Web Site Designing

By Anand Vishwanathan

CONTENTS

THE INTERNET

What is the Internet?

The Internet is a network of many smaller networks that exist all over the world. It contains millions and millions of pages of information waiting for you to access. All the computers connected to the Internet share equal rights i.e. there is no central controller.

What are the various Applications of the Internet?

The main applications of the Internet are as follows:

SERVICE	PROTOCOL USED
HTTP	Web Browsing
SMTP and POP	E-mail
FTP	Transferring files across the Internet
Telnet	To log on and work on another system as though you are sitting just next to it.

Web Browsing / Surfing

One of the main applications of the Internet is Browsing or Surfing, as it is more popularly known as. This helps one to get information from across the globe.

In order to access a Web Site, you need a software called a Web Browser and the address of the Site. These addresses are also referred to as **URLs**. Some examples include http://www.hotmail.com, http://www.harward.edu, etc. If you look closely at these names you will find certain similarities.

Servers, Protocols and Pathnames

In order that so many computers can be networked and communicate with each other, they should use a common **protocol** or a language. A protocol is a standard way to send and receive information.

Each file on the Internet has a unique address called a **Uniform Resource Locator (URL)**. It is determined by the type of protocol, name of the server on the Internet and location of the file on the Internet. Let us analyze the URL http://www.hotmail.com. The http stands for Hypertext Transfer Protocol. The www indicates World Wide Web whereas the .com indicates a commercial Site. Other suffixes include ".gov", a Government Site, ".edu", an Educational Institution's Site, ".org", an organization's Site, etc.

WEB - The Mechanism

Let us suppose you type in an address, say http://www.hotmail.com, in your Browser. Then the following steps are executed before the page actually shows up on your screen:

- By typing the URL in your Browser, you have requested for the Home Page of Hotmail. Note that the default web page (i.e. the on which will be loaded the first) on most Web Servers is index.htm (or index.html) or content.htm (or content.html).

Hence, typing http://www.hotmail.com is equivalent to typing http://www.hotmail.com/index.htm.

- Next, your Browser needs the IP address of Hotmail's Server. The IP Address consists of four numbers (0 - 255) each separated by a (.) (e.g. 202.54.1.180. Every computer on the Internet is assigned a unique IP Address. Now, of course, it is not possible for a human being to remember all these numbers. Hence, a mechanism known as **DNS (Domain Naming System)** is used. Thus Web Sites can be referred to by their **Domain Names** only. This means that at some stage the name has to be translated to the IP address. This is done by **DNS Servers**. Your Browser will contact your nearest DNS Server to get the IP address of Hotmail (say, VSNL).
- If VSNL's server knows the IP address, it will send it to your Browser. Else, it will contact another DNS Server. This continues until the top level DNS Servers are contacted. If the IP is still not found, your Browser returns a "Web Site not found" message.
- If IP is found, your Browser automatically connects to the Hotmail Server and requests the page(s) required.

Nowadays, Browsers have a History (like IE) or maintain a cache (like Netscape) of recently visited Sites. If you visit Hotmail regularly, it will reload quickly and directly from the Cache itself.

If your page is partly downloaded and seems to have frozen, it is probably because a packet has been lost on its way from Hotmail to your Desktop. Try refreshing your Browser.

HTML

HTML - A Markup Language

You are probably under the impression that HTML is a difficult-to-learn language like C/C++. This is not true. HTML stands for HyperText Markup Language. It is associated with the formatting and positioning of text, images, etc. You can compare it with the ".doc" files created by MS Word.

Then why is HTML so popular? The reason is that unlike the various formats which are application and platform dependent, HTML can be universally used. To put it in other words, an HTML file can be viewed on a Windows machine or Linux or even a Unix machine. Since, on the Internet any machine can be used, we need a generalized format for presenting information. Here comes the use of HTML.

What HTML does?

We have seen that HTML is not a programming language. Then why is it used on the Internet? Let us suppose that you want to create a page about yourself to be posted on the Internet. This page will include your name, age, address, hobbies, etc. and even your photograph. You want the page heading to be *"About Me"*, followed by your photograph at the center, etc. HTML allows you to position, align and format Text as well as images. You can even include tables, forms for h4andling user input, etc.

What HTML does not do?

Although HTML is quite capable of presenting information neatly and in an organized manner, it is capable of creating only, what are called, *"Static Pages"*. You are not able to add interactivity to your Site. As a result, your viewers will get bored and will never come back to your Site.

To overcome this drawback, Web Designers today use DHTML, JavaScript, Flash, etc. Thus, a truly professional Web Site should include all the technologies available today to stand out amongst the others.

What do you need to start designing Web Pages?

- **HTML Editor**: To type all the code in. Even though a normal text editor like Notepad will do, most Web Designers prefer using professional packages such as **Frontpage**, **Dreamweaver**, etc. Most of these provide a WYSIWYG (What You See Is What You Get) layout.
- **Web Browser**: To view all that you create.
- **Webspace**: a place to put up your creation and show off to the others.
- Some amount of creativity and patience.

Once you have conceptualised and completed your Site, you need to put it up on the Internet for people to admire. You can either hire space or use the space provided by certain free Web Space Providers. My personal favourite is Tripod (http://www.tripod.com) which provides you upto 15MB space along with freebies like Guestbooks, Counters, etc.)

Your First Page

We'll now create the simple *"Hello World!"* page. Type the following onto your HTML Editor:

```
<html>
<head>
<title>My first Page</title>
</head>
<body>
      <p>Hello World!</p>
</body>
</html>
```

Save the file with a ".htm" (or ".html") extension and view it in your Browser. Congrats! You have created your first HTML page.

Let us analyze the code line by line:
1. Every HTML file should begin with <HTML> and end with the </HTML. Tag.
2. The page is divided into two sections:
   a) **HEAD:** identified by the `<head>…</head>` tags
   b) **BODY:** identified by the `<body>…</body>` tags.
3. Every page can also have a title enclosed in the <title>…</title> tag. The title is displayed in the **Title Bar** of the Browser Window and is used while saving Bookmarks. Only one title can appear in a document, it cannot contain links or highlighting.

Now we shall analyze the HEAD and BODY sections in greater detail and, at the same time, study the various tags available in HTML.

The HEAD of a Page

This section is not displayed directly to the user. It is used to include Page Titles, Keywords, etc. which are used by the Browser for various purposes. The typical head section of a page will look like

```
<head>
<title>My first Page</title>
<meta name="keywords" content="About Me">
<meta name="description" content="My first Web Page created
      in a Text Editor.">
</head>
```

The Meta Tags serve various purposes. One of the most important of these is the indexing of the Site in a **Search Engine**. Most Search Engines divide Web Sites into various categories. They look for specific Meta Tags in the HEAD section and, accordingly, index your Site in the appropriate category. The *"keywords"* and the *"description"* are used for this purpose.

There is yet another application of Meta Tags. This allows you to add some interactivity to your page.

```
<head>
<title>My first Page</title>
<meta http-equiv="refresh" content="5">
</head>
```

This piece of code automatically refreshes your page after 5 seconds. Similarly, you can redirect your user to any other URL by the following code:

```
<head>
<title>My first Page</title>
<meta http-equiv="refresh" content="0;URL=newpage.htm">
</head>
```

This automatically redirects the Browser to *newpage.htm*.

Now that you have come across the Meta Refresh Tag, can you think of any interesting application? Let me give you a clue,

- The first piece of code was used by a certain Site to display Cricket Scores Live during the last World Cup.
- You can create a slideshow using the second example. Think about it.

The `<basefont>` tag sets the default font size, colour and typeface of a particular document. With this, the HEAD section of your Page is completed.

The BODY of a Page

This section contains all the text, images, hyperlinks and other material to be displayed on the page, together with all the HTML formatting codes that control the layout of the page. The <BODY> tag has numerous attributes that control the layout of the page. The are listed below:

Attribute	Description	Usage
`Text`	Decides the default colour of all the text on the page	`Text = "green"`
`Bgcolor`	Defines the default Background Colour of the page	`Bgcolor="#FFFFFF"`
`Background`	Defines the Background Image. This image is tiled throughout the page	`background="image1.gif"`
`Link`	Defines the default colour of all the hyperlinks in the page	`link="#0000FF"`
`Vlink`	Defines the default colour of all the visited hyperlinks in the page	`vlink="#FF00FF"`
`Alink`	Defines the default colour of the active hyperlink in the page	`Alink="#FF0000"`
`Topmargin`	Defines, in pixels, the topmargin space	`topmargin="0"`
`Leftmargin`	Defines, in pixels, the leftmargin space	`Leftmargin="0"`

As you might have already noticed, I have referred to the *text* as *green* whereas the *bgcolor* is *#FFFFFF*. Well, HTML supports two colour schemes. Colours are defined on a **RGB scale**. Here Red has a value #FF0000, Green has that of #00FF00 and Blue has a value of #0000FF. Can you guess the value of White and Black?
To make life easier for the Designer, certain standard colours can be referred to by their names. These include

```
aqua, black, blue, fuchsia, gray, green, lime, maroon,
navy, olive, purple, red, silver, teal, yellow, white
```

and some others. But, not all browsers support this naming scheme. So Beware!

Secondly, Internet widely supports the **GIF** and **JPEG** image formats. The former is preferred for Web Buttons while the latter is used for Photos.

Text on your Page

Okay, now that you have designed the look of your page, how about actually entering some text in it. Text on a Web Page is entered in a paragraph and is enclosed within the `<p>. . . </p>` tags. This tag takes the align (=left, center, right) attribute.

A line break can be inserted with the `<br>` tag.

You can change the font of the text using the `<font>. . .</font>` tags. This tag takes in the following attributes:

```
Color = color; Set the colour of enclosed Text
Face = list; Set the typeface of the enclosed text
Size = value; set the size to an absolute value (1 to 7)
```

Web Fonts:
When your Browser comes across a *font face* tag, it checks to see if the font is loaded on your system. If not, it tries to use the most similar font. Hence, the effect won't be the same as the one the Web Designer wanted. Hence, refrain from using too fancy-looking fonts on your page. The fonts you can safely use are Arial, Courier, Helvetica, Times New Roman and Verdana.
Later I will tell you how to use those special Text Effects on your Web Site without having to worry about the font face.

Lets study some more tags which can appear within the `<p>` tag.

Tag	Description
`<u>. . . </u>`	Underlines the Text within the tag
`<b>. . . </b>`	Makes the text appear Bold
`<i>. . . </i>`	Converts the text to italics

Apart from the paragraph tag, HTML also provides Heading tags. You can use upto 6 headers in a page. These tags are `<h1>. . . </h1>`, `<h2>. . . </h2>` and so on till `<h6>. . . </h6>`.

Now that we have learnt how to present text, we will write some code.

```
<html>
<head>
<title>My first page</title>
</head>

<body bgcolor="#FFFFFF">
<h1 align="center">My first page</h1>
<h2>About Me</h2>
<p>Hi, how are you?<br>
I am fine thank you.</p>
</body>
</html>
```

This page will appear as

My First Page

About Me

Hi, How are you?
I am fine thank you.

Note that HTML tags are defined between '<' and '>' signs. So what do you do when you want to write a sentence like "`x < y`"? For this reason, HTML defines certain character entities for the above two character and also some other characters like:

Symbol	Named Entity	Symbol	Named Entity
[space]		"	&quot;
&	&amp;	<	&lt;
>	&gt;	©	&copy;

Thus, the above statement could be written as "`x < y`"

Some other HTML tags are:

```
<sub>. . . </sub>
<sup>. . . </sup>
<em>. . . </em>
<strong>. . . </strong>
<blockquote>. . . </blockquote>
<pre>. . . </pre>
<address>. . . </address>
```

Please refer the attached list of tags for further details.

Hyperlinks

We discussed about URLs earlier. Every page on the Web must have a unique URL or address. Hence, any Web Page can be linked to any other Web Page or a resource through its URL. The HTML tag used for this is `<a>. . . </a>`, also called the anchor tag. For example, to guide a user from *"My first Page"* to *"My Interests" (say, interests.htm)*, we could have a text on the page which reads *"Click here to know my interests"*. The user, on clicking the link will be directed to interests.htm. The code for this will appear as follows:

```
<a href="interests.htm">Click here to know my interests</a>
```

The text within the tags will appear in the colour defined in the `link` attribute in the `body` tag. You can create a hyperlink to not only another page, but also to a specific portion of your page. This can be done by defining anchors in your page as shown in the example below.

```
<a href="#myaddress>Click here to see my address></a>
                              .
                              .
                              .
<a name="myaddress>Address: Mumbai</a>
```

Try out this tag. The Web works only because of linking of Web Pages together.

When you use frames, you need to specify the `target` attribute of the hyperlink. This attribute will contain the Frame Name where you want the Hyperlink to take effect. The following are various targets supported by the Browsers:

Target	Description
_blank	Loads Document in new Window
_parent	Loads Document in parent of Current Frame
_self	Loads Document in same Frame
_top	Loads document in same Window

Directory Structure on the Internet

Now that you know how to link Web Pages, you can start trying your hand at Designing small Sites. Your site will have, say, 5 pages. You can store your pages in different folders. How do you link to pages in a different sub-directory.

Directory names are separated by a front slash "/". You can thus use relative paths on the Web. To go on level higher, *use* "../"

Tables

So, what do you know now? Well, you know how to display text on a Web Page. You also know how to align your text, change it font face, colour and create hyperlinks. But in the world of Web Design, this will not take you far. Every Web Designer has to master the skills of creating tables consisting of rows and columns. Let us see how this is done?

A table comprises three basic elements

- the table itself,
- the table rows and
- the table columns.

And so, when playing with tables, there are three tags you need to burn into your memory:

The <TABLE></TABLE> tag is the main tag used to create a table. The <TR></TR> tag is used to define a horizontal row. Each row may be further divided into 'cells'. The <TD></TD> tag is used to define cells (or columns) in a row. Pretty simple, right? The <TD> tags always appear within <TR> tags, which in turn are always sandwiched between the <TABLE></TABLE> tags.

Now we'll create a simple table with two rows and three columns:

```
<TABLE BORDER="1">
     <TR>
        <TD>Row 1, Column 1</TD>
        <TD>R1,C2</TD>
        <TD>R1,C3</TD>
     </TR>
     <TR>
        <TD>Row 2,Column 1</TD>
        <TD>R2,C2</TD>
        <TD>R3,C3</TD>
     </TR>
</TABLE>
```

You should see something like this:

Row 1, Column 1	R1, C2	R1, C3
Row 2, Column 1	R2, C2	R3, C3

Note: Please refer to the various attributes of the `<table>` tag provided.

Next, we'll see how to stretch rows and columns. To stretch (i.e. merge) two columns, use the `COLSPAN` attribute within the `<table>` tag. Thus,

```
<TABLE BORDER="1">
     <TR>
        <TD>Row 1, Column 1</TD>
        <TD COLSPAN="2">R1,C2 and C3</TD>
     </TR>
     <TR>
        <TD>Row 2,Column 1</TD>
        <TD>R2,C2</TD>
        <TD>R3,C3</TD>
     </TR>
</TABLE>
```

will result in

Row 1, Column 1	R1, C2 and C3	
Row 2, Column 1	R2, C2	R3, C3

Similarly, Rows can be merged using the `ROWSPAN` attribute.

Images

Statutory Warning: Adding images enhances the look of a page, but they also reduce the speed at which a page downloads on the Net.

The above statement is very true. In fact, the Web Designers worst nightmare is to make his page look attractive without causing his viewers to wait for ages for the page to download.

This, however, does not mean that you should avoid images altogether. Instead, you should design your page optimizing both speed and appearance.

As explained before, the Internet supports GIF, JPEG, ART formats. Of these, GIF is used for buttons and JPEG for photos. The `<img>` tag is used for inserting images in Web Pages.

It has the following attributes.

Attribute	Description
`Src=url`	Source URL of the image to be displayed
`Align`	Align the image to the `top, middle, bottom, left or right` of text in the line.
`Height`	Height of image in pixels
`Width`	Width of image in pixels
`Border`	Thickness of border around the image in pixels
`Alt`	Specifies alternative text to be displayed in place of the image, if browser does not support images or image handling is turned off.

Consider the following example:

```
<img src="photo.jpg" width="300" height="300" alt="My
      Photograph">
```

This will display the image "photo.jpg" on your Browser. How about making an image as a hyperlink. Viewers can then click on an image to navigate your Site. The code will be similar to

```
<a href="pagetogoto.htm"><img src="photo.jpg"></a>
```

Note that if you do not specify the image width and height, the Browser will render the image in its default size.

Image Maps

An image map is a single map that sends you to different places depending on which part of it you click. They come in two flavours,

- **Server-side maps**, which are dependent on and interpreted by the remote server.
- **Client-side maps**, which are defined in regular HTML documents and are processed by the Web Browser.

An image map specification consists of one or more <AREA> elements enclosed within a `<MAP>...</MAP>` sandwich. The <MAP> element takes a single attribute, NAME, used to define a unique name for the image map, like this:

```
<MAP NAME="mapname">
   remainder of code
</MAP>
```

This NAME is used to reference the image map from elsewhere in the document, and is case-sensitive. The <AREA> tag is an 'empty' tag, and is used to define the various clickable regions of the map. It takes a whole bunch of attributes, which we've listed below. The HREF attribute specifies the URL to jump to when that region is clicked. The ALT attribute provides a textual description of the hotspot, useful to non-graphical browsers. The SHAPE attribute defines the shape of the hotspot, and may take any of the following values:

- `SHAPE="DEFAULT"`--the areas on the map not covered by a shape.
- `SHAPE="CIRCLE"`--for a circular region.
- `SHAPE="POLY"`--for an irregular polygonal region.
- `SHAPE="RECT"`--for a rectangular region.

The `COORDS` attribute contains the list of co-ordinates required to define the hotspot area, separated by commas. The number of co-ordinate pairs varies, depending on the SHAPE defined as shown in the box below:

SHAPE	CO-ORDINATES	EXPLANATION
CIRCLE	x,y,r	Define a circle with x,y as the centre and r as the radius.
POLY	x1,y1,x2,y2...,xn,yn	Define a polygon with n points, using an x,y pair for each point of the polygon.
RECT	x1,y1,x2,y2	Define a rectangle with upper-left corner (x1,y1) and lower-right corner (x2,y2).

So, if we know that the co-ordinates for the a particular region of the image, we can turn it into a clickable hotspot with the following code:

```
<MAP NAME="first_map">
  <AREA SHAPE="RECT" HREF="first.htm" COORDS="35,51,63,59"
      ALT="Hotspot1">
</MAP>
```

A hotspot may be any one of three shapes--rectangular, circular or polygonal; and the number of (x,y) pairs to be defined would depend on the shape chosen. A rectangle, for instance, can be completely defined by the co-ordinates for the top-left and bottom-right corners, whereas a circle is defined by its centre and radius. These co-ordinates can be calculated using an image-processing program like Adobe PhotoShop, CorelDraw or other such worthies.

Lists

How many times have you seen documents with points bulleted out neatly and in order? Well, HTML allows you create two types of lists:

- **Unordered Lists:** These are simply bulleted items. The tag is `<UL></UL>`.
- **Ordered Lists:** These lists consists of numbered items. It is defined by the `<OL></OL>` tag. It uses two attributes:
  - `START` attribute specifies the starting value for the first item in the list (the default is 1)
  - `TYPE` attribute controls the numbering style (default value is 1). Other attributes include a, A, i, and I.

Consider the following example:

```
<ul>
      <li>The balloon is red.</li>
      <li>The balloon is blue.</li>
</ul>
<ol>
      <li>The balloon is orange.</li>
      <li>The balloon is green.</li>
</ol>
```

Would appear as

- The balloon is red.
- The balloon is blue.

1. The balloon is orange.
2. The balloon is green.

Forms

Forms are a great way to add interactivity to your pages, and find out more about the people who visit your website. You can use forms to create a digital "guestbook" for your visitors to sign, to conduct a survey or to ask for feedback.

The `<FORM>...</FORM>` tag is used to indicate the beginning and end of a form, and encloses the various form fields. It takes the following attributes:

Attributes	Description
`Action=url`	Specify the URL of the application that will process the form.
`Method=style`	Specify the parameter-passing style, either `get` or `post`.

Forms can have various elements. Each element is identified by a `name`. The various elements include:

Element	Attribute	Description

`Input`	`Type(=text, password, hidden, radio, checkbox)`
	`Name`
	`Value`
	`Size (only for text, password, hidden)`
	`Checked (only for radio and checkbox)`
		Indicates type of Input element
		
		Name of element
		Value of element
		Size of textbox, etc.
		
		Status of radio and checkbox

`<textarea>`
`</teatarea>`
	`Name`
	`Rows`
	`Cols`
	`Wrap (=virtual, physical, off)`
		Virtual => display wrap, but do not transmit to server,
		Physical => display and transmit wrap,
		Off => no wrap

`<select>`
`</select>`
 	
`<option value=>`
		Creates drop-down menus
		
		Used to enter entries in drop-down menu

Form Processing:
Forms are processed using Server Side Scripting Languages like Perl, ASP, etc. Does that mean that you cannot use them without learning these languages? Thankfully, HTML provides a solution. Just write the action tag as `action="mailto:youremailaddress@something.com"` and your form will be mailed there.

Frames

Ever wondered how your Browser can display two or even three pages simultaneously? Well the answer is in **FRAMES**.

When dealing with frames, there are two basic principles you need to keep in mind:

1. every "frame" is actually a normal HTML document. If you wanted to divide your page into two frames, you would need two HTML documents, one for each frame.

2. there is always a master document ("container" or "frameset" document), which contains all information to design frames.

The master document is created using the `<frameset></frameset>` tag. This tag should include the `<frame>` tag for every HTML document. Note that the frameset page should not include any `<body></body>` tags.

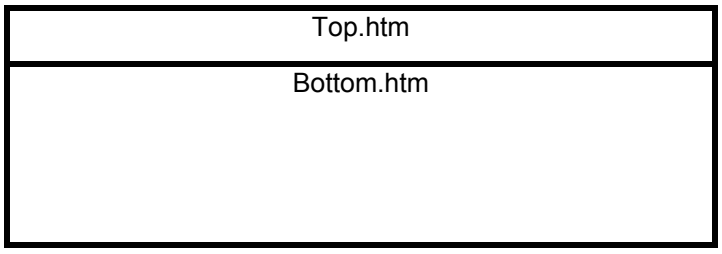
The `<frame>` tag uses the following attributes:

Attribute	Description
`Frameborder=[1/0]`	Enable or Disable a border around the frame.
`Name=string`	Define a name for the frame
`Noresize`	Disable user resizing of the frame
`Scrolling=type`	Always add scrollbars (yes), never add scrollbars(no), or add scrollbars whenever needed (auto).
`Src=url`	Define the URL of the source document for this frame.

Consider the following commonly used examples:

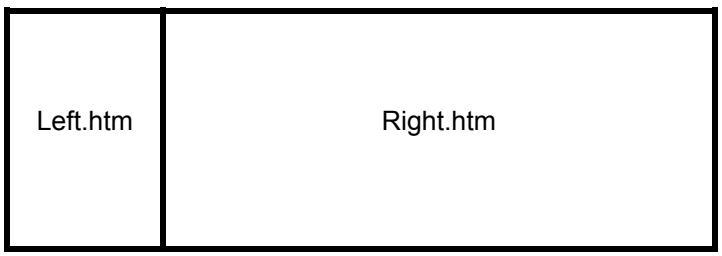
```
<frameset rows="80,*" frameborder="NO" border="1"
      framespacing="0">
  <frame name="topFrame" scrolling="NO" noresize
      src="top.htm" >
  <frame name="mainFrame" src="bottom.htm">
</frameset>
```

which will appear as




```
<frameset cols="80,*" frameborder="NO" border="0"
      framespacing="0">
  <frame name="leftFrame" scrolling="NO" noresize
      src="left.htm">
  <frame name="mainFrame" src="right.htm">
</frameset>
```

which will appear as

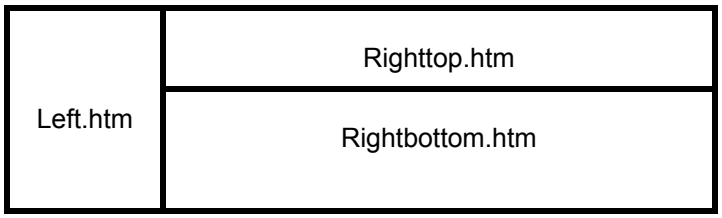


When the size is specified as a *, then it means that the Browser can resize the frame to fit in the screen.

We can even nest frames as shown in the example below.

```
<frameset rows="*" cols="80,*" frameborder="NO" border="0"
      framespacing="0">
  <frame name="leftFrame" scrolling="NO" noresize
      src="left.htm">
  <frameset rows="80,*" frameborder="NO" border="0"
      framespacing="0">
    <frame name="topFrame" noresize scrolling="NO"
      src="righttop.htm">
    <frame name="mainFrame" src="rightbottom.htm">
  </frameset>
</frameset>
```

This will produce:



Layers

Layers make it possible to create and define one or more precisely positioned blocks of HTML in a Web page, and make these blocks move, hide, expand and contract at your whim and fancy. Layers can even be stacked on top of each other, nested within each other, or made to float besides each other. And what we consider to be the coolest thing about this technology--layers can be either transparent or opaque, making it possible to view or hide the content of underlying layers.

Layers were first invented by Netscape, and have been supported since Netscape Navigator version 4.0. Unfortunately, the good folks at Microsoft have yet to include support for this technology in Internet Explorer. Consequently, a Web designer who has a burning desire to use layers on his website needs to create two sets of HTML documents, one set written specifically for Internet Explorer and the other created for Netscape Navigator.

Take a look at this code:

```
<HTML>
<BODY>
<LAYER ID="layer1" LEFT="250" TOP="325" WIDTH="400"
      HEIGHT="200" BGCOLOR="yellow">
      This text is written in a Layer.
</LAYER>
</BODY>
</HTML>
```

The Layer Tag used above has various attributes.

Attribute	Description
ID or NAME	Used to refer to a specific layer
LEFT	used to specify the horizontal coordinates for the upper left corner of the layer
TOP	used to specify the vertical coordinates for the upper left corner of the layer,
WIDTH	controls the width of the layer
HEIGHT	controls the height of the layer
BGCOLOR	Specifies the background colour of layer
BACKGROUND	Adds a background image to a layer
SRC	Specifies a HTML file containing the content for the layer
CLIP	Defines the boundaries of the visible area of the layer. This "clipped" area may be less than the actual width and height of the layer, as specified via the WIDTH and HEIGHT attributes. The value for this attribute consists of a set of four comma-separated numbers, which indicate the left, top, right, and bottom positions. The left and right values are specified as the number of pixels from the left edge of the layer, while the top and bottom values are specified as pixels from the top edge of the layer
VISIBILITY (=HIDE, SHOW, INHERIT)	Specifies whether or not the layer is visible--it takes the values "HIDE", "SHOW", and "INHERIT" (to make the layer inherit the visibility attribute from its parent)
ABOVE and BELOW	used to position a layer above or below another layer.
ZINDEX	Specifies the stacking order of a layer--layers with higher z-index values are placed above those with lower ones

The various layer attributes also have corresponding style sheet definitions. Layers can also be accessed via the document.layers array of the DOM. A layer named "mylayer", for example, could be accessed in the following ways:

```
document.layer1
document.layers[0]
document.layers["layer1"]
```

And here's a quick example that demonstrates just how layers can be made to appear and disappear at will:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- start hiding
function showAndHide()
{   if(document.layers["layer1"].visibility == "show")
    {  document.layers["layer1"].visibility = "hide";
       document.thisForm.clickMe.value = "Show layer";
    }
    else
    {  document.layers["layer1"].visibility = "show";
       document.thisForm.clickMe.value = "Hide layer";
    }
}
// stop hiding -->
```

```
</SCRIPT>
</HEAD>
<BODY>
<LAYER NAME="layer1" Z-INDEX="1" VISIBILITY="SHOW">
         Now you see me...
</LAYER>
<P><BR><BR><BR>
<FORM NAME="thisForm">
    <INPUT NAME="clickMe" TYPE="button" VALUE="Hide layer"
      onClick="showAndHide()">
</FORM>
</BODY>
</HTML>
```

Layers and IE

The LAYER tag is substituted by the DIV and SPAN tag in IE. The <DIV> element is used to define sections of text as 'block-level', while the <SPAN> tag defines them as 'inline'. And once the various sections of your document have been appropriately defined, you can write style sheet rules that affect the layout and presentation of specific sections, leaving other sections untouched.

Let's rewrite the LAYER1 example in IE:

```
<HTML>
<HEAD>
<STYLE type="text/css">
<!--
    .evil {position: absolute;
    top: 300px; left: 100px; color:
    red; background-color: yellow;
    border-width: 10px; border-
    color: blue; border-style:ridge}
 -->
</STYLE>
</HEAD>
<BODY>
<DIV CLASS="evil">
    <P>Lookin' good, feelin' good!
</DIV>
</BODY>
</HTML>
```

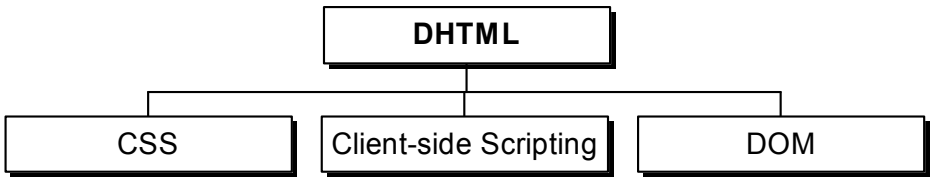
DHTML

What is DHTML?

Dynamic HTML or DHTML as it is more commonly called is the name given to the technique of changing HTML code on the fly. Traditionally HTML is *static* - images once defined cannot be changed and a pre-designed layout cannot be altered. A document coded in HTML is not very interactive - its appearance cannot change in response to user action.

DHTML, on the other hand, makes it possible for your Web page to react and change in response to the user's actions--for example, images animate when the user moves his mouse over them, sections of text change colour in response to a click, and drop-down menus become interactive.

DHTML is not a technology by itself; rather, it is a combination of three different technologies--client-side scripting (JavaScript or VBScript), Cascading Style Sheets (CSS) and the Document Object Model (DOM).



We will study each of these in detail.

Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) are 'rules' or 'styles' for organizing every aspect of an HTML document, including but not limited to, its typefaces, colours, margins, links and other formatting elements. CSS allows you to separate the tags that control the page structure from those that control appearance. Since a single CSS rule can be applied across a number of Web pages, updating the appearance of a series of Web pages is simple.

Consider the following example:

```
<HTML>
<HEAD>
   <TITLE> My first CSS page</TITLE>
   <STYLE TYPE="text/css">
      <!-- -- H1 {color: red} -- -->
   </STYLE>
</HEAD>
<BODY>
   <H1>CSS Rules!!!</H1>
      CSS is so cool we're freezing!
   <H1> Someone get us a blanket!</H1>
</BODY>
</HTML>
```

As you see, all the `H1` tagged text appears in red. Under normal circumstances, we would have applied the `COLOR` attribute of the `<FONT>` tag to the relevant sections of text. Now, you can simply change the colour in the `H1` style rule to green, and watch all text enclosed in the **<H1>** tags turn green.

`H1 {color: red}` is broadly known as a rule, and consists of two components--the 'selector' and the 'declaration'. The selector in this case is the `H1` tag, and the declaration consists of the code enclosed within the curly braces. The declaration can be broken down into a 'property' [color] and a 'value' [red]. Almost every HTML tag can be used as a 'selector'. Over 50 properties are available for use. These are shortly listed below:

Property	Value
`Font`	Any font face
`Size`	Size of font
`Style`	Normal, italic, oblique
`Line-height`	Value in points
`Decoration`	Underline, overline, line-through, blink, none
`Weight`	Normal, bold, etc.
`Variant`	Normal, small-caps
`Case`	Capitalize, lowercase, uppercase, none
`Color`	Colour of HEX
`Background color`	Colour of HEX
`Background image`	Src of image

For a more detailed list, use any good HTML Editor.

There are four different methods of applying style sheet rules to an HTML document:

1. Embed the style rules within the `<HEAD>` of the document itself as shown in the above example. This information is enclosed in comment tags. Hence, older browsers that do not understand style sheet rules simply ignore them.
2. If you'd like to apply a single style or set of styles to multiple HTML documents, you need to first create a separate text file containing the style information, and then either link or import it

into your documents. This text file should only contain the various style rules, and should not include the <STYLE>...</STYLE> tags. Let's assume that we have an external style sheet named 'coolstyle.css', which contains the following rule:

```
H1 {color: blue ; font-family: Courier}
```

Linking your HTML document to this style sheet is as simple as adding the <LINK> tag with its attributes to the <HEAD> of the document. Example:

```
<HEAD>
    <TITLE> My first CSS page</TITLE>
    <LINK REL="style sheet" HREF="coolstyles.css"
      TYPE="text/css">
</HEAD>
```

3. To import a style sheet, you only need to replace the <LINK> tag with the <STYLE>...</STYLE> tags once again, and add a line pointing at the file to be imported. Take a look:

```
<HEAD>
    <TITLE> My first CSS page</TITLE>
    <STYLE TYPE="text/css">
        <!-- -- @import url(coolstyles.css);
        P {color: yellow} -- -->
    </STYLE>
</HEAD>
```

This method allows you to import an external style sheet as well as specify additional style rules for that document within the <STYLE>...</STYLE> tags--something not possible when linking to an external sheet. However, this is currently supported only by IE 4.0.

4. The fourth method of inserting style information is to insert it in the middle of your HTML code, by adding a STYLE attribute to the appropriate tag, and placing the style rules in quotes. Example:

```
<BODY>
    <H1 STYLE="color: blue ; font-family: Courier">
        CSS Rules!!!</H1><P>
        CSS is so cool we're freezing!<P>
    <H1> Someone get us a blanket!</H1>
</BODY>
```

In this case, the rule would only apply to that particular instance of the tag--the line without the H1 tags will appear in the default browser typeface.

Now, we will introduce the 'inheritance' and 'classes' concepts. The following example will clarify the concepts. Suppose if you would like different style rules for the same selector, you can create different 'classes' of a particular selector, each with its own style rules, and specify where each class should be used in the document. These classes can then be invoked by adding the CLASS attribute to the appropriate HTML element.

Consider the following example:

```
<HTML>
<HEAD>
    <STYLE TYPE="text/css">
        <!- - P.top { background-color: black; color: red }
        P.middle { background-color: yellow; color: black }
        P.bottom { background-color: green; color: white }
      - ->
    </STYLE>
</HEAD>
```

```
<BODY>
    <P CLASS="top">I'm a King</P>
    <P CLASS="middle">I'm a rock star</P>
    <P CLASS="bottom">I'm myself</P>
</BODY>
</HTML>
```

The browser will display each line of text in a different background and foreground colour, depending on the style rule defined for that particular class. Note that the period precedes each class definition.

You can even define standalone classes and use them with any tag in the document. Example:

```
<HEAD>
    <STYLE TYPE="text/css">
        <!- - .beast { background-color: black; color: red
      } - ->
    </STYLE>
</HEAD>
<BODY>
    <H1 CLASS="beast">maneater</H1><BR>
    Domesticated canine
    <P CLASS="beast">Piranha
</BODY>
```

Now, let's move on to an interesting species known as pseudo-classes--essentially, pre-defined classes for various selectors.
For example, the anchor element <A> has the following three pre-defined classes:

- A:link--for all hyperlinks;
- A:visited--for visited hyperlinks;
- A:active--for the last hyperlink clicked.
- A:hover -- when you position your mouse over the hyperlink

This covers the basics of CSS. The best way to use CSS is through a good HTML Editor like Macromedia Dreamweaver.

Now we'll see how to use CSS to position elements on the Web Page. For this, we'll study two tags, `<DIV>` and `<SPAN>`. These tags are commonly used to define and segregate content without altering it or imposing any formatting rules on it--in other words, permitting you to divide your document into sections that can be worked upon independently, without affecting the layout of the content. The `<DIV>` element is used to define sections of text as block-level, while the `<SPAN>` tag defines them as inline. Consider the `<I>, <b> and <u>` tags as inline tags while, the `<p>, <ul>,` etc. apply to 'blocks' of text like paragraphs or lists.

Once the various sections of your document have been appropriately defined, you can write style sheet rules that affect the layout and presentation of specific sections, leaving other sections untouched. For example:

```
<HTML>
<HEAD>
    <STYLE TYPE="text/css">
        SPAN {color: red}
        DIV {color: blue}
    </STYLE>
</HEAD>
```

```
<BODY>
    <SPAN>
        Spanning the Red Sea!
    </SPAN>
    <DIV>
        Turning blue!
    </DIV>
</BODY>
</HTML>
```

The position property accepts the values absolute, relative and static. With absolute positioning, you have the power to locate elements in specific positions on a page, regardless of the position of other elements on that same page, while relative positioning allows elements to be placed relative to the positions of other elements on the same page. In both cases, the left and top properties are used to specify coordinates for the upper left corner of the 'box' that contains it.

```
<HEAD>
    <STYLE TYPE="text/css">
        SPAN.walker {color: red;
            position: absolute;
            left: 50; top: 100;
            font-family: "Comic Sans MS"}
        SPAN.cliff {color: blue;
            position: absolute; left: 125; top: 10;
            font-family: "Comic Sans MS";
            font-size: xx-large}
    </STYLE>
</HEAD>
<BODY>
    <SPAN CLASS="walker">
        Hanging off a cliff,
        walking on thin air!
    </SPAN>
    <SPAN CLASS="cliff">
        The cliff, as seen from a
        browser
    </SPAN>
</BODY>
</HTML>
```

Next, we have relative positioning, which does almost the same thing with one minor difference--it allows you to specify the location of an element relative to where it would normally appear. The next example should make this clear:

```
<HEAD>
    <STYLE TYPE="text/css">
        H2 {position: relative;
        left: 55 px; top: 25 px;
        color: blue}
    </STYLE>
</HEAD>
<BODY>
    <H2>Relativity theory!</H2>
</BODY>
```

And finally, we come to position: static which simply specifies that the element should be placed where it would normally appear; the left and top properties do not apply in this case.
Now for something that sounds like it was stolen from a bad sci-fi novel--the z-index. Take a look at the two examples below, and things will become clearer:

```
<BODY>
    <H3 style="position: relative;
        top: 15; z-index: 100;
        left:25; color: black">
        The man</H3>
    <H2 style="position: relative;
        top: -25; z-index: 200;
        left: 105; color: red">
        The woman</H2>
    <H1 style="position: relative;
        top: -75; z-index: 300;
        left: 85; color: blue">
        The fish</H1>
</BODY>
```

Paste the above code in a text editor and view the resulting .html file in your browser. You will see that the z-index allows you to specify which element appears on top, and the order in which all elements below it are arranged. A higher z-index value pushes the element to the top, and a lower value pulls it down. This technique is very useful, particularly when overlapping text. The most common example of this kind of thing is the drop shadow. We've demonstrated this in the example below:

```
<BODY>
    <H1 style="position: relative;
        top: 15; left: 25; z-index: 10;
        color: #00000">
        Make my day</H1>
    <H1 style="position: relative;
        top: -40; left: 27; z-index: 8;
        color: #8C8C8C">
        Make my day</H1>
    <H1 style="position: relative;
        top: -95; left: 29; z-index: 6;
        color: #ADADAD">
        Make my day</H1>
    <H1 style="position: relative;
        top: -150; left: 31; z-index: 4;
        color: #BDBDBD">
        Make my day</H1>
</BODY>
```

Now, here's a cool one--the incredible visibility property, which allows you to make sections of your page appear and disappear at will. Take this code:

```
<H3 STYLE="visibility: visible">
    Seen nothin' yet</H3>
```

Now replace it with this code:

```
<H3 STYLE="visibility: hidden">
    Seen nothin' yet</H3>
```

As you can see, the visibility property allows you to make text or images visible or invisible at will. It accepts the self-explanatory values visible, hidden and inherit (this is the initial value, and indicates that the element should take on the visibility characteristics of its parent).

A few things to keep in mind when playing with style sheets:

- Test, test and test again!
- Since the two major browsers don't see eye to eye on many of the important style sheet properties, what looks good in one may look terrible in the other!
- Always use style sheets in combination with good ol' HTML. Style sheets have only come into their own with the 4.x versions of Navigator and Explorer. Consequently, older browsers will not be able to recognize the style rules you've worked on all night--for these poor creatures, we suggest you back up your style rules with regular HTML tags. For example, use the font-size property, but also use the <FONT SIZE=whatever> tag, so that users with older browsers also get a chance to enjoy your page.
- Exercise restraint.

JavaScript

What is Javascript?

Cascading Style Sheets (CSS) do some pretty wonderful things to your Web page. However, they cannot add interactivity--such as images changing as you move your mouse; sections of a page appearing and disappearing; content changing dynamically and complex calculations being performed automatically.

Enter JavaScript--a scripting language that lets you do the near impossible, while simultaneously being quick and easy to learn. JavaScript is not Java. The latter is a language designed to run on any platform, anywhere. JavaScript, on the other hand, only works in a Web browser.

JavaScript is embedded within your HTML page, and is interpreted by the browser as the page loads. This reduces the load on the Web server, as opposed to CGI scripts.

What are the alternatives?

There are numerous alternatives to Javascript. These include:

- **Jscript:** According to Microsoft, JScript is an "object-oriented and interpreted scripting language that operates within the purview of the browser". One of JScript's biggest advantages is that it builds on and enhances many of Internet Explorer's existing capabilities. But this power comes at a price--JScript will only work in Internet Explorer, and is not supported by other browsers. One of JScript's biggest advantages is that it builds on and enhances many of Internet Explorer's existing capabilities. But this power comes at a price--JScript will only work in Internet Explorer, and is not supported by other browsers.
- **VBScript: I**t's Microsoft's attempt to help Visual Basic programmers make the transition to Web programming. If you're familiar with Microsoft's Visual Basic family of programming languages, you'll quickly recognize the syntax and structure of VBScript. However, it is supported only by Internet Explorer.
- **ActiveX:** Developed by Microsoft, ActiveX refers to a "set of strategic object-oriented program technologies and tools." The concept here is pretty simple--an ActiveX component is a self-sufficient program that can be run anywhere on a Windows or Macintosh system. This component is also known as an ActiveX "control". ActiveX controls can be downloaded as small programs or animations for Web pages, or can be created using programming tools like C, Visual Basic or PowerBuilder. One of the technology's main advantages is that it can be reused by different applications--so an ActiveX control created for use in a Web page can also be used in your Visual Basic application with no modification required.
- **Java:** Java can be used to write both applications and "applets". Java applets can perform calculations, interactive animations, file input/output and other simple tasks without having to send a user request back to the server. Applets are incorporated into Web pages via the `<APPLET>` tag--here's an example:

```
<APPLET CODE="godfather"
      WIDTH=100 HEIGHT=100>
</APPLET>
```

  This will run the Java applet named **godfather** in a window with a 100 by 100 pixel dimension.
- **Flash:** Flash allows Web designers to create resizable and extremely compact user interfaces, illustrations, animations and other dazzling effects for their site. If you're looking for interactivity, this is one technology that is well worth investing in--it lets you produce Web

sites with vector and bitmap graphics, full-motion video, animation and MP3 audio. Since Flash uses vector graphics, file size is extremely small, and it's also possible to "stream" Flash files over slow modem connections. Another advantage of vector graphics is that they are resolution-independent--consequently, a Flash animation looks the same regardless of monitor size or screen resolution.

What is the difference between Java and Java Applets?

Java (or Java Applications) are stand-alone programs run by the operating applications. They are compiled. Applets, on the other hand, are designed to be embedded in a HTML file and executed by the Browser.

An applet is in the form of a .class file which is often referred to as the *"byte code"*. It is loaded by the Browser along with the HTML file and then executed. Unlike the application, the applet has certain security concerns. For example, it cannot access files, etc.

How to write JavaScript?

According to me, the best way to write Javascripts is to start coding. However, you will need to know some basics first, which shall be explained now.

Firstly, a Javascript is included in the HTML file using the `<SCRIPT></SCRIPT>` tag like this,

```
<HTML>
<HEAD>
<TITLE>JavaScript Page</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    <!-- hide from older browsers

    // stop hiding -->
</SCRIPT>
</HEAD>
<BODY>
    My first bit of JavaScript code.
</BODY>
</HTML>
```

The `language` tag is used to specify which scripting language is being used. The script may be placed either in the `HEAD` or the `BODY` section.Comments are used to hide the code from older browser's which may not support JavaScript. We can also use the `<NOSCRIPT></NOSCRIPT>` tag as follows:

```
<SCRIPT LANGUAGE="JavaScript">
    <!-- hide from older browsers

    // stop hiding -->
</SCRIPT>
<NOSCRIPT>Your Browser does not support
      JavaScript</NOSCRIPT>
```

Comments

Like C, Javascript supports single line or multi-line comments:

```
// This is a single line comment
/* This is a multi-
      line comment. */
```

Variables

Variables in Javascript are used to store values. These may be numbers or strings.

Rules for valid variable names:

- The first time you use a variable, you need to declare or define it using the special JavaScript keyword var. This keyword is followed by the variable name, an assignment operator and the value to be assigned to the variable.
- Variable names are case-sensitive
- Variable names must begin with either a letter or an underscore--they cannot begin with a number. However, they may contain numbers, but not symbols such as #, @, ! or &. So while number_one and _god are perfectly valid variable names, 3some is not!
- Reserved JavaScript keywords such as var, goto, case and others cannot be used as variable names.
- Variable names should be descriptive. This is not a JavaScript rule--it's common sense!

```
Var x = 10;
Var y = "hello!";
Var name = "";
```

Operators

JavaScript supports the following different category of operators:

- **Arithmetic:** `+ - * / % ( ) ++ --`
- **Comparision:** `== != < >  <= >=`
- **Logical:** `&& ||`

Input and Output

By now you must be craving to try out some real scripting. But before that, you need to study the following two statements.

- `Alert`: This function allows you to display messages, variables, etc.
- `Prompt`: This displays a dialog box for accepting input from the user.
- `Confirm`: This displays a dialog box providing the user with two option "OK" and "Cancel".

Consider the following examples;

```
<script language="Javascript"
      alert("Hello!");
      var name;
      name = prompt("Enter your name", "your name here");
      alert("Your name is " + name);
      var reply = confirm("Are you Sure?");
</script>
```

It produces the following outputs.

1. It greets you with the messahe "Hello!"

2. Then it prompts you to enter your name.

3. Then, it displays what you had enteres. Note the use of the variable *name* here

4. It then confirms whether your name is correct. If you click on OK, it returns `true`, else it returns `false`.

Conditional Statement

Like any other language, Javascript supports the if..else statement.

```
If (condition)
      {     statements;
      }else
      {     statements;
      }
```

Loops

Javascript supports the following Loop Structures:

- **For loop**:

```
for (  initialize counter ;
        conditional expression ;
        update counter)
    {
        do this
    }
```

For example,

```
<SCRIPT LANGUAGE="JavaScript">
    <!- - Start hiding script
    var count ;
    // Loop
    for ( count = 0 ; count <= 10 ; count++)
    {alert ( "You have clicked this "  count  " times!");
    }
     alert ("Enough!");
    // stop hiding - ->
</SCRIPT>
```

- **While loop:**

```
While(condition)
{     statements;
}
```

For example,

```
<SCRIPT LANGUAGE="JavaScript">
    <!- - Start hiding script
    var count = 0;
    // Loop
    while(count <= 10)
    {alert ( "You have clicked this "  count  " times!");
      count++;
    }
     alert ("Enough!");
    // stop hiding - ->
</SCRIPT>
```

As seen, the working of these loops is similar to those in C language.

Arrays

What do you do when you want to store a number of related items? Simple, you use Arrays. JavaScript allows you to declare arrays using the `new` statement.

```
// Define the array
Var colours = new array(3);

//Assign values to the array.
 colours[0] = "red";
 colours[1] = "blue";
 colours[2] = "green";
 colours[3] = "yellow";
```

```
```

Another method for defining as well as assigning values to an array is:

```
Var colours = new array("red", "blue", "green", "yellow");
```

Functions

A function that performs a single task may be used at different points within the same script without the need to rewrite the code again.

So here's what the average function is like,

```
function function_name
    (parameters)
{
    statements ;
    return [variable] ;
}
```

Rules for function names:

- The function name must abide by the same rules as a variable name--the first character must be a letter or an underscore, and should not contain characters like `! @ #`.
- The function name is followed by a list of parameters, separated by commas, in parentheses. A parameter is nothing more frightening than a variable, which is defined when the function is called.

The body (enclosed within curly brackets) consists of statements to be executed, and usually also includes the return command, used to return the results of calculations to the main program.

JavaScript and Mathematics

JavaScript supports a Math Object which provides various Mathematical, Statistical and Trigonometric functions. The various functions included are:

`Math.sin(n)` - returns the sine of the angle (in radians) passed
`Math.cos(n)` - returns the cosine of the angle (in radians) passed
`Math.tan(n)` - returns the tangent of the angle (in radians) passed
`Math.asin(n)` - returns the arc sine of the angle (in radians) passed
`Math.acos(n)` - returns the arc cosine of the angle (in radians) passed
`Math.atan(n)` - returns the arc tangent of the angle (in radians) passed
`Math.log(n)` - returns the natural logarithm of the number passed
`Math.exp(n)` - returns a value equal to e^n, where "e" is an animal named the Euler constant
`Math.random()` - returns a random number between 0 and 1
`Math.round(n)` - rounds the number passed
`Math.abs(n)` - returns the absolute value of the number passed
`Math.sqrt(n)` - returns the square root of the number passed
`Math.pow(a, b)` - returns the value of a^b
`Math.floor(n)` - returns the highest integer greater than or equal to the passed number
`Math.ceil(n)` - returns the lowest integer greater than or equal to the passed number.

Cookies

Ever wondered how sites like Yahoo! and Microsoft know to call you by name each time you visit their Web site? Or how Amazon.com prints your e-mail address on the front page each time you drop by their online bookstore? How do they do this? Well, sites such as the ones listed above [and a billion more besides] save pieces of personal information, such as your name, email address and taste in the opposite sex, in files on your hard drive, and read these files every time you re-visit the site. These files are known as 'cookies'

Cookies are used to record information about your activities on a particular site, and can only be read by the site that created them. The Excite portal, for example, allows you to customise the kind of content displayed on the site, and uses a cookie to store this data. The next time you visit the Excite portal, the contents of the cookie are read, and the appropriate information displayed. It's kind of like going to a chic restaurant, and having the maitre'd call you by name--something which hasn't happened to us of late.

Before you start using cookies, there are a few things you should know:

- Cookie technology has been supported correctly in Netscape Navigator since version 2.0. and in Internet Explorer 4.0 and above.
- Every browser allows the user to turn off the 'cookie' feature. If a user decides to turn off this feature, your cookies will not be saved, and any attempt to access them will fail.
- A single domain cannot set more than twenty cookies. A single cookie cannot exceed 4 KB in size. The maximum number of cookies that may be set is 300.


In case you are the kind of guy who likes to tweak around, note that cookies are stored in the following locations:

Netscape Communicator: C:\Program Files\Netscape\Users\<username>\cookies.txt

Internet Explorer: C:\Windows\Cookies\

What is a cookie?

A cookie usually possesses five types of attributes;

- The first of these is a NAME=VALUE pair, used to store information such as a username, e-mail address, credit-card number, etc. The NAME is a string used to identify the cookie, while the VALUE is the data to be stored in the cookie.
  Example: `username=god`
- The second is the EXPIRES attribute, which defines the date on which the cookie expires. The date must be in the format "weekday, dd-mon-yy hh:mm:ss GMT". For example:
  `expires="Fri, 31-Dec-1999 02:15:03 GMT"`
- The third is the PATH attribute, used to set the top-level directory on the Web server from which your cookies can be accessed. In most cases, this is set to `path=/` to ensure that the cookie can be accessed by each and every HTML document on the server.
- The fourth is the DOMAIN attribute, used when the server is setting the cookie [something we'll cover much, much later on], and the fifth is the SECURE attribute, which indicates that a cookie should only be set if there exists a secure protocol between the browser and the server.

Of all the five attributes, the first is the only one that is not optional. If you plan to use more than one, you should separate them with semi-colons.

How to write a Cookie?

The following code sets a cookie:

```
<HTML>
<HEAD>
<TITLE>My First Cookie</TITLE>
<SCRIPT
    LANGUAGE = "JavaScript">
    <!-- start hiding
    function setcookie()
    {
        // Ask for input
        var your_name = prompt("What's your name?", "");
        // Set expiry date
        var expiry_date = new Date("December 31, 1999");
        var the_cookie_date = expiry_date.toGMTString();
        // Set cookie parameters
        var the_whoami_cookie =
            "username="  escape(your_name);
        the_whoami_cookie =
            the_whoami_cookie  ";expires="
            the_cookie_date;
        // Write cookie
        document.cookie = the_whoami_cookie;
    }
    // stop hiding -->
</SCRIPT>
</HEAD>
<BODY
    onLoad="setcookie()">
</BODY>
</HTML>
```

The onLoad event calls the function setcookie() when the page loads. The function accepts the user name through a prompt call. Next, we define a new Date object called expiry_date, and initialise it to December 31, 1999. This date is then converted to Greenwich Mean Time via the toGMTString method, and stored in yet another variable called the_cookie_date. Finally, it's time to actually write the cookie -- we do this by creating a variable called the_whoami_cookie, and writing the NAME [username=your_name] and EXPIRES [expires=the_cookie_date] parameters to the cookie file, separated by semi-colons. The document.cookie property is used to actually set the cookie. An important point to note here is that a cookie cannot contain white space -- hence our use of the inbuilt JavaScript escape() method, which replaces non-alphanumeric characters by their HTML equivalents. White spaces, for example, are converted to %20". And of course, you can reverse this conversion with the intelligently-named unescape() method. The document.cookie property returns a list of all the cookies available to the current HTML document.

How to read a Cookie?

The readCookie() function in the following example reads the cookie just set above.

```
<HTML>
<HEAD>
<TITLE>Reading a cookie</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    function readCookie(name)
    {   var omega = name  "=";
        var alen = omega.length;
        var clen = document.cookie.length;
        var i = 0;
        while (i < clen)
        {   var j = i + alen;
            if (document.cookie.substring(i,j) == omega)
            {   return getCookieValue(j);
            }
            i = document.cookie.indexOf( " ", i)  1;
            if (i == 0) break;
        }
        return null;
    }

    function getCookieValue(offset)
    {   var endstr = document.cookie.indexOf(";", offset);
        if (endstr == -1)
        {
            endstr = document.cookie.length;
        }
        return unescape(document.cookie.substring(offset,
      endstr));
    }

    function writePage()
    {       document.write(readCookie("username"));
    }
</SCRIPT>
</HEAD>
<BODY onLoad="writePage()">
</BODY>
</HTML>
```

These functions make use of the following variables:

- `omega` = used to store the name of the cookie.
- `alen` = length of the string defined in "omega".
- `clen` = total length of the cookie
- `offset` = the position of the first character in the cookie data string.
- `endstr` = the position of the last character in the cookie data string.

We"ll begin with the readCookie() function, which accepts the NAME of a cookie as a parameter. This cookie must have [obviously!] been set previously. This NAME is then placed in the omega variable with an equality operator, and a while loop is used in combination with the document.substring method to search the document.cookie property until it finds the NAME. The position of this value in the string is then passed to the second function, getCookieValue(), in the form of the variable j. If it is unable to find the NAME, it exits the loop without doing anything.

The getCookieValue() function receives the position of the string [j], and uses the indexOf method to locate the first semicolon (which, as you remember, is the delimiter used in a cookie) that

appears after this position. This semicolon marks the end of the cookie VALUE, and its position is assigned to the variable endstr. If a semicolon is not found, the indexOf method returns a value of -1, and the end of the requested cookie is assumed to be equal to the length of document.cookie.

Now that the start and end positions of the cookie value are known, the unescape() function is used to convert the extracted string back into readable ASCII, which is passed to the writePage() function to be displayed on the page.

How to delete a cookie?

Setting the EXPIRES attribute to a date in the past will cause the browser to delete the cookie. However, it's been found that many browsers refuse to delete the cookie unless the expiry date is specifically set to April 13, 1970 00:00:00 GMT

The following Code Illustrates this,

```
function killCookie(name)
    {
        var cookie_value = readCookie(name);
        document.cookie = name  '='  cookie_value
            '; expires=Fri, 13-Apr-1970 00:00:00 GMT';
    }
```

Examples of JavaScript

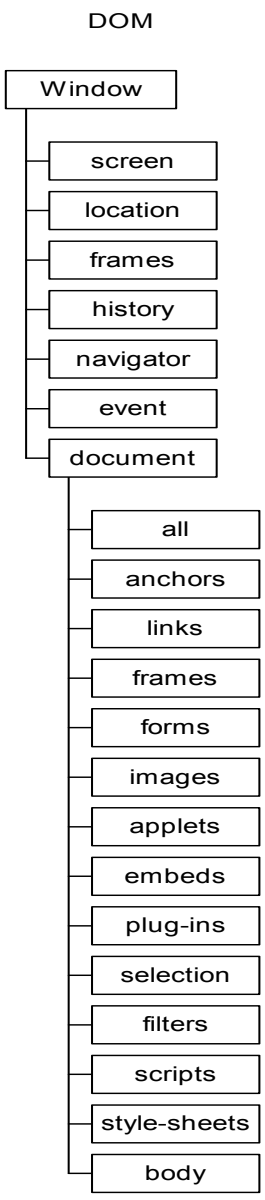
Now we'll put everything together in something useful. We'll create a **Calculator** which will accept as input the two operands and the operation (`+ - * / %`). This is one of the favourite examples amongst learners.

```
<html>
<head>
<title>A JavaScript Calculator</title>
<script LANGUAGE="JavaScript">

// Input
var a, b, result, operation;
a = prompt("Enter the first operand", 0);
b = prompt("Enter the second operand", 0);
operation = prompt("Enter the operation desired +, -, *, /,
      %", "+");

if(operation == "+")
{     result = a + b;   // Addition
}
if(operation == "-")
{     result = a - b;   // Subtraction
}
if(operation == "*")
{     result = a * b;   // Multiplication
}
if(operation == "/")
{     result = a / b;   // Division
}
if(operation == "%")
```

```
{      result = a % b;    // Modulus
}

// Display result
alert("The result is: " + result);
</script>
</head>
<body>
</body>
</html>
```

Document Object Model (DOM)

What is the DOM?



The Document Object Model is the essence of what makes DHTML so dynamic--it is a method of referencing and accessing every element on your Web page, or in your browser window, via a hierarchical tree structure.

This includes elements like the current URL, the browser type, the text on the page and the data entered by the user into a form. And the DOM also makes it possible to modify every element on your page via a scripting language such as JavaScript, or even create new elements as and when you desire.

The DOM uses objects, which in turn may spawn child objects. Every object has certain properties or characteristics, such as style, colour, and position. Objects are referenced either via the tree-like hierarchy, or by explicitly naming them with the ID or NAME attribute, and then calling them by their given ID or NAME.

The DOM hierarchy looks something like the tree to the left.

Of all the objects, the most frequently used object is the document object which is used to reference elements of the HTML document currently loaded into the browser window. This includes the title, background and link colours, as well as all the child objects like images, anchors and forms.

The BODY object

Now that you have learnt CSS and JavaScript, you must be set to start Web Designing in a big way. You have learnt input and output in JavaScript through `prompt` and `alert` function respectively. You also know how to define your own styles and thereby redefine the functionality of the HTML tags themselves. You have the freedom to position elements anywhere on the page independent of the screen size, resolution, Browser, etc.

However, you still lack something. That something is in manipulating existing HTML elements in a page after the page has been loaded. For this, you have to combine the powers of JavaScript with DOM.

Name your Objects

What are the basic elements you would most likely want to play with? Guess? Well images, hyperlinks and, of course, form elements. Note that these elements are a part of the BODY object in the DOM. Hence, you have to first name every element as follows:

```
<BODY>
    <IMG SRC="image1.gif" NAME="first_image" ><BR>
    <IMG SRC="image2.gif" NAME="second_image"><BR>
    <A HREF="first.htm" NAME="first_anchor">
         This is an anchor</A>
</BODY>
```

Here there are three objects:

1. `first_image`: an image with src = "image1.gif".
2. `second_image`: an image with src = "image2.gif".
3. `first_anchor`: an anchor pointing to "first.htm".

How to access your Objects

Now that you have defined three objects, how do you refer to them? We have two methods;

1. Refer to all the images as `document.images[number]`. Thus, `first_image` can be referred to as `document.images[0]`, and so on.
   The anchor can be accessed with `document.anchors[0]`.
2. Refer to objects by their names.
   Thus, `document.first_image` would refer to the first image.

Note that, the first method, while somewhat complicated when dealing with long documents, works in both the major browsers, while the second method is more Internet Explorer-specific.

Modify Objects

Every object has properties, which may be modified by a scripting language. These properties usually correspond closely to the attributes taken by the relevant HTML tag. So, if you wanted to access the SRC attribute of the first image in our example, you could use `document.images[0].src` or `first_image.src`.

Or we could access the HREF attribute of the solitary link via either `document.anchors[0].href` or `first_anchor.href`.

Some functions in the document object

The `document.write()` and `document.writeln()` functions help you write HTML content onto a page.

```
document.write("<p>This text is being generated through
      JavaScript.</p>");
```

Similarly the `document.close()` closes the current Document.

Events and Event Handlers

If you are a hard-core programmer in a language like VB or VC++, you are probably searching for Command Button Click or Mouse Move events. Well, HTML provides you these events as well. All these actions, or 'events', are handled by special JavaScript triggers called 'event handlers'. Here's a list of common event handlers, and their functions:

- `onAbort`--invoked when the user aborts the loading of an image by clicking the Stop button
- `onClick`--invoked when the user clicks the specified object
- `onFocus`--invoked when the target object receives focus
- `onBlur`--invoked when the target object loses focus
- `onMouseOver`--invoked when the user passes the mouse pointer over the target object
- `onMouseOut`--invoked when the mouse pointer leaves the target object
- `onSubmit`--invoked when the user clicks the Submit button in a form
- `onReset`--invoked when the user clicks the Reset button in a form
- `onLoad`--invoked when the target image or document is loaded
- `onUnload`--invoked when the target image or document is unloaded

Consider the following example,

```
<HTML>
<HEAD>
<TITLE>Event handlers</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- start hiding
// Write function to pop up alert box
function click_form_button()
{   alert ("You clicked me!!!");
}
// stop hiding -->
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="smoking_gun">
    <INPUT TYPE="Submit" NAME="submit_button"
      VALUE="Click me!" onClick="click_form_button()">
</FORM>
</BODY>
</HTML>
```

As you can see, we've first written a JavaScript function called click_form_button, which displays an alert box. This function has then been linked to the onClick event handler in our HTML page, such that the function is invoked when the browser receives a click on the target object (the 'Submit' button in the form).
Similarly, the following example changes the text on the button.

```
// Write function to change text on button
function click_form_button()
{ document.forms[0].submit_button.value="You clicked
      me!!!";
}
```

In this case, we're referencing the value attribute of the submit_button object, which is, in turn, a child of the first form in the document, and then changing the text on it when the button receives a click--a combination of JavaScript and the DOM. The result is an HTML document that changes in response to user activity--exactly what Dynamic HTML is all about!

Consider the next example. This changes the image when the user moves the moves over it. This is called the **Mouse Over Effect** by Designers.

```
<HTML>
<HEAD>
<TITLE>Mouse Over Effect</TITLE>
<SCRIPT LANGUAGE="JavaScript">
// Change image when the mouse
    // is moved over the image.
    function imageSwap()
    {
        document.images[0].src="image2.gif" ;
    }
    // Revert to the old image when the
    // mouse leaves the image area.
    function imageRevert()
    {
        document.images[0].src="image1.gif" ;
    }
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
    <A HREF="#" onMouseOver="imageSwap()"
      onMouseOut="imageRevert()"><IMG SRC="image1.gif"
            BORDER="0"></A>
</CENTER>
</BODY>
</HTML>
```

We've used the `onMouseOver` and `onMouseOut` event handlers to accomplish the image swap. We've written two functions, imageSwap() and imageRevert(), used to change the image back and forth respectively. The first function is invoked by the onMouseOver handler, and replaces the current image with image2.gif. The second function is invoked by the onMouseOut event handler when the mouse pointer leaves the image area, and replaces image2.gif with the original image1.gif.

The Window Object

The window object has its own properties and functions. One of these is the `window.open()` which accepts the following parameters.

```
window.open ("URL-to-open", "window-name",
                    "list-of-appendages-to-be-displayed")
```

The first parameter is the URL to be opened, The third includes the following;

- Width--controls the width of the new window
- Height--controls the height of the new window
- Location--displays the address or location box
- Toolbar--displays the toolbar
- Status--displays the status bar
- Menubar--displays the menu bar
- Scrollbars--displays the scrollbars
- Directories--displays the directories or links bar
- Resizable--is the window resizable?

These components can be turned on or off by equating them with either a 1 [on] or 0 [off].


Use the following code to open a window in the fullscreen mode. The window will be opened sans the menubar and even the title bar! It will occupy every inch of space on your desktop.

Other functions / properties include:

- `Window.close()` : Closes the window.
- `Window.status`: Changes text on the Status Bar of Browser.
- `Widow.defaultstatus`: Defines the Default Status of the Widow.

The following example changes the status bar text when the mouse is moved over a hyperlink.

```
<html>
<head>
<title>Status Bar Effects</title>
</head>
<body>
<p><a href="#" onmouseover="window.status='Click Now.';
      return true;" Onmouseout="window.status=''; return
      true;">Click Here</a></p>
</body>
</html>
```

The above script uses what are called inline functions, i.e. functions written within HTML tags itself.

Form Processing through JavaScript

```
<HTML>
<HEAD>
<TITLE>Javascript form</TITLE>
</HEAD>
<BODY>
    <FORM NAME="good_dog">
        Your dog's name?<BR>
        <INPUT TYPE="Text" NAME="name">
    </FORM>
</BODY>
```

```
</HTML>
```

The forms object lies under the document object in the DOM hierarchy, and may be accessed in either of the following ways:
`document.forms[0]` -- by its index number in the forms array
`document.good_dog` -- by its NAME

Just as all the forms in a document constitute a forms array, all the elements of a particular form make up a so-called elements array. This makes it possible to reference the various form elements via the DOM; for example, the text box in the example above could be referenced in any of the following three ways:

```
document.forms[0].elements[0]
document.forms[0].name
document.good_dog.name
```

All form elements possess certain properties, which may be accessed or modified by JavaScript's. For example, text fields have the value property, which represents the text entered into the box, while check boxes and radio buttons possess the checked property, which indicates their current status, whether off or on. Take a look at the example below, which demonstrates how to access text box properties:

```
function displayName()
    {
        alert(document.good_dog.name.value);
```

There are four JavaScript event handlers that come in handy when dealing with text fields:

- `onFocus` --invoked when the user clicks, or 'focuses', on the text field.
- `onBlur` --invoked when the user clicks, or 'focuses', outside the text field.
- `onChange` --invoked when the user changes the contents of the text field.
- `onSelect` --invoked when the user selects the contents of the text field.

Similar properties exist for the other Form Elements.


One of the most important use of JavaScript in Form Processing is in Validation of Forms. We know that Forms are processed through Server side scripting. However, JavaScript can be used to validate the form and submit it only if data is correct.

Form Validation

```
<HTML>
<HEAD>
<TITLE>Javascript form</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function validate()
{     if (document.frmMain.name.value == "")
      {     alert("Please enter your name");
      } else
      {     frmMain.submit();
      }
}
</SCRIPT>
```

```
</HEAD>
<BODY>
    <FORM NAME="frmMain">
        Enter your Name: <BR>
        <INPUT TYPE="Text" NAME="name">
        <INPUT TYPE="Submit" VALUE="Submit"
      onclick="javascript:validate()">
    </FORM>
</BODY>
</HTML>
```

The above code will not Submit the Form until the user enters something into the Text Field.

JavaScript and Frames

One of the objects in the DOM hierarchy is the frames object, usually found nestling comfortably under the document object. This frames object makes it possible to access each and every frame currently loaded into the browser window.

A browser window that contains frames is called the 'parent' of the frames contained within it, while the frames within it are called 'children'. These so-called 'child' frames may be accessed either by their NAME, or via the 'frames' array.

In order to explain just how frames are accessed in the DOM, let's assume that we have a Web page vertically divided into two frames, sensibly named 'left' and 'right'. In the DOM, the left frame can be accessed either as:

`top.left` - via its NAME

or

`top.frames[0]` - via its index number in the 'frames' array

The top object refers to the topmost frame in the frame hierarchy, and is followed by the name or array value of the frame to be accessed. And since one of the most common uses of frames is to set up a Web page such that a link in the left frame opens a new HTML document in the right frame, let's do just that in our next example.

First, set up the frameset, which contains two frames named 'lefto' and 'righto':

```
<HTML>
<FRAMESET COLS="15%,*">
    <FRAME NAME="lefto"
        SRC="left.htm" NORESIZE>
    <FRAME NAME="righto"
        SRC="right.htm" NORESIZE>
</FRAMESET>
</HTML>
```

As you can see, we've used the NAME attribute to assign names to the two frames--we'll use these a little further down. Next, create the 'left.htm' file, which looks like this:

```
<html>
<head>
<script LANGUAGE="JavaScript">
function modify(linkname)
{     parent.righto.document.write(linkname);
}
```

```
</script>
</head>
<body>
<a href="javascript:modify('Link 1')">Link 1</a><br>
<a href="javascript:modify('Link 2')">Link 2</a>
</body>
</html>
```

Open the Frameset file. When you click on a Link in the left frame, its name appears in the right frame.

Web Resources

The following is a list of Web-Sites where you can find useful content on Web Site Designing:

Free Web Site Hosting:

http://www.tripod.com, http://www.geocities.com, http://www.xoom.com, etc.

Virtual URLs:

The above sites provide you very long URLs. For example, a Web Site hosted at Tripod might have a name like http://members.tripod.com/yourname. Virtual URLs such as the one by http://welcome.to (e.g. http://welcome.to/yourname) , http://www.cjb.net (e.g. http://www.yourname.cjb.net), etc. provide you more shorter and attractive URLs.

Counters, Guestbooks, etc.:

For all Site Management Functions, I find http://www.bravenet.com the best. It provides free counters, Guestbooks, Site specific Search, Advertising, etc.

HTML:

The best HTML tutorial can be found at http://www.timescomputing.com.

DHTML:

Visit http://dynamicdrive.com

JavaScript:

The following sites have excellent JavaScript ready to use. http://www.javascript.com, http://wsabstract.com, http://www.javaboutique.internet.com.

And, last but not the least, http://www.anandvishwanathan.com, my own Web Site is always present to act as your Guide.