HTML Beginner's Guide
htmldog.com

This **HTML Beginner's Guide** assumes that you have no previous knowledge of HTML or CSS.

It should be quite easy to follow if you work through each step, which are all brought together at the end, before moving on to the CSS Beginner's Guide.

The thing to keep in mind is that HTML and CSS are all about separating the **content** (HTML) and the **presentation** (CSS). HTML is nothing more than fancy structured content and the formatting of that content will come later when we tackle CSS.

If you have looked at other HTML tutorials, you may find that they mention certain things that HTML Dog does not. This is because many methods are obsolete, non-standard or just plain bad practice. Getting into the frame of mind of doing things the RIGHT way from the start will turn in to much better results in the end.

Getting Started

Most of the stuff on the web is no different than the stuff on your computer - it's just a whole load of files sorted into a whole load of directories.

HTML files are nothing more than simple text files, so to start writing in HTML, you need nothing more than a simple text editor. **Notepad** is a common example (on Windows this is usually found under the Programs > Accessories menu).

Type this in to your text editor:

```
This is my first web page
```

Now create a folder called 'html' and save the file as 'myfirstpage.html' (it is important that the extension '.html' be specified - some text editors, such as Notepad, will automatically save it as '.txt' otherwise).

To look at HTML files, they don't even need to be on the web. Open **Internet Explorer**, or any other web browser and in the address bar, where you usually type web addresses, type in the location of the file you just saved (for example, 'c:\html\myfirstpage.html') and hit return.

Pow. There it is. Your first web page. How exciting. And all it took was a few typed words.

Note

We've said here to use a basic text-editor, such as Notepad, but you may be tempted to use a dedicated software program such as **Macromedia Dreamweaver** or **Microsoft Frontpage**.

You should be very careful when using these programs, especially if you are a beginner as they often throw in unnecessary or non-standard code.

If you're serious about learning HTML, you should read through a tutorial such as this first, so that you at least have a basic understanding of what is going on.

Software programs such as these will never give you the same control over a web page as coding by hand.

Tags, Attributes and Elements

Although the basics of HTML are plain text, we need a bit more to make it a valid HTML document.

Tags

The basic structure of an HTML document includes **tags**, which surround content and apply meaning to it.

Change your document so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<body>
This is my first web page
</body>
</html>
```

Now save the document again, go back to the web browser and select 'refresh' (which will reload the page).

The appearance of the page will not have changed at all, but the purpose of HTML is to apply meaning, not presentation, and this example has now defined some fundamental elements of a web page.

The first line on the top that starts **<!DOCTYPE**... is to let the browser know that you know what the hell you're doing. You may think that you don't actually know what you're doing yet, but it's important to stick this in. If you don't, browsers will switch into 'quirks mode' and act in a very peculiar way. Don't worry about this just yet, you can learn more about 'document types' in the HTML Advanced Guide if you really want to. For the moment, just remember to shove this line at the top of your web pages and you're laughin'.

To get back to the point, `<html>` is the **opening tag** that kicks things off and tells the browser that everything between that and the `</html>` **closing tag** is an HTML document. The stuff between `<body>` and `</body>` is the main content of the document that will appear in the browser window.
Closing tags

The `</body>` and `</html>` close their respective tags. ALL HTML tags should be closed. Although older versions of HTML lazily allowed some tags not to be closed, latest standards require all tags to be closed. This is a good habit to get into anyway.

Not all tags have closing tags like this (`<html></html>`) some tags, which do not wrap around content will close themselves. The line-break tag for example, looks like this : `<br />`**.** We will come across these examples later. All you need to remember is that all tags must be closed and most (those with content between them) are in the format of opening tag - content - closing tag.

Attributes

Tags can also have **attributes**, which are extra bits of information. Attributes appear inside the opening tag and their value is always inside quotation marks. They look something like <tag attribute="value">Margarine</tag>. We will come across tags with attributes later.

Elements

Tags tend not to do much more than mark the beginning and end of an **element**. Elements are the bits that make up web pages. You would say, for example, that everything that is in-between and includes the `<body>` and `</body>` tags is the body element. As another example, whereas '`<title>`' and '`</title>`' are tags, '`<title>Rumple Stiltskin</title>`' is a title element.

Page Titles

All HTML pages should have a page **title**. To add a title to your page, change your code so that it looks like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
</head>
<body>
This is my first web page
</body>
</html>
```

We have added two new elements here, that start with the **head** tag and the **title** tag (and see how both of these close).

The head element (that which starts with the <head> opening tag and ends with the </head> tag) appears before the body element (starting with <body> and ending with </body>) and contains information that will load before the body information. The information in the head element does not appear in the browser window. We will see later on that other elements can appear inside the head element, but the most important of them is the **title** element.

If you look at this document in the browser (save and refresh as before), you will see that 'My first web page' will appear on the title bar of the window (not the actual canvas area). The text that you put in between the title tags has become the title of the document (surprise!). If you were to add this page to your 'favorites', you would see that the title is also used there.

Paragraphs

Now that you have the basic structure of an HTML document, you can mess about with the content a bit.

Go back to notepad and add another line to your page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
</head>
<body>This is my first web page How exciting</body>
</html>
```

Look at the document in your browser.

You might have expected your document to appear as you typed it, on two lines, but instead you should see something like:

```
This is my first web page How exciting.
```

This is because web browsers don't usually take any notice of what line your code is on. It also doesn't take any notice of spaces (you would get the same result if you typed 'This is my first web page How exciting').

If you want text to appear on different lines, you need to explicitly state that.

Change your two lines of content so that they look like this:

```
<p>This is my first web page</p>
<p>How exciting</p>
```

The `p` tag is for '**paragraph**'.

Look at the results of this. The two lines will now appear on two lines.

The HTML content should be seen just like a book - with paragraphs where appropriate.

Emphasis

You can emphasise text in a paragraph using `em` and `strong`. These are two ways of doing pretty much the same thing, although traditionally, browsers display `em` in italics and `strong` in bold.

```
<p>Yes, that <em>is</em> what I said. How <strong>very</strong>
exciting.</p>
```

Line breaks

The line-break tag can also be used to separate lines like this:

```
This is my first web page<br />
How exciting
```

However, this method is over-used and shouldn't be used if two blocks of text are intended to be separate from one another.

(Because nothing appears between the line-break tag, there is no closing tag and it closes itself with a '/' after the 'br').

Headings

The p tag is just the start of text formatting.

If you have documents with genuine **headings**, then there are HTML tags specifically designed just for them.

They are **h1**, **h2**, **h3**, **h4**, **h5** and **h6**, **h1** being the almighty emperor of headings and **h6** being the lowest pleb.

Change your code to the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
</head>
<body>
<h1>My first web page</h1>
<h2>What this is</h2>
<p>A simple page put together using HTML</p>
<h2>Why this is</h2>
<p>To learn HTML</p>
</body>
</html>
```

Note that the **h1** tag is only used once - it is supposed to be the main heading of the page and shouldn't be used multiple times.

h2 to **h6** however, can be used as often as is desired, but they should always be used in order, as they were intended. For example, an **h4** should be a sub-heading of an **h3**, which should be a sub-heading of an **h2**.

Lists

There are three types of list; **unordered lists**, **ordered lists** and **definition lists**. We will look at the first two here, and definition lists in the HTML Intermediate Guide.

Unordered lists and ordered lists work the same way, except that the former is used for non-sequential lists with list items usually preceded by bullets and the latter is for sequential lists, which are normally represented by incremental numbers.

The **ul** tag is used to define unordered lists and the **ol** tag is used to define ordered lists. Inside the lists, the **li** tag is used to define each list item.

Change your code to the following:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
</head>
<body>
<h1>My first web page</h1>
<h2>What this is</h2>
<p>A simple page put together using HTML</p>
<h2>Why this is</h2>
<ul>
   <li>To learn HTML</li>
   <li>To show off</li>
   <li>Because I've fallen in love with my computer and want to
   give her some HTML loving.</li>
</ul>
</body>
</html>
```

If you look at this in your browser, you will see a bulleted list. Simply change the ul tags to ol and you will see that the list will become numbered.

Lists can also be included in lists to form a structured hierarchy of items.

Replace the above list code with the following:

```
<ul>
   <li>To learn HTML</li>
   <li>To show off
      <ol>
      <li>To my boss</li>
      <li>To my friends</li>
      <li>To my cat</li>
      <li>To the little talking duck in my brain</li>
      </ol>
   </li>
   <li>Because I've fallen in love with my computer and want to
   give her some HTML loving.</li>
</ul>
```

Ay vwah lah. A list within a list. And you could put another list within that. And another within that. And so on and so forth.

Links

So far you've been making a stand-alone web page, which is all very well and nice, but what makes the internet so special is that it all **links** together.

The 'H' and 'T' in 'HTML' stand for '**hypertext**', which basically means a system of linked text.

An **anchor** tag (**a**) is used to define a link, but you also need to add something to the anchor tag - the **destination** of the link.

Add this to your document:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
</head>
<body>
<h1>My first web page</h1>
<h2>What this is</h2>
<p>A simple page put together using HTML</p>
<h2>Why this is</h2>
<p>To learn HTML</p>
<h2>Where to find the tutorial</h2>
<p><a href="http://www.htmldog.com">HTML Dog</a></p>
</body>
</html>
```

The destination of the link is defined in the **href** attribute of the tag. The link can be **absolute**, such as 'http://www.htmldog.com', or it can be **relative** to the current page.

So if, for example, you had another file called 'flyingmoss.html' then the line of code would simply be **The miracle of moss in flight** or something like this.

A link does not have to link to another HTML file, it can link to any file anywhere on the web.

A link can also send a user to another part of the same page they are on. You can add an **id** attribute to just about any tag, for example **<h2 id="moss">Moss</h2>**, and then link to it by using something like this: **Go to moss.** Selecting this link will scroll the page straight to the element with that id.

Note

The a tag allows you to open the link in a **newly spawned window**, rather than replacing the web page the user is on, which at first thought may sound like a good idea as it doesn't take the user away from your site.

There are a number of reasons why you shouldn't do this however.

From a **usability** point of view, this method **breaks navigation**. The most commonly used navigation tool on a browser is the '**back**' button. Opening a new window disables this function.

On a wider, more general usability point, users do not want new windows to be popping up all over the place. If they want to open a link in a new window then they can choose to do so themselves.

Images

Things might seem a little bland and boring with all of this text formatting. Of course, the web is not just about text, it is multi-media and the most common form of media is the **image**.

The **img** tag is used to put an image in an HTML document and it looks like this:

```
<img src="http://www.htmldog.com/images/logo.gif" width="157"
height="70" alt="HTML Dog logo" />
```

The **src** attribute tells the browser where to find the image. Like the **a** tag, this can be absolute, as the above example demonstrates, but is usually relative. For example, if you create your own image and save it as 'alienpie.jpg' in a directory called 'images' then the code would be

```
<img src="images/alienpie.jpg"...
```

The **width** and **height** attributes are necessary because if they are excluded, the browser will tend to calculate the size as the image loads, instead of when the page loads, which means that the layout of the document may jump around while the page is loading.

The **alt** attribute is the **alternative description**. This is used for people who cannot or choose not to view images. This is a **requirement** in the latest versions of HTML.

Note that, like the **br** tag, because the **img** tag does not have a closing tag, it closes itself, ending with '**/>**'

Note

The construction of images for the web is a little outside of the remit of this website, but it is worth noting a few things...

The most commonly used file formats used for images are **GIFs** and **JPEGs**. They are both compressed formats, and have very different uses.

GIFs can have no more than 256 colours, but they maintain the colours of the original image. The lower the number of colors you have in the image, the lower the file size will be.

GIFS SHOULD BE USED FOR IMAGES WITH SOLID COLOURS.

JPEGs on the other hand use a mathematical algorithm to compress the image and will distort the original slightly. The lower the compression, the higher the file size, but the clearer the image.

JPEGS SHOULD BE USED FOR IMAGES SUCH AS PHOTOGRAPHS.

Images are perhaps the largest files a new web designer will be handling. It is a common mistake to be oblivious to the file size of images, which can be extremely large. Web pages should download as quickly as possible, and if you keep in mind that most people use modems that download at **less than 7Kb a second** (realistically it is less than 5Kb), you can see how a large file will greatly slow down the download time of a full page.

You need to strike a balance between image quality and image size. Most modern image manipulation programs allow you to compress images and the best way to figure out what is best suited for yourself is trial and error.

Tables

Across the worldwide web, HTML **tables** are used and abused to **layout** pages. We will come across how to layout a page without tables, in the CSS Advanced Guide. The correct use for tables is to do exactly what you would expect a table to do - to **layout tabular data**.

There are a number of tags used in tables, and to fully get to grips with how they work is probably the most difficult area of this HTML Beginners Guide.

Copy the following code into the body of your document and then we will go through what each tag is doing:

```
<table>
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
        <td>Row 1, cell 3</td>
    </tr>
    <tr>
        <td>Row 2, cell 1</td>
        <td>Row 2, cell 2</td>
        <td>Row 2, cell 3</td>
    </tr>
    <tr>
        <td>Row 3, cell 1</td>
        <td>Row 3, cell 2</td>
        <td>Row 3, cell 3</td>
    </tr>
    <tr>
        <td>Row 4, cell 1</td>
        <td>Row 4, cell 2</td>
        <td>Row 4, cell 3</td>
    </tr>
</table>
```

The **table** element defines the table.

The **tr** element defines a table **row**.

The **td** element defines a **data cell**. These must be enclosed in **tr** tags, as shown above.

If you imagine a 3x4 table, which is 12 cells, there should be four **tr** elements to define the rows and three **td** elements within each of the rows, making a total of 12 **td** elements.

DANA'S TABLE WITH ADDITIONAL ATTRIBUTES

```
<table width="800" border="0" cellspacing="0" cellpadding="0">
      <tr>
            <td valign="top"> </td>
            <td valign="top"> </td>
      </tr>
      <tr>
            <td valign="top"> </td>
            <td valign="top"> </td>
      </tr>
</table>
```

Forms

Forms can be used to send data across the web and are often used as **contact forms** to convert information inputted by a user into an email, such as the one used on this website.

On their own, forms are useless. They need to be hooked up to a program that will process the data inputted by the user. These take all manner of guises and are outside of the remit of this website. If you use an internet service provider to host your HTML, they will be able to help you with this and will probably have clear and simple instructions on how, for example, to make a form-to-email form work.

The tags used in the actual HTML of forms are **`form`**, **`input`**, **`textarea`**, **`select`** and **`option`**.

`form` defines the form and within this tag, there is one required **`action`** attribute which tells the form where its contents will be sent to when it is submitted.

The optional **`method`** attribute tells the form how the data in it is going to be sent and it can have the value **`get`** (which is default) or **`post`**. This is commonly used, and often set to **`post`** which hides the information (**`get`** latches the information onto the URL).

So a form element will look something like this:

```
<form action="processingscript.php" method="post">
</form>
```

The **`input`** tag is the daddy of the form world. It can take ten forms, outlined below:

_ `<input type="text" />` is a standard **textbox**. This can also have a **value** attribute, which sets the text in the textbox.
_ `<input type="password" />` is the same as the textbox, but will display asterisks instead of the actual characters that the user types.
_ `<input type="checkbox" />` is a **checkbox**, which can be toggled on and off by the user. This can also have a **checked** attribute, which would be used in the format **`<input type="checkbox" checked="checked" />`**.
_ `<input type="radio" />` is similar to a checkbox, but the user can only select one **radio button** in a group. This can also have a checked attribute, used in the same way as the checkbox.
_ `<input type="file" />` is an area that shows the **files** on your computer, like you see when you open or save a document in most programs.

_ `<input type="submit" />` is a button that when selected will **submit** the form. You can control the text that appears on the submit button (as you can with `button` and `reset` types - see below) with the `value` attribute, for example `<input type="submit" value="Ooo. Look. Text on a button. Wow" />`.
_ `<input type="image" />` is an **image** that when selected will submit the form. This also requires a `src` attribute, like the `img` tag.
_ `<input type="button" />` is a **button** that will not do anything without extra code added.
_ `<input type="reset" />` is a button that when selected will **reset** the form fields.
_ `<input type="hidden" />` is a field that will not be displayed and is used to pass information such as the page name that the user is on or the email address that the form should be posted to.

Note that the `input` tag closes itself with a '`/>`' at the end.

A **textarea** is, basically, a large textbox. It requires a `rows` and `cols` attribute and is used like this:

```
<textarea rows="5" cols="20">A big load of text here</textarea>
```

The select tag works with the `option` tag to make **drop-down select boxes**.

They work like this:

```
<select>
<option value="first option">Option 1</option>
<option value="second option">Option 2</option>
<option value="third option">Option 3</option>
</select>
```

When the form is submitted, the value of the selected option will be sent.

Similar to the `checked` attribute of checkboxes and radio buttons, an `option` tag can also have a `selected` attribute, which would be used in the format `<option value="mouse" selected="selected">Rodent</option>`.

All of the tags mentioned above will look very nice presented on the page, but if you hook up your form to a form-handling program, they will all be ignored. This is because the form fields need **names**. So to all of the fields, the attribute `name` needs to be added, for example `<input type="text" name="talkingsponge" />`

A form might look like the one below. (Note: this form will not work unless there is a 'contactus.php' file, which is stated in the action attribute of the form tag, to handle the submitted date)

```
<form action="contactus.php" method="post">
<p>Name:</p>
<p><input type="text" name="name" value="Your name" /></p>
<p>Comments: </p><p><textarea name="comments" rows="5"
cols="20">Your comments</textarea></p>
<p>Are you:</p>
<p><input type="radio" name="areyou" value="male" /> Male</p>
<p><input type="radio" name="areyou" value="female" /> Female</p>
<p><input type="submit" /></p>
<p><input type="reset" /></p></form>
```

There is a whole other level of complexity you can delve into in the HTML Advanced Guide if you are so inclined.

Putting It All Together

The following code incorporates all of the methods that have been explained in the previous pages:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>My first web page</title>
<!-- By the way, this is a comment -->
</head>
<body>
<h1>My first web page</h1>
<h2>What this is</h2>
<p>A simple page put together using HTML. <strong>A simple page
put together using HTML.</strong> A simple page put together
using HTML. A simple page put together using HTML. A simple page
put together using HTML. A simple page put together using HTML. A
simple page put together using HTML. A simple page put together
using HTML. A simple page put together using HTML.</p>
<h2>Why this is</h2>
<ul>
    <li>To learn HTML</li>
    <li>
    To show off
        <ol>
            <li>To my boss</li>
            <li>To my friends</li>
            <li>To my cat</li>
            <li>To the little talking duck in my brain</li>
        </ol>
    </li>
    <li>Because I've fallen in love with my computer and
    want to give her some HTML loving.</li>
```

```
</ul>
<h2>Where to find the tutorial</h2>
<p><a href="http://www.htmldog.com"><img
src="http://www.htmldog.com/images/logo.gif" width="157"
height="70" alt="HTML Dog logo" /></a></p>
<h3>Some random table</h3>
<table>
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
        <td>Row 1, cell 3</td>
    </tr>
    <tr>
        <td>Row 2, cell 1</td>
        <td>Row 2, cell 2</td>
        <td>Row 2, cell 3</td>
    </tr>
    <tr>
        <td>Row 3, cell 1</td>
        <td>Row 3, cell 2</td>
        <td>Row 3, cell 3</td>
    </tr>
    <tr>
        <td>Row 4, cell 1</td>
        <td>Row 4, cell 2</td>
        <td>Row 4, cell 3</td>
    </tr>
</table>
<h3>Some random form</h3>
<p><strong>Note:</strong> It looks the part, but won't do a
thing</p>
<form action="somescript.php" method="post">
<p>Name:</p>
<p><input type="text" name="name" value="Your name" /></p>
<p>Comments: </p>
<p><textarea rows="10" cols="20" name="comments">Your
comments</textarea></p>
<p>Are you:</p><p><input type="radio" name="areyou" value="male"
/> Male</p>
<p><input type="radio" name="areyou" value="female" /> Female</p>
<p><input type="submit" /></p>
<p><input type="reset" /></p>
</form>
</body>
</html>
```

There you have it. Save the file and play around with it - this is the best way to understand how everything works. Go on. Tinker.

When you're happy, you can move on to the CSS Beginner's Guide.

OPTIMIZING IMAGES

NOTE: On the web, images will always be viewed 72 DPI. It is important to think of image size as purely in terms of pixel dimensions. Keep it in mind that many users' computer screens are still set to 800 x 600 pixels, size your design and images in relation to that scale.

1. Resize the image:
    - Open the image in Photoshop
    - Under **Image**, click **Image Size**
    - Make size adjustments under Pixel Dimensions
2. Optimize the image
    - Under F**ile**, Click **Save for Web**
    - Click **Optimized** tab
    - Choose between JPEG or GIF
        - JPEG – use for a photographic image.
        - GIF – use for color graphic (think comics)
    - Adjust **Quality** to lowest point without compromising image quality.
    - Hit **Save**

KEEP IN MIND: **IMAGES ARE COSTLY** TO YOUR WEBSITE IN TERMS OF FILE SIZE/DOWNLOAD TIME. THE LARGER THE FILE, THE LONGER TO LOAD, THE LESS PATIENT THE END USER!

ORGANIZING FILES

As soon as you start to work on your images, you should save them in a General Site folder, and within that site folder, inside an image folder. When creating a website it is imperative that you are organized. Below is a typical "Site". A Site folder, this one called 'Summer', holds one image folder and a number of HTML files. These HTML files will link to one another and link to images in the images folder. If the files get moved out of the site folder or images folder, your links will break and your site will not work right.