<frameset cols="120,*">
<frame src="menupage.htm" name="menu">
<frameset rows="*,50">
<frame src="welcomepage.htm" name="main">
<frame src="bottombanner.htm" name="bottom">
</frameset>
</frameset>

</html>
```

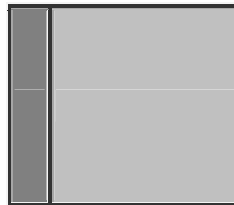
Note that the frameset is only seven lines!

Let's split it all up and add the lines one by one...

CREATING A FRAMESET

As stated on the previous page, a frameset is simply an HTML document that tells the browser how to divide the screen into split windows.

If the frameset looked like this:



The code would be:

```
<frameset cols="120,*">
</frameset>
```

The screen is divided into two columns.
The left being 120 pixels and the right using the rest of the screen (indicated by the *).

The frame windows would have no names, so the frameset really couldn't be used for any purpose.

DEFAULT PAGES

You can add default pages to frame windows with the src setting.

Default pages are the pages that will be loaded when the frameset is opened the first time.

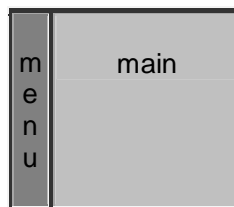
Furthermore, we can add names to each frame window using the name setting.

This will allow us to make a link in one frame window, open a page in another frame window.

In this example we added names and default pages to the frame windows:

```
<frameset cols="120,*" >  
<frame src="menu.htm" name="menu" >  
<frame src="frontf.htm" name="main" >  
</frameset>
```

The entire frameset will look like this:



We still have the screen divided in two columns, the left being 120 pixels the right using the rest of the screen. (some screens are set to 640 pixels across, some to 800 and some to 1024, thats why the * is needed).

But now we also have told the browser that the left frame window should load an HTML page called menu.htm and that the right window should load an HTML document called frontf.htm.

In addition we have assigned the names menu and main to the two frame windows, so now we're even able to link to specific windows.

We called the frame windows menu and main, but you could name them whatever you pleased.

The frameset with a menu window to the left and a main window to the right is the most common frameset seen on the web.

There are a few more settings we could add to the frameset.

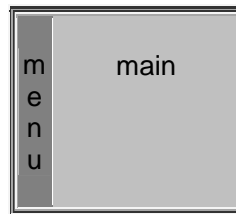
For instance you might want the frame borders to be invisible

BORDERS

To make frame borders invisible you simply need to add the parameters "cols-line" to the frameset :

```
<frameset cols="120,*" frameborder="0" border="0" framespacing="0">
<frame src="menu.htm" name="menu" >
<frame src="frontf.htm" name="main" >
</frameset>
```

The entire frameset would now look like this:



We could still add a few more parameters to our frameset....

RESIZEABLE WINDOWS

If you don't want the frame windows to be resizeable, you should add the parameter "noresize" to the frame src lines:

```
<frameset cols="120,*" frameborder="0" border="0" framespacing="0">
<frame src="menu.htm" name="menu" noresize>
<frame src="frontf.htm" name="main" noresize>
</frameset>
```

SCROLLBARS

Lets say you don't want a scroll bar in the menu window.

Furthermore the main window should have a scrollbar if needed (if the HTML document doesn't fit in the window), but if not needed - there should be no scrollbars.

Then the code should look like this:

```
<frameset cols="120,*" frameborder="0" border="0" framespacing="0">
<frame src="menu.htm" name="menu" noresize scrolling=no>
<frame src="frontf.htm" name="main" noresize scrolling=auto>
</frameset>
```


LINKS WITHIN

If you have an HTML document with a hyperlink on the text "Analysis" for instance, that links to a page called "analysis.htm" then it appears in the document as:

Jump to the Analysis of the project

Now if the link was in the menu window of our example, and we wanted it to load a page in the main window, the HTML code would be:

Jump to the Analysis of the project

We simply added the parameter target="main" to the <a href> tag.

Now the link will be opened in the main frame window instead of the menu frame window where the link itself is located.

Four target names are reserved, and will be interpreted by the browser in this way:

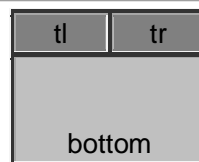
- `_blank` loads the page into a new browser window
- `_self` loads the page into the current window.
- `_parent` loads the page into the frame that is superior to the frame the hyperlink is in.
- `_top` cancels all frames, loads in full browser window.

EXAMPLES

On this page you can see examples of different framesets.



```
<frameset rows="16%,84%">  
<frame src="top.htm" name="top">  
<frame src="bottom.htm" name="bottom">  
</frameset>
```



```
<frameset rows="16%,84%">
<frameset cols="50%,50%">
<frame src="tl.htm" name="tl">
<frame src="tr.htm" name="tr">
</frameset>
<frame src="bottom.htm" name="bottom">
</frameset>
```



```
<frameset rows="16%,84%">
<frame src="top.htm" name="top">
<frameset cols="50%,50%">
<frame src="left.htm" name="left">
<frame src="right.htm" name="right">
</frameset>
</frameset>
```



```
<frameset rows="50%,50%" cols="50%,50%">
<frame src="topleft.htm" name="topleft">
<frame src="topright.htm" name="topright">
<frame src="botleft.htm" name="botleft">
<frame src="botright.htm" name="botright">
</frameset>
```



```
<frameset rows="50%,50%" cols="50%,50%">
<frame src="topleft.htm" name="topleft">
<frame src="topright.htm" name="topright">
<frame src="botleft.htm" name="botleft">
<frameset rows="50%,50%">
```

```

<frameset cols="50%,50%">
<frame src="brtl.htm" name="brtl">
<frame src="brtr.htm" name="brtr">
</frameset>
<frame src="botrbot.htm" name="botrbot">
</frameset>
</frameset>

```



```

<frameset rows="240,240" cols="320,320">
<frame src="topleft.htm" name="topleft">
<frame src="topright.htm" name="topright">
<frame src="botleft.htm" name="botleft">
<frame src="botright.htm" name="botright">
</frameset>

```



```

<frameset rows="50%,*" cols="320,*">
<frame src="topleft.htm" name="topleft">
<frame src="topright.htm" name="topright">
<frame src="botleft.htm" name="botleft">
<frame src="botright.htm" name="botright">
</frameset>

```

HTML FORMS

A form is simply an area that can contain form fields.

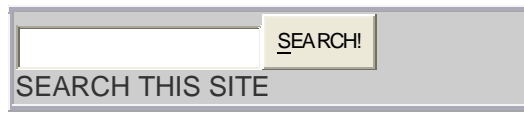
Form fields are objects that allow the visitor to enter information - for example text boxes, drop-down menus or radio buttons.

When the visitor clicks a submit button, the content of the form is usually sent to a program that runs on the server. However, there are exceptions.

Javascript is sometimes used to create magic with form fields. An example could be when turning options in a drop-down menu into normal links

EXAMPLES

A typical form example would be a search engine.



A search form with a text input field, a "SEARCH!" button, and the text "SEARCH THIS SITE".

This is what happens when the form is submitted:

- The search words are sent to a program on the server.
- The program will search a database for matches.
- The program creates a webpage with the results.
- The results webpage is sent back to the visitor.

Another example would be a logon page.



A logon form for "FREE WEB-EMAIL AT ECHOECHO.COM". It includes fields for "Username:" and "Password:", a "Log In" button, and links for "NEW USERS: [SIGN UP HERE!](#)" and "[FORGOT YOUR PASSWORD?](#)".

This is what happens when the form is submitted:

- The ID and password are sent to a program on the server.
- The program will search a database for valid entries.
- If the entry is valid the visitor is sent to the protected page.
- If the entry is invalid the visitor is sent to a "failure" page.

Both examples send the contents of the form fields to programs running on the server.

CGI SCRIPTS

When your form is submitted you need a program that can receive the information and do something with it.

Such programs are sometimes referred to as: CGI programs.

CGI stands for Common Gateway Interface, which is computer latin for a program that translates information.

This translation is necessary because the server might be a UNIX machine while the visitor might be sending information from a Windows platform.

Windows and UNIX handle information differently - so if there were no CGI, then UNIX machines could only communicate with other UNIX machines etc. and that is pretty far from the basic idea of the world wide web.

THE FORM TAG

When a form is submitted, all fields on the form are being sent.

The <form> tag tells the browser where the form starts and ends. You can add all kinds of HTML tags between the <form> and </form> tags.

This means that a form can easily include a table or an image along with the form fields mentioned on the [next page](#).

Look at this example:

```
<html>
<head>
<title>My Page</title>
</head>

<body>
<!-- Here goes HTML -->
<form>
<!-- Here goes form fields and HTML -->
</form>
<!-- Here goes HTML -->
</body>
</html>
```

Note:

Unlike a table, forms are not visible on the page.

The form in our example is useless for two obvious reasons:

- First it contains no form fields. It is simply comparable to a blank sheet of paper.
- Second, it does not contain a recipient for the form once it is submitted.

To let the browser know where to send the content we add these properties to the <form> tag:

- action=address
- method=post or method=get

The address is the url of the cgi script the content should be sent to. The post and get methods are simply two different methods for submitting data to the script.

If you are using a pre-programmed script (which we assume here) it is not important to

understand the difference between get and post.
In the description of the script you are using it will be made clear whether the scripts should be addressed using one method or the other.

Below is an example of a typical form tag, with both action and method specified.

```
<html>
<head>
<title>My Page</title>
</head>

<body>
<!-- Here goes HTML -->
<form method="post" action="http://www.echoecho.com/cgi-
bin/formmail.cgi">
<!-- Here goes form fields and HTML -->
</form>
<!-- Here goes HTML -->
</body>
</html>
```

TEXT FIELD

Text fields are one line areas that allow the user to input text.

If you want several lines you should use a [text area](#) instead.

SETTINGS:

Below is a listing of valid settings for text fields:

HTML	EXPLANATION	EXAMPLE
text	One line text field	
size=	Characters shown.	
maxlength=	Max characters allowed.	
name=	Name of the field.	<input type="text"/>
value=	Initial value in the field.	
align=	Alignment of the field.	
tabindex=	Tab order of the field.	

The size option defines the width of the field. That is how many visible characters it can contain.

The maxlength option defines the maximum length of the field. That is how many characters can be entered in the field.

If you do not specify a maxlength, the visitor can easily enter more characters than are visible in the field at one time.

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The value setting defines what will appear in the box as the default value.

The align setting defines how the field is aligned.

Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM. The alignments are explained in the image section. You can learn about the different alignments [here](#).


The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<br><br>
<input type="text" size="25" value="Enter your name here!">
<br><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:




PASSWORD FIELD

Password fields are similar to text fields.

The difference is that what is entered into a password field shows up as dots on the screen. This is, of course, to prevent others from reading the password on the screen.

SETTINGS:

Below is a listing of valid settings for password fields:

HTML	EXPLANATION	EXAMPLE
password	One line password field	
size=	Characters shown.	
maxlength=	Max characters allowed.	
name=	Name of the field.	
value=	Initial value in the field.	
align=	Alignment of the field.	
tabindex=	Tab order of the field.	

The size option defines the width of the field. That is how many visible characters it can contain.

The maxlength option defines the maximum length of the field. That is how many characters can be entered in the field.

If you do not specify a maxlength, the visitor can easily enter more characters than are visible in the field at one time.

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The value setting defines what will appear in the box as the default value.

The align setting defines how the field is aligned.

Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM. The alignments are explained in the image section. You can learn about the different alignments [here](#).

The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
Enter Password : <input type="password" size="25">
<br><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:



Enter Password :

HIDDEN FIELD

Hidden fields are similar to text fields, with one very important difference!

The difference is that the hidden field does not show on the page. Therefore the visitor can't type anything into a hidden field, which leads to the purpose of the field:

To submit information that is not entered by the visitor.

SETTINGS:

Below is a listing of valid settings for hidden fields:

HTML	EXPLANATION	EXAMPLE
hidden	Hidden field	
name=	Name of the field.	
value=	Value of the field.	

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The value setting defines what will be sent once the form is submitted.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<input type="text" size="25" value="Enter your name here!">
<input type="hidden" name="Language" value="English">
<br><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:

The hidden field does not show, but still, when the form is submitted the hidden field is sent with it.

In this example the hidden field would tell the program that handles the form, that the users preferred language is English.

TEXT AREA

Text areas are text fields that can span several lines.

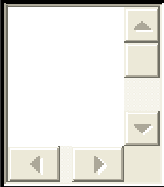

Unlike most other form fields, text areas are not defined with an `<input>` tag.

Instead you enter a `<textarea>` tag where you want the text area to start and a closing `</textarea>` tag where you want the area to end.

Everything written between these tags will be presented in the text area box.

SETTINGS:

Below is a listing of valid settings for text areas:

HTML	EXPLANATION	EXAMPLE
<code>textarea</code>	Text area - several lines	
<code>rows=</code>	Rows in the field.	
<code>cols=</code>	Columns in the field.	
<code>name=</code>	Name of the field.	
<code>tabindex=</code>	Tab order of the field.	
<code>wrap=</code>		
<code>off</code>	Turns off linebreaking	
<code>virtual</code>	Shows linebreaking, but sends text as entered.	
<code>physical</code>	Inserts linebreaks when needed and even sends it.	

The `cols` and `rows` settings are straightforward and simple. They specify how many columns and rows you want in your text area.

The `name` setting adds an internal name to the field so the program that handles the form can identify the fields.

The `tabindex` setting defines in which order the different fields should be activated when the visitor clicks the tab key.

The wrap options are the most tricky part of text areas.
If you turn wrap off the text is handled as one long sequence of text without linebreaks.
If you set it to virtual the text appears on your page as if it recognized linebreaks - but when the form is submitted the linebreaks are turned off.
If you set it to physical the text is submitted exactly as it appears on the screen - linebreaks included.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>


<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
This is outside the area<br><br>
<textarea cols="40" rows="5" name="myname">
Now we are inside the area - which is nice.
</textarea>
<br><br>
And now we are outside the area again.
</div>
</form>
</body>
</html>
```

CHECK BOX

Check boxes are used when you want to let the visitor select one or more options from a set of alternatives. If only one option is to be selected at a time you should use [radio buttons](#) instead.

SETTINGS:

Below is a listing of valid settings for check boxes:

HTML	EXPLANATION	EXAMPLE
checkbox	Choose one or more options	
name=	Name of the field.	
value=	Value that is submitted if checked.	
align=	Alignment of the field.	
tabindex=	Tab order of the field.	
checked	Default check this field.	

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The value setting defines what will be submitted if checked.

The align setting defines how the field is aligned.

Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.

The alignments are explained in the image section. You can learn about the different alignments [here](#).

The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center"><br>
<input type="checkbox" name="option1" value="Milk"> Milk<br>
<input type="checkbox" name="option2" value="Butter" checked>
Butter<br>
<input type="checkbox" name="option3" value="Cheese"> Cheese<br>
<br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:




☐ Milk
☒ Butter
☐ Cheese

RADIO BUTTON

Radio buttons are used when you want to let the visitor select one - and just one - option from a set of alternatives. If more options are to be allowed at the same time you should use [check boxes](#) instead.

SETTINGS:

Below is a listing of valid settings for radio buttons:

HTML	EXPLANATION	EXAMPLE
radio	Choose one - and only one - option	
name=	Name of the group.	
value=	Value that is submitted if checked.	
align=	Alignment of the field.	
tabindex=	Tab order of the field.	
checked	Default check this field.	

The name setting tells which group of radio buttons the field belongs to. When you select one button, all other buttons in the same group are unselected.
If you couldn't define which group the current button belongs to, you could only have one group of radio buttons on each page.

The value setting defines what will be submitted if checked.

The align setting defines how the field is aligned.
Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.

The alignments are explained in the image section. You can learn about the different alignments [here](#).

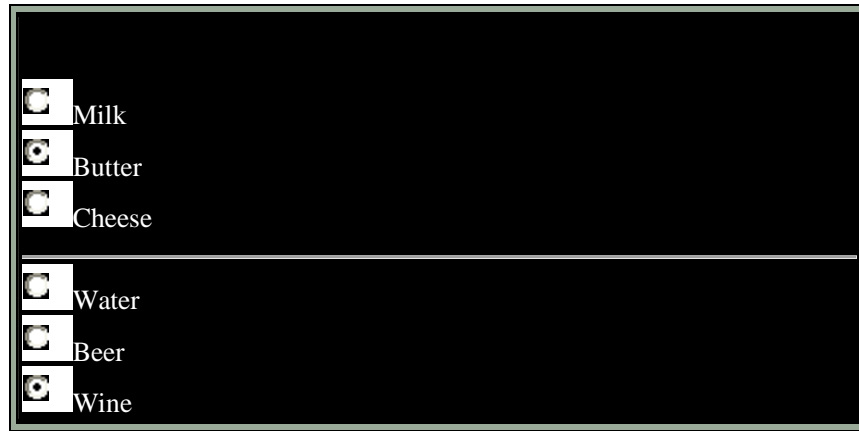
The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center"><br>
<input type="radio" name="group1" value="Milk"> Milk<br>
<input type="radio" name="group1" value="Butter" checked> Butter<br>
<input type="radio" name="group1" value="Cheese"> Cheese
<hr>
<input type="radio" name="group2" value="Water"> Water<br>
<input type="radio" name="group2" value="Beer"> Beer<br>
<input type="radio" name="group2" value="Wine" checked> Wine<br>
</div>
</form>
</body>
</html>
```

And the resulting output:



DROP DOWN MENU

Drop-down menus are probably the most flexible objects you can add to your forms.

Depending on your settings, drop-down menus can serve the same purpose as radio buttons (one selection only) or check boxes (multiple selections allowed).

The advantage of a drop-down menu, compared to radio buttons or check boxes, is that it takes up less space.

But that is also a disadvantage, because people can't see all options in the menu right away.

There is a workaround for this - with the size setting, you can customize the menu so it shows more than just one option at a time, but when you do that - you also lose the advantage of taking up less space.

So whatever you decide - there is always a bonus and a price to pay.

Sometimes you may want to replace text fields with drop-down menus. This might be because selecting from a menu is easier than typing. But it could also be because the script that handles the form can't interpret just any text entry.

For example, you will often be asked to choose your state from a drop-down menu. This might be because picking it from the menu is easier than typing the name of the state.

Along the same line, you may often be asked to enter the 2 letter initials of your state from a drop-down menu as well.


This could prevent confusion for the script that handles the form input. If, say, the script was programmed to only accept capital letters, then a drop-down menu would secure that no invalid entries were made.

Another typical example would be replacing links with drop-down menus.

This can be done with javascript. If you're not into programming you can easily create a drop-down link menu with our [online tool](#).

SETTINGS:

Below is a listing of valid settings for drop-down menus:

HTML	EXPLANATION	EXAMPLE
<code>select</code>	Drop-down menu	
<code>name=</code>	Name of the field.	
<code>size=</code>	Visible items in list.	
<code>multiple=</code>	Allows multiple choices if yes.	
<code>option</code>	Individual items in the menu.	
<code>selected</code>	Default select the item.	
<code>value=</code>	Value to send if selected.	

Drop-down menus combine `<select>` and `<option>`.
Both tags have an opening and a closing tag.

A typical example of the syntax would be:

```
<select>
<option>Milk</option>
<option>Coffee</option>
<option>Tea</option>
</select>
```

The `<select>` tag defines the menu.

The name setting adds an internal name to the field so the program that handles the form can identify the fields.

The size option defines how many items should be visible at a time. Default is one item.

The multiple setting will allow for multiple selections if present.
The `<option>` tag defines the single items in the menu.

The value setting defines what will be submitted if the item is selected. This is not always the same as what it says in the menu. If our field was defined this way:

```
<option value="ID">Idaho</option>
```

then, in the menu it would say "Idaho" but when the form was submitted the abbreviated "ID" would actually be sent.

You can force an item to be default selected by adding the selected option: `<option selected>`

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
```

```
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<select name="mydropdown">
<option value="Milk">Fresh Milk</option>
<option value="Cheese">Old Cheese</option>
<option value="Bread">Hot Bread</option>
</select>
</div>
</form>
</body>
</html>
```

And the resulting output from it:



SUBMIT BUTTON

When a visitor clicks a submit button, the form is sent to the address specified in the action setting of the <form> tag.

SETTINGS:

Below is a listing of valid settings for submit buttons:

HTML	EXPLANATION	EXAMPLE
submit	Submit button	
name=	Name of the button.	
value=	Text written on the button.	
align=	Alignment of the button.	
tabindex=	Tab order of the button.	

The name setting adds an internal name to the button so the program that handles the form doesn't confuse the button with the other fields.

The value setting defines what is written on the button.

The align setting defines how the button is aligned.
Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.
The alignments are explained in the image section.
You can learn about the different alignments [here](#).

The tabindex setting defines in which order the different fields should be activated

when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<br><br>
<input type="text" size="25" value="Enter your name here!">
<br><input type="submit" value="Send me your name!"><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:



RESET BUTTON

When a visitor clicks a reset button, the entries are reset to the default values.

SETTINGS:

Below is a listing of valid settings for reset buttons:

HTML	EXPLANATION	EXAMPLE
reset	Reset button	
name=	Name of the button.	
value=	Text written on the button.	
align=	Alignment of the button.	
tabindex=	Tab order of the button.	

The name setting adds an internal name to the button so the program that handles the form doesn't confuse the button with the other fields.

The value setting defines what is written on the button.

The align setting defines how the button is aligned.

Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.

The alignments are explained in the image section.

You can learn about the different alignments [here](#).

The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<br><br>
<input type="text" size="25" value="Enter your name here!">
<br><input type="submit" value="Send me your name!"> <input
type="reset" value="Reset!"><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:




IMAGE BUTTON

Image buttons have the same effect as submit buttons. When a visitor clicks an image button the form is sent to the address specified in the action setting of the <form> tag.

Since visitors aren't always perfectionists you might consider adding a [javascript validation](#) of the content before it is actually sent.

SETTINGS:

Below is a listing of valid settings for image buttons:

HTML	EXPLANATION	EXAMPLE
image	Submit button	
name=	Name of the image.	
src=	Url of the image.	
align=	Alignment of the image.	
border=	Border width around the image.	

width=	Width of the image.	
height=	Height of the image.	
vspace=	Spacing over and under image.	
hspace=	Spacing left and right of image.	
tabindex=	Tab order of the image.	

The name setting adds an internal name to the image button so the program that handles the form doesn't confuse it with the other fields.

The src setting defines the URL of the image.

The align setting defines how the image is aligned.
Valid entries are: TOP, MIDDLE, BOTTOM, RIGHT, LEFT, TEXTTOP, BASELINE, ABSMIDDLE, ABSBOTTOM.
The alignments are explained in the image section.
You can learn about the different alignments [here](#).

The border setting defines the width (in pixels) of the border around the image.

The width setting defines the width of the image.

The height setting defines the height of the image.

The vspace setting defines the spacing over and under the image (in pixels).

The hspace setting defines the spacing to the left and right of the image (in pixels).

The tabindex setting defines in which order the different fields should be activated when the visitor clicks the tab key.

AN EXAMPLE:

Look at this HTML example:

```
<html>
<head>
<title>My Page</title>
</head>
<body>
<form name="myform"
action="http://www.mydomain.com/myformhandler.cgi"
method="POST">
<div align="center">
<br><br>
<input type="text" size="25" value="Enter your name here!">
<br><input type="image" src="rainbow.gif" name="image" width="60"
height="60"><br>
</div>
</form>
</body>
</html>
```

And the resulting output from it:

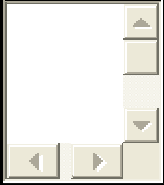







Enter your name here!



QUICK REFERENCE

If you're not familiar with the form tags you can learn in detail about each tag listed in the left menu. Otherwise - use this quick reference for an easy overview of form tags and properties.

HTML	EXPLANATION	EXAMPLE
textarea rows= cols= name= wrap= off virtual physical	Text area - several lines Rows in the field. Columns in the field. Name of the field. Control linebreaks. Turns off linebreaks. Shows linebreaks, but sends text as entered. Inserts linebreaks when needed and even sends it.	
text size= maxlength= name= value=	One line text field Characters shown. Max characters allowed. Name of the field. Initial value in the field.	
password size= maxlength= name= value=	Password field. Characters shown. Characters allowed to enter. Name of the field. Initial value in the field.	
checkbox name= value=	Choose one or more options Name of the field. Initial value in the field.	
radio name= value=	Choose only one option Name of the field. Initial value in the field.	
select name= size= multiple=	Drop-down menu Name of the field. Number of items in list. Allow multiple choice if yes.	
option selected value=	Individual items in the menu. Make an item default. Value to send if selected.	
hidden	Does not show on the form.	

name=	Name of the field.	
value=	Value to send.	
reset	Button to reset all fields	
name=	Name of the button.	<input type="button" value="Reset"/>
value=	Text shown on the button.	
submit	Button to submit the form	
name=	Name of the button.	<input type="button" value="Submit"/>
value=	Text shown on the button.	
image	Image behaving as button	<input type="image" value=""/>
name=	Name of the image.	

Note: This is a quick reference showing the most common settings for each field.