CHAPTER 04 (3 Hours)

Web Basics

CHAPTER 5 (6 Hours)

Web Development with HTML and DHTML

BE (Computer) - IV

Note: This handout is for simple reference only. Do not completely depend on it.

CHAPTER 04 – WEB BASICS

A **web browser** or **Internet browser** is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content Hyperlinks present in resources enable users to easily navigate their browsers to related resources.

In order of release:

- Netscape Navigator and Netscape Communicator, October 13, 1994
- Internet Explorer 1, August 16, 1995
- Opera, 1996
- Mozilla Navigator, June 5, 2002
- Safari, January 7, 2003
- Mozilla Firefox, November 9, 2004
- Google Chrome, September 2, 2008

Web Servers

Web servers are computers that deliver (*serves up*) Web pages. Every Web server has an IP address and possibly a domain name. Web servers are computers on the Internet that host websites, serving pages to viewers upon request. This service is referred to as web hosting. Every web server has a unique address so that other computers connected to the Internet know where to find it. The Internet Protocol (IP) address looks something like this: 69.93.141.146. This address maps to a more human friendly address, such as http://www.sarojpandey.com.np. Web hosts rent out space on their web servers to people or businesses to set up their own websites. The web server allocates a unique website address to each website it hosts.

For example, if you enter the URL *http://www.sarojpandey.com.np/index.html* in your browser, this sends a request to the Web server whose domain name is *sarojpandey.com.np*. The server then fetches the page named *index.html* and sends it to your browser.

Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. Two leading Web servers are Apache, the most widely installed Web server, and Microsoft's Internet Information Server (IIS).

Apache HTTP Server FREE & Open Source

Apache HTTP Server (also referred to as simply "Apache") has, at the time of writing, been the most popular web server on the web since 1996. Apache is developed and maintained by the Apache Software Foundation, which consists of a decentralized team of developers. The software is produced under the Apache license, which makes it free and open source. Apache is available for a range of operating systems, including Unix, Linux, Novell Netware, Windows, Mac OS X, Solaris, and FreeBSD.

Nginx FREE & Open Source

Nginx (pronounced "engine X") is the second most popular open source web server currently on the Internet. Though development only started in 2002, its currently used by over 6% of web domains. It is a lightweight HTTP server, and can also serve as a reverse proxy and IMAP/POP3 proxy server. It's licensed under a BSD-like license. It runs on UNIX, GNU/Linux, BSD, Mac OS X, Solaris, and Windows.

Nginx was built with performance in mind, in particular to handle ten thousand clients simultaneously. Instead of using threads to handle requests, like traditional servers, Nginx uses an event-driven (asynchronous) architecture. Its more scalable and uses less, and more predictable, amounts of memory. In addition to the basic HTTP features, Nginx also supports name-based and IP-based virtual servers, keep-alive and pipelined connections, and FLV streaming. It can also be reconfigured and upgraded online without interruption of the client processing.

Lighttpd FREE & Open Source

Lighttpd (pronounced "lighty") is the third most popular open source web server. This lightweight server was initially released in 2003 and currently serves less than 1% of web domains. It's licensed under a revised BSD license and runs on Unix and Linux.

Like nginux, lighttpd is a lightweight server built for performance with a goal of handling ten thousand clients simultaneously. It also uses an event-driven (asynchronous) architecture.

Tornado FREE & Open Source

Tornado is a scalable non-blocking web server, meant to be used for real-time web services. According to the website, it is able to handle thousands of simultaneous standing connections. It has built-in cross-site request

forgery (or XSRF) protection, aggressive file caching (to improve performance), support for user interface (UI) modules, etc. The server is available in source form and requires Python to be installed.

Microsoft Personal Web Server (**PWS**) **(Proprietary)** is a scaled-down web server software for Windows operating systems. It has fewer features than Microsoft's Internet Information Services (IIS) and its functions have been superseded by IIS and Visual Studio. Microsoft officially supports PWS on Windows 95, Windows 98, Windows 98 SE, and Windows NT 4.0. Prior to the release of Windows 2000, PWS was available as a free download as well as included on the Windows distribution CDs. PWS 4.0 was the last version and it can be found on the Windows 98 CD and the Windows NT 4.0 Option Pack.

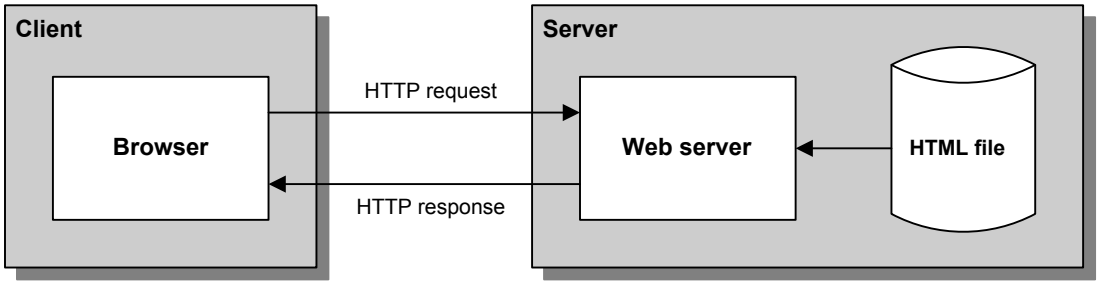
Microsoft Internet Information Services (IIS) (Proprietary)

IIS is, at the time of writing, the second most popular web server on the web. It is however, gaining market share, and if the current trend continues, it won't be long before it overtakes Apache.

IIS comes as an optional component of most Windows operating systems. You can install IIS by using *Add/Remove Windows Components* from *Add or Remove Programs* in the Control Panel.

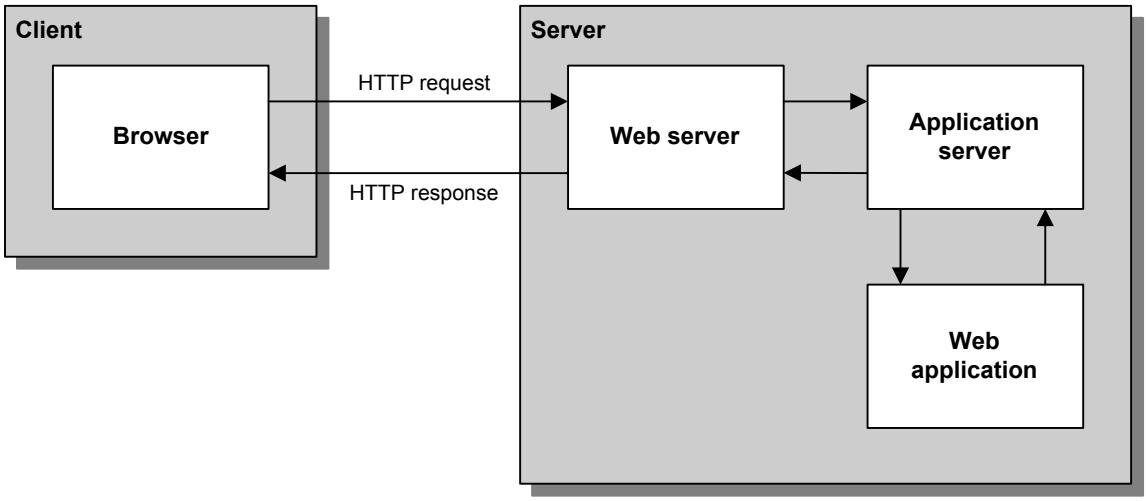
Types of Web Pages and its Processing

Static web page



- A *static web page* is an HTML document that is the same each time it's viewed. It doesn't change in response to user input.
- Static web pages are usually simple HTML files that are stored on the web server with a file extension of .htm or .html. When a browser requests a static web page, the web server retrieves the file from disk and sends it back to the browser.
- A web browser requests a page from a web server by sending the server an HTTP message known as an HTTP *request*.
- The HTTP request includes, among other things, the name of the HTML file being requested and the Internet address of both the browser and the web server.
- A user working with a browser can initiate an HTTP request in several ways. One way is to type the address of a web page, called a *URL*, or *Uniform Resource Locator*, into the browser's address area and press Enter. Another way is to click a link that refers to a web page.
- A web server replies to an HTTP request by sending a message known as an *HTTP response* back to the browser.
- The HTTP response contains the addresses of the browser and the server as well as the HTML document that's being returned.

Dynamic web page



- A *dynamic web page* is an HTML document that's generated by a web application. Often, the web page changes according to information that's sent to the web application by the browser.
- When a web server receives a request for a dynamic web page, the server passes the request to an *application server*.
- The application server executes the web application, which generates an HTML document. This document is returned to the application server, which passes it back to the web server. The web server, in turn, sends the document back to the browser.
- After the page is displayed, the user can interact with it using its controls. Some of those controls let the user *post* the page back to the server, so it's processed again using the data the user entered.
- Each application mapping specifies which application should be run to process files with that extension.
- If the file extension isn't in the list of application mappings, the requested file is returned to the browser without any processing.
- The process that begins with the user requesting a web page and ends with the server sending a response back to the client is called a *round trip*.
- After a web application generates an HTML document, it ends. Then, unless the data the application contains is specifically saved, that data is lost.

HTTP Overview

- HTTP, Hyper Text transfer Protocol is the standard Web transfer protocol.
- The primary technology protocol on the Web that allows linking and browsing.
- The HTTP is the language that Web clients and Web servers use to communicate with each other.
- It is essentially the backbone of the web.
- It is constantly evolving protocol with several versions in use and others are still under development.
- This protocol has two items: the set of requests from browsers to servers and the set of responses going back the other way.
- It is a stateless protocol and does not maintain any information from one transaction to the next, so the next transaction needs to start all over again
- The advantage is that an HTTP server can serve a lot more clients in a given period of time, since there's no additional overhead for tracking sessions from one connection to the next.
- Default Port used by HTTP is 80.

HTTPS

- HTTPS stands for Hyper Text Transfer Protocol Secure. It is a secured version of the HTTP protocol that uses SSL (protocol primarily developed with secure, safe Internet transactions in mind as the encryption layer. SSL layer also offers strong authentication methods.
- HTTPS allows secure ecommerce transactions, such as online banking.
- HTTPS connects on port 443, while HTTP is on port 80
- HTTPS encrypts the data sent and received with SSL, while HTTP sends it all as plain text.
- When a user connects to a website via HTTPS, the website encrypts the session with a digital certificate. A user can tell if they are connected to a secure website if the website URL begins with https:// instead of http://.
- Secure Sockets Layer uses a cryptographic system that encrypts data with two keys.
- SSL transactions are negotiated by means of a key based encryption algorithm between the client and the server, this key is usually either 40 or 128 bits in strength (the higher the number of bits the more secure the transaction).
- When a SSL Digital Certificate is installed on a web site, users can see a padlock icon at the bottom area of the navigator. When an Extended Validation Certificates is installed on a web site, users with the latest versions of Firefox, Internet Explorer or Opera will see the green address bar at the URL area of the navigator.

[***Note***: HTTPS should not be confused with S-HTTP, a security-enhanced version of HTTP. SSL and S-HTTP have very different designs and goals so it is possible to use the two protocols together. Whereas SSL is designed to establish a secure connection between two computers, S-HTTP is designed to send individual messages securely.]

Why Is A SSL Certificate Required?

With booming Internet trends and fraud, most will not submit their private details on the web unless they know that the information they provide is securely transmitted and not accessible for anyone to view.

HTTP Transaction

All HTTP transactions follow the same general format. Each client request and server response has three parts: the request or response line, a header section, and the entity body. ***The client initiates a transaction as follows:***

1. The client contacts the server at a designated port number (by default, 80). Then it sends a document request by specifying an HTTP command called a *method*, followed by a document address, and an HTTP version number. For example:

   GET /index.html HTTP/1.0

   Uses the GET method to request the document *index.html* using version 1.0 of HTTP.

2. Next, the client sends optional header information to inform the server of its configuration and the document formats it will accept. All header information is given line by line, each with a header name and value. For example, this header information sent by the client indicates its name and version number and specifies several document preferences:

   User-Agent: Mozilla/2.02Gold (WinNT; I)
   Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

   The client sends a blank line to end the header.

3. After sending the request and headers, the client may send additional data. CGI programs using the POST method mostly use this data. Clients like Netscape Navigator-Gold, to publish an edited page back onto the Web server, may also use it.

The server responds in the following way to the client's request:

1. The server replies with a status line containing three fields: HTTP version, status code, and description. The HTTP version indicates the version of HTTP that the server is using to respond. The status code is a three digit number that indicates the

server's result of the client's request. The description following the status code is just human-readable text that describes the status code. For example, this status line:

HTTP/1.0 200 OK

- Indicates that the server uses version 1.0 of HTTP in its response. A status code of 200 means that the client's request was successful and the requested data will be supplied after the headers.

2. After the status line, the server sends header information to the client about itself and the requested document. For example:

Date: Fri, 20 Sep 1996 08:17:58 GMT
Server: NCSA/1.5.2
Last-modified: Mon, 17 Jun 1996 21:53:08 GMT
Content-type: text/html
Content-length: 2482

A blank line ends the header.

3. If the client's request is successful, the requested data is sent. This data may be a copy of a file, or the response from a CGI program. If the client's request could not be fulfilled, additional data may be a human-readable explanation of why the server could not fulfill the request.

In HTTP 1.0, after the server has finished sending the requested data, it disconnects from the client and the transaction is over unless a *Connection: Key Admin* header is sent. In HTTP 1.1, however, the default is for the server to maintain the connection and allow the client to make additional requests.

Being a stateless protocol, HTTP does not maintain any information from one transaction to the next, so the next transaction needs to start all over again. The advantage is that an HTTP server can serve a lot more clients in a given period of time, since there's no additional overhead for tracking sessions from one connection to the next.

Client-side scripting is the class of computer programs on the web that are executed in *client-side*, by the user's web browser, instead of *server-side* (on the web server). This type of computer programming is an important part of the Dynamic HTML (DHTML) concept, enabling web pages to be scripted; that is, to have different and changing content depending on user input, environmental conditions (such as the time of day), or other variables. Web authors write client-side scripts in languages such as JavaScript (Client-side JavaScript) and VBScript.

Client-side scripts are embedded within an HTML document (hence known as an "embedded script"), but they may also be contained in a separate file, which is referenced by the document (or documents) that uses it (known as an

"external script"). Upon request, the necessary files are sent to the user's computer by the web server (or servers) on whom they reside. The user's web browser executes the script, and then displays the document, including any visible output from the script. By viewing the file that contains the script, users may be able to see its source code.

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where the viewing web browser, usually in JavaScript, runs scripts. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements or queries into data stores. Server side scripts are written in languages such as Perl, PHP, ASP.NET etc.

SSS produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed.

From security point of view, server-side scripts are never visible to the browser as these scripts are executed on the server and emit HTML corresponding to user's input to the page.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.

Comparison: Client-side scripts have greater access to the information and functions available on the user's browser, whereas server-side scripts have greater access to the information and functions available on the server. Server-side scripts require that their languages interpreter be installed on the server, and produce the same output regardless of the client's browser, operating system, or other system details. Client-side scripts do not require additional software on the server; however, they do require that the user's web browser understands the scripting language in which they are written.

FTP & Its Types

***F**ile **T**ransfer **P**rotocol,* the protocol for exchanging files over the Internet. FTP works in the same way as HTTP. FTP uses the Internet's TCP/IP protocols to enable data transfer. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server (e.g., uploading a Web page file to a server).

- File Transfer Protocol (FTP) is a method of transferring files from a client to a server or vice versa.
- Files are transferred over the Internet using TCP/IP protocol.
- FTP is old-time protocol that maintains two simultaneous connections.
- The first connection uses the telnet remote login protocol to log the client into an account and process commands via the *protocol interpreter*.
- The second connection is used for the *data transfer process*.
- Whereas the first connection is maintained throughout the FTP session, the second connection is opened and closed for each file transfer.
- The FTP protocol also enables an FTP client to establish connections with two servers and to act as the third-party agent in transferring files between the two servers.
- FTP servers rarely change, but new FTP clients appear on a regular basis.
- Although FTP is a command-line oriented protocol, the new generation of FTP clients hides this orientation under a GUI environment.
- A client makes a TCP connection to the server's port 21. This connection, called the *control connection*, remains open for the duration of the session, with a second connection, called the *data connection*, either opened by the server from its port 20 to a negotiated client port or opened by the client from an arbitrary port to a negotiated server port as required to transfer file data.

[FTP is a TCP based service exclusively. There is no UDP component to FTP. FTP is an unusual service in that it utilizes two ports, a 'data' port and a 'command' port (also known as the control port). Traditionally these are port 21 for the command port and port 20 for the data port. The confusion begins however, when we find that depending on the mode, the data port is not always on port 20.]

Types of FTP

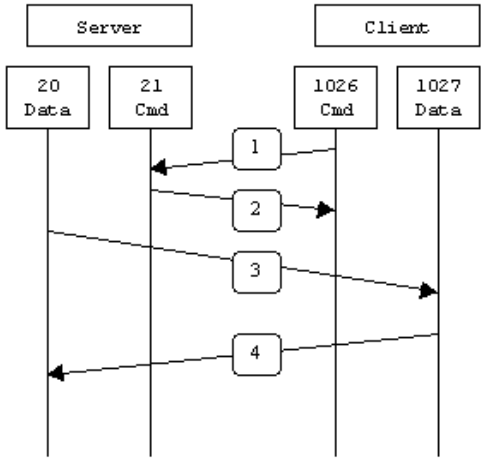
- Active FTP

In active mode FTP the client connects from a random unprivileged port (N > 1023) to the FTP server's command port, port 21. Then, the client starts listening to port N+1 and sends the FTP command PORT N+1 to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20.

From the server-side firewall's standpoint, to support active mode FTP the following communication channels need to be opened:

- FTP server's port 21 from anywhere (Client initiates connection)
- FTP server's port 21 to ports > 1023 (Server responds to client's control port)
- FTP server's port 20 to ports > 1023 (Server initiates data connection to client's data port)
- FTP server's port 20 from ports > 1023 (Client sends ACKs to server's data port)

When drawn out, the connection appears as follows:



- In step 1, the client's command port contacts the server's command port and sends the command PORT 1027.
- The server then sends an ACK back to the client's command port in step 2.
- In step 3 the server initiates a connection on its local data port to the data port the client specified earlier.
- Finally, the client sends an ACK back as shown in step 4.

The main problem with active mode FTP actually falls on the client side. The FTP client doesn't make the actual connection to the data port of the server - it simply tells the server what port it is listening on and the server connects back to the specified port on the client. From the client side firewall this appears to be an outside system initiating a connection to an internal client - something that is usually blocked.

- Passive FTP

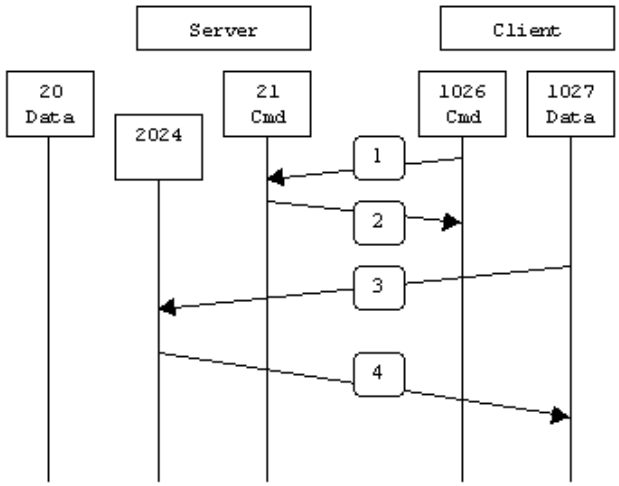
In order to resolve the issue of the server initiating the connection to the client a different method for FTP connections was developed. This was known as passive mode, or PASV, after the command used by the client to tell the server it is in passive mode.

In passive mode FTP the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally (N > 1023 and N+1). The first port contacts the server on port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port (P > 1023) and sends the PORT P command back to the client. The client then initiates the connection from port N+1 to port P on the server to transfer data.

From the server-side firewall's standpoint, to support passive mode FTP the following communication channels need to be opened:

- FTP server's port 21 from anywhere (Client initiates connection)
- FTP server's port 21 to ports > 1023 (Server responds to client's control port)
- FTP server's ports > 1023 from anywhere (Client initiates data connection to random port specified by server)
- FTP server's ports > 1023 to remote ports > 1023 (Server sends ACKs (and data) to client's data port)

When drawn, a passive mode FTP connection looks like this:



- In step 1, the client contacts the server on the command port and issues the PASV command.
- The server then replies in step 2 with PORT 2024, telling the client which port it is listening to for the data connection.
- In step 3 the client then initiates the data connection from its data port to the specified server data port. Finally, the server sends back an ACK in step 4 to the client's data port.

- While passive mode FTP solves many of the problems from the client side, it opens up a whole range of problems on the server side. The biggest issue is the need to allow any remote connection to high numbered ports on the server. Fortunately, many FTP daemons, including the popular WU-FTPD allow the administrator to specify a range of ports, which the FTP server will use. The second issue involves supporting and troubleshooting clients, which do (or do not) support passive mode. FTP Clients like FileZilla give user option to select the FTP mode.

With the massive popularity of the World Wide Web, many people prefer to use their web browser as an FTP client. Most browsers only support passive mode when accessing ftp:// URLs. This can either be good or bad depending on what the servers and firewalls are configured to support.

CHAPTER 05 – WEB DEVELOPMENT WITH HTML & DHTML

What is HTML?

- HTML or **H**yper**T**ext **M**arkup **L**anguage is designed to specify the logical organization of a document, with important hypertext extensions.
- HTML instructions divide the text of a document into blocks called *elements*.
- These can be divided into two broad categories:
- Those that define how the **BODY** of the document is to be displayed by the browser, and
- Those that define information about the document, such as the **title** or relationships to other documents.
- The detailed rules for HTML (the names of the tags/elements, how they can be used) are defined using another language known as the SGML (**S**tandard **G**eneralized **M**arkup **L**anguage).
- HTML is a set of special codes that can be embedded in text to add formatting and linking information.
- HTML is the language interpreted by a Browser.
- The HTML file must have an extension "**.htm**" or "**.html**".
- Any text editor can be used to create HTML file.

HTML stands for **H**yper **T**ext **M**arkup **L**anguage. HTML is a presentation language. We use HTML language to display information according to our need. An HTML file is a text file containing small **markup tags**. The markup tags tell the Web browser **how to display** the page. An HTML file must have an .**htm** or .**html** file extension. An HTML file can be created using a **simple text editor**.

HTML is very popular language used on web because of its interoperability. HTML language is platform independent i.e. HTML files can be opened on any platform. HTML files can be written using a simple text editor like notepad which is present in all the operating system.

HTML language is used to create web pages. Web pages can be viewed using application software called a **Web browser**. Popular browsers are **Internet Explorer, Mozilla Firefox and Netscape Navigator**. A web browser parses the HTML file containing markups (html tags) and displays the information with the proper format as specified in the HTML document. HTML tags are also called mark-up. HTML tags are surrounded by the two characters < and >. The surrounding characters are called angle brackets. HTML tags normally come in pairs like and . The first tag in a pair is the start tag, the second tag is the end tag. The text between the start and end tags is the element content. HTML tags are not case sensitive; means the same as <B>.

Structure of an HTML document is shown below:

```
<html>
<head>
    <title> Title of page </title>
</head>
<body>
    This is the place where the information to be displayed in a web page is written.
</body>
</html>
```

Every html document must start with <html> tag. It shows the starting of HTML document. <head> tag contains information like the title of the document and other information which describes about the content of the document.

BODY part of a HTML document contains the information and its format to be displayed by the browser. HEAD part of a HTML document contains the information that is not displayed on the browser window. It defines information `about' the document, such as the title or relationships to other documents.

<body> tag is the place where we write all the information that is to be displayed in the web browser. It also contains other tags which defines how information are to be displayed in the web browser. <body> tag shows the starting of the body tag and </body> tag shows the ending of the body tag. </html> shows the ending of the HTML document. Every ending tag must have a forward slash as shown in </html> tag.

HTML discards whitespaces. HTML only considers a single space as a space. The browser automatically discards rest of the whitespace. Hence, we can use as much whitespace as we want while creating our HTML document. This makes html document easy to read or edit.

Versions of HTML

HTML 2.0

- It set the standard for core HTML features based upon current practice in 1994.

HTML 3.2

- W3C's first Recommendation for HTML which represented the consensus on HTML features for 1996.
- **HTML 3.2** added widely-deployed features such as tables, applets, text-flow around images, superscripts and subscripts, while providing backwards compatibility with the existing HTML 2.0 standard.

HTML 4.0

- First released as a W3C Recommendation on 18 December 1997.
- A second release was issued on 24 April 1998 with changes limited to editorial corrections.
- This specification has now been superseded by HTML 4.01.

HTML 4.01

- HTML 4.01 is the current official standard.
- It includes support for most of the proprietary extensions, plus support for extra features (Internationalized documents, support for **C**ascading **S**tyle **S**heets, extra TABLE, FORM, and JavaSctipt enhancements), that are not universally supported.
- This is the last version of HTML.
- After this XHTML was released which stands for e**X**tensible **H**yper**T**ext **M**arkup **L**anguage.

HTML 5.0

- This is the new version of HTML with many exciting new features. This version is still under development.

HTML Elements/Tags

- The HTML instructions, along with the text to which the instructions apply, are called HTML *elements*.
- The HTML instructions are themselves called ***tags***, and look like <element_name> -- that is, they are simply the element name surrounded by left and right angle brackets.
- The content in the web-page is written after the starting tag, and closed with the end tag.
- E.g: **<element_name>** text to be written HERE **</element_name>**
- The end tag has slash character in front of it.
- HTML tags are not case sensitive; **<b>** means same as **<B>**.

Empty Elements

- Some elements are *empty* -- that is, they do not affect a block of the document in some way.
- These elements do not require an ending *tag*.
  - An example is the <HR> element, which draws a horizontal line across the page.

HTML Tag Attributes

- Many elements can have arguments that pass parameters to the interpreter handling the element.
- These arguments are called *attributes* of the element.
- An attribute is a customizable option for a tag.
- In other words, attributes are used to define the properties of a tag.
    - Example: **<p** align = "left"**>** Trial Example **</p>**.
    - In the above example the align attribute allows you to specify how text in a paragraph is arranged on the page.
- Not all tags support attributes.
- Some tags support multiple attributes, and the attributes are listed one after another in the start tag, seperated by space.
- Attributes are always set to the opening tag.

HTML4.01 Tags Lists

TITLE	TAG	DESCRIPTION
Basic Elements		
Document Type	<HTML> </HTML>	document root element, beginning and end of file
Title	<TITLE> </TITLE>	document title, must be in header
Header	<HEAD> </HEAD>	descriptive info, such as title
Body	<BODY> </BODY>	bulk of the page, notes body of document
Formatting		
Bold	<B> </B> or	bold text style
Italic	<I> </I>	italic text style
Underline	<U> </U>	underlined text (not widely implemented)
Strikeout	<STRIKE> </STRIKE>	strike-through text (not widely implemented)
Strikeout	<S> </S>	strike-through text (not widely implemented)
Subscript		subscript numbers like footnotes
Superscript		superscript numbers like cross - reference numbers

Pre formatted	<PRE> </PRE>	pre formatted text (display text spacing as-is)
Center	<CENTER> </CENTER>	centers text and images
Blinking	<BLINK> </BLINK>	blinking text, Netscape only
Font Size	<FONT SIZE=?> </FONT>	local font size(ranges from 1-7)
Change Font Size	<FONT SIZE="+|-?"> </FONT>	controls font size rendered
Font Color	<FONT COLOR="#$$$$$$"> </FONT>	controls font color rendered
Select Font	<FONT FACE="***"> </FONT>	the style of the text, such as Times New Roman
Marquee	<MARQUEE> </MARQUEE>	scrolling text (IE only)
Links		
Link Something	<A HREF="URL"> </A>	links text or graphic to another URL
Link to Location	<A HREF="URL#***"> </A>	links text or graphic an anchor in an other document
Link to Location in Current Page	<A HREF="#***"> </A>	links text or graphic an anchor in current document
Target Window	<A HREF="URL" TARGET="***"> </A>	links text or graphic to a URL in a new browser widow
Action on Click	<A HREF="URL" ONCLICK="***"> </A>	takes effect when user clicks on the item (Javascript)
Mouseover Action	<A HREF="URL" ONMOUSEOVER="***"> </A>	takes effect when user moves pointer over item
Link to Email	<A HREF="mailto:@"> </A>	creates blank e-mail to indicated address with visitor's default e-mail client
Graphics and Sound		
Display Image	<IMG SRC="URL">	displays image from the indicated URL
Alignment	<IMG SRC="URL" ALIGN= TOP|BOTTOM|MIDDLE|LEFT|RIGHT>	aligns the image
Dimensions	<IMG SRC="URL" WIDTH=? HEIGHT=?>	the dimensions, in pixels, of the image
Border	<IMG SRC="URL" BORDER=?>	border, in pixels, around the image

Dividers		
Paragraph	<P> </P>	paragraph (closing tag often unnecessary)
Align Text	<P ALIGN=LEFT|CENTER|RIGHT> </P>	aligns paragraph
Justify Text	<P ALIGN=JUSTIFY> </P>	justify's paragraph's text
Line Break	 	a single carriage return
Horizontal Rule	<HR>	horizontal line
Alignment	<HR ALIGN=LEFT|RIGHT|CENTER>	alignment of horizontal line
Thickness	<HR SIZE=?>	thickness, in pixels, of horizontal line
Width	<HR WIDTH=?>	width, in pixels, of horizontal line
Width Percent	<HR WIDTH="%">	width(as a percentage of page width), in pixels, of horizontal line
Solid Line	<HR NOSHADE>	horizontal line without the 3D cutout look
No Break	<NOBR> </NOBR>	prevents line breaks
Structural Elements		
Heading	<H?> </H?>	document header, the ? defines 6 levels (#'s 1-6)
Strong Emphasis	<STRONG> </STRONG>	strongly emphasized text, usually displayed as bold
Address	<ADDRESS> </ADDRESS>	author information
Large Font Size	<BIG> </BIG>	uses a large text size
Small Font Size	<SMALL> </SMALL>	use a small text size
Backgrounds		
Tiled Background	<BODY BACKGROUND= "URL">	causes the image to tile as the background of the page
Watermark	<BODY BGPROPERTIES= "FIXED">	Static image which remains in the same location as visitors scroll.
Background Color	<BODY BGCOLOR= "#$$$$$$">	solid background color of the page

Text Color	<BODY TEXT="#$$$$$$">	color of the text throughout the page
Link Color	<BODY LINK="#$$$$$$">	color of all links throughout the page
Visited Link	<BODY VLINK="#$$$$$$">	color of all links that have already been clicked on by visitor
Active Link	<BODY ALINK="#$$$$$$">	color of link while being selected
Lists		
Unordered List	<UL> </UL>	list with bulleted items
List Item	<LI> </LI>	indicates an item on the list
Bullet Type	<UL TYPE=DISC|CIRCLE|SQUARE>	shape of bullet for the whole list
Bullet Type	<LI TYPE=DISC|CIRCLE|SQUARE>	shape of bullet for specific list item
Ordered List	<OL> <LI> </OL>	numbered list
Numbering Type	<OL TYPE=A|a|I|i|1>	type of numbering for the whole list
Numbering Type	<LI TYPE=A|a|I|i|1>	type of numbering for specific list item
Starting Number	<OL START=?>	starting number for list
Starting Number	<LI VALUE=?>	starting number for this & subsequent items
Definition List	<DL> </DL>	a list of definitions
Definition Term	<DT> </DT>	definition term
Definition	<DD> </DD>	definition of a term
Menu List	<MENU> </MENU>	display menu type list
Directory List	<DIR> </DIR>	directory link
Tables		
Define Table	<TABLE> </TABLE>	signals the beginning of a table
Table Alignment	<TABLE ALIGN= LEFT|RIGHT|CENTER>	aligns the table within the browser window
Table Border	<TABLE BORDER=?> </TABLE>	border of table, you can set the value (aka width)
Cell Spacing	<TABLE CELLSPACING=?>	places specific amount of space between the individual cells within a table

Cell Padding	<TABLE CELLPADDING=?>	places specific amount of space between the cells border and its contents
Desired Width	<TABLE WIDTH=?>	width of table in pixels
Width Percent	<TABLE WIDTH=%>	width of table in percentage of page
Table Color	<TABLE BGCOLOR="#$$$$$$"> </TABLE>	overall background color of table
Border Color	<TABLE BORDERCOLOR="#$$$$$$"> </TABLE>	the color of the table border
Table Row	<TR> </TR>	table row
Alignment	<TR ALIGN= LEFT| RIGHT| CENTER| MIDDLE| BOTTOM>	alignment of the table row
Table Cell	<TD> </TD>	specific table cell, must appear within table rows
Alignment	<TD ALIGN= LEFT|RIGHT|CENTER VALIGN= TOP|MIDDLE|BOTTOM>	alignment of the table cell
Columns to Span	<TD COLSPAN=?>	identifies the the number of columns the cell should span
Rows to Span	<TD ROWSPAN=?>	identifies the the number of rows the cell should span
Desired Width	<TD WIDTH=?>	width of cell in pixels
Width Percent	<TD WIDTH="%">	width of cell as percentage of table
Cell Color	<TD BGCOLOR="#$$$$$$">	background color of table cell
Header Cell	<TH> </TH>	table cell for header information (bold & centered)
Alignment	<TH ALIGN= LEFT| RIGHT| CENTER| MIDDLE|BOTTOM>	alignment of the header cell
Table Body	<TBODY>	identifies the specific body section of the table
Table Footer	<TFOOT> </TFOOT>	separates group of cells to serve as footer material for the table (must come before <THEAD>)
Table Header	<THEAD> </THEAD>	separates group of cells to serve as header material for the table

Table Caption	<CAPTION> </CAPTION>	caption for a table
Alignment	<CAPTION ALIGN=TOP|BOTTOM|LEFT|RIGHT>	alignment for the caption of a table
Frames		
Frame Document	<FRAMESET> </FRAMESET>	creates layouts of frames (instead of <BODY>)
Row Heights	<FRAMESET ROWS=,,,> </FRAMESET>	comma separated list of size of each row within the frameset (pixels or %)
Column Widths	<FRAMESET COLS=,,,> </FRAMESET>	comma separated list of size of each column within the frameset (pixels or %)
Borders	<FRAMESET FRAMEBORDER= "yes|no"> </FRAMESET>	identifies if a frame has a visible border or not
Border Width	<FRAMESET BORDER=?> </FRAMESET>	width of frame border if visible
Border Color	<FRAMESET BORDERCOLOR="#$$$$$$"> </FRAMESET>	color of frame border if visible
Frame Spacing	<FRAMESET FRAMESPACING=?> </FRAMESET>	number of pixels of reserved space between frames
Define Frame	<FRAME>	specific contents of an individual frame
Display Document	<FRAME SRC="URL">	identifies the initial contents of the frame
Frame Name	<FRAME NAME="***"| _blank| _self| _parent |_top>	assigns a name to the current frame
Margin Width	<FRAME MARGINWIDTH=?>	distance between content and frame's left and right margins
Margin Height	<FRAME MARGINHEIGHT=?>	distance between content and frame's top and bottom margins
Scroll bar	<FRAME SCROLLING="YES|NO|AUTO">	controls how the window is or isn't scrolled

Not Re-sizable	<FRAME NORESIZE>	prohibits the document viewer from changing dimensions of the frame
Borders	<FRAME FRAMEBORDER="yes|no">	controls wither frame has a border
Border Color	<FRAME BORDERCOLOR="#$$$$$$">	color of border of frame

HTML Lists

- HTML provides three type of lists.
- They are listed below:

1. **Ordered List:**
   - A list of multi-line paragraphs, listed separately and ordered numerically in some way.
   - The list items are marked with numbers.
   - <OL ...> creates an ordered list.
   - "Ordered" means that the order of the items in the list is important.
   - By default, the number starts with 1,2,3…….
   - An ordered list starts with the <ol> tag.
   - Each list item starts with the <li> tag.
     - Example:

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

2. **Unordered List:**
   - A list of multi-line paragraphs, listed separately and usually marked by a bullet or similar symbol (Unordered List)
   - <UL ...> creates an unordered list.
   - The *unordered* part means that the items in the list are not in any particular order.

- The list items are marked with bullets (typically small black circles).
- An unordered list starts with the <ul> tag.
- Each list item starts with the <li> tag.
  - Example:

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee
- Milk

3. Definition List:

- A definition list is **not** a list of items.
- This is a list of terms and explanation of the terms.
- A definition list starts with the <dl> tag.
- Each definition-list term starts with the <dt> tag.
- Each definition-list definition starts with the <dd> tag.
  - Example:

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

Coffee
  Black hot drink
Milk
  White cold drink

Frames

- Frames allow displaying more than one web-page in a single browser at a same instance of time.
- HTML tags <frameset>……..</frameset> is used to divide a browser screen into two or more HTML recognizable unique regions.
- Each unique region is called frame.
- Each frame can be loaded with a different document and hence, allow multiple HTML documents to be seen concurrently.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page.

The major advantages of using frames are:

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

The Frameset Tag

- The <frameset> tag defines how to divide the window into frames.
- Each <frameset> defines a set of rows **or** columns.
- The <frameset> tags require one of the following two attributes depending on whether the screen has to be divided into rows or columns.

The two attributes are:

1. Rows:

- This attribute is used to divide the screen into multiple rows.
- The each row can be set with different values depending on the required size of the row.

2. Cols:

- This attribute is used to divide the screen into multiple columns.
- The values for both Rows and Cols can be:
    - A number in pixels ***(Not commonly used.)***

- Expressed as a percentage of the screen resolution.
- The symbol *, which indicates the remaining space.

Example of <frameset> tag:

<frameset rows="33%,33%,*"**>**

divides the browser screen into 3 equal horizontal sections.

<frameset cols="33%,*"**>**

divides the browser screen into 2 different vertical sections.

The <frame> Tag

- Once the screen is divided into rows and columns, each unique section can be loaded with different HTML documents.
- This is achieved by using the <frame> tag.
- The **<frame>** tag defines what HTML document to put into each frame.
- The attributes of the **<frame>** tag are:

Attributes	Description
SRC="url"	Indicates the url of the document to be loaded into the frame.
MARGINHEIGHT="n"	Specifies the amount of white space to be left at the top and bottom of the frame.
MARGINWIDTH="n"	Specifies the amount of white space to be left along the sides of the frame.
NAME="name"	Gives the same unique name so it can be targeted by other documents. The name given must begin with an alphanumeric character.
NORESIZE	Disables the frames resizing capability.
Scrolling	Controls the appearance of horizontal and vertical scrollbars in a frame. This takes the values YES/NO/AUTO.

Example of using <frame> tag:

<frameset cols="25%,75%"**>**

```
    <frame src="frame_a.htm">
    <frame src="frame_b.htm">
</frameset>
```

In the example above we have a frameset with two columns.

- The first column is set to 25% of the width of the browser window.
- The second column is set to 75% of the width of the browser window.
- The HTML document "frame_a.htm" is put into the first column, and the HTML document "frame_b.htm" is put into the second column.

HTML Links

- A link is a connection from one Web resource to another.
- A *link* has two ends -- called *anchors* -- and a direction.
- The link starts at the "source" anchor and points to the "destination" anchor, which may be any Web resource (e.g., an image, an HTML document, an element within an HTML document, etc.).
- The text or an image that provides such linkages is called hypertext, hyperlink, or hotspot.

What is Hyperlink?

- A Hyperlink is a connection between an HTML element such as text, an image, or anything else on a page and other resource.
- That link might be to another web page, an external image, or an e-mail address.

Difference between Hyperlink and Normal HTML Text:

- Appears in blue color.
    - The default color setting in a browser for hyperlink text or image.
    - The color can be set dynamically via HTML program if required.
- The Hyperlink text/image is underlined.
- When the mouse cursor is placed over it, the standard arrow shaped mouse cursor changes to the shape of a hand.

Changing the color of Links:

- To change the link color there are three attributes that can be specified with the **<body>** tag.
- These are:
    - LINK (Normal)

- ALINK (Active)
- VLINK (Visited)

Types of Hyperlink

There are three types of Hyperlinks:

1. **Inter-page Hyperlink**

- In this type of link the control flows from one-page to another.

Example:

<a HREF="myExample.htm"> Click for Example ****

You can specify the relative as well as the absolute path of the file that you want to call.

2. **Intra-page Hyperlink**

- Intra-page Hyperlink is a link within a same page.
- Sometimes, a jump is required to a different location in the same document.
- Since the jump has to be targeted to a specific location the two steps need to perform.
    a) Identify the location with a name and
    b) Jump to that location using the name.

Example:

<a name ="top"> The HTML text is written here ****

<a HREF="#top"> Goto Top ****

3. **Email Hyperlink**

- This type of Hyperlink is used especially to write e-mail.
- The link does not open any web-pages but opens the outlook express for writing mail.
- You can write the mail and send.

Steps:

a) First type any text like:

Email: info@kcc.edu.np

Surround the email address with the anchor tags i.e. <a>, but instead linking to the web page, use the '*mailto'* command to link it to an e-mail program.

Email: **<a** HREF="mailto: info@kcc.edu.np"> info@kcc.edu.np ****

-> Save the page and view it in browser.

4. **External Links**

You can also have external links like links, when clicking upon them you can jump to next web page.

In such scenario you have to give the path of web page like:

<a HREF="http://www.google.com"> Goto Google ****

HTML Forms

Forms are the most popular way to make web pages interactive. A form on a web page looks similar to a form on a sheet of paper that allows the user to enter requested information and submit it for further processing. A form can have different types of form elements for different purpose like textbox, list box, checkbox, radio buttons, dropdown menus, text area etc.

1. HTML Text Fields

type - Determines what kind of input field it will be. Possible choices are text, submit, and password.

name - Assigns a name to the given field so that you may reference it later.

size - Sets the horizontal width of the field. The unit of measurement is in blank spaces.

maxlength - Dictates the maximum number of characters that can be entered.

HTML Code:

```
<form method="post">
    Name: <input type="text" size="10" maxlength="40" name="name"> <br />
    Password: <input type="password" size="10" maxlength="10" name="password">
</form>
```

'Do not use the password feature for security purposes. The data in the password field is not encrypted and is not secure in any way.'

2. Submit Buttons

HTML Code:

```
<form method="post">
    Name: <input type="text" size="10" maxlength="40" name="name"> <br />
```

Password: <input type="password" size="10" maxlength="10" name="password">

<input type="submit" value="Send">

</form>

3. HTML Radio Buttons

Radio buttons are a popular form of interaction. You may have seen them on quizzes, questionnaires, and other web sites that give the user a multiple-choice question. Below are a couple attributes you should know that relate to the radio button.

value - specifies what will be sent if the user chooses this radio button. Only one value will be sent for a given group of radio buttons.

name - defines which set of radio buttons that it is a part of.

HTML Code:

```
<form method="post">
What kind of shirt are you wearing? <br />
Shade:
<input type="radio" name="shade" value="dark">Dark
<input type="radio" name="shade" value="light">Light <br />
Size:
<input type="radio" name="size" value="small">Small
<input type="radio" name="size" value="medium">Medium
<input type="radio" name="size" value="large">Large <br />
<input type="submit" value="Email Myself">
</form>
```

4. HTML Check Boxes

Check boxes allow for multiple items to be selected for a certain group of choices. The check box's name and value attributes behave the same as a radio button.

HTML Code:

```
<form method="post">
Select your favorite cartoon characters.
```

```
<input type="checkbox" name="toon" value="Goofy">Goofy
<input type="checkbox" name="toon" value="Donald">Donald
<input type="checkbox" name="toon" value="Bugs">Bugs Bunny
<input type="checkbox" name="toon" value="Scoob">Scooby Doo
<input type="submit" value="Email Myself">
</form>
```

5. HTML Drop down Lists (Known as Combo Box)

Drop down menu are created with the <select> and <option> tags. <select> is the list itself and each <option> is an available choice for the user.

HTML Code:

```
<form method="post">
College Degree?
    <select name="degree">
        <option>Choose One</option>
        <option>Some High School</option>
        <option>High School Degree</option>
        <option>Some College</option>
        <option>Bachelor's Degree</option>
        <option>Doctorate</option>
    <input type="submit" value="Email Yourself">
    </select>
</form>
```

TRY IT YOURSELF

HTML Code:

```
<form method="post" action="mailto:youremail@email.com">
Musical Taste
<select multiple name="music" size="4">
    <option value="emo" selected>Emo</option>
    <option value="metal/rock" >Metal/Rock</option>
    <option value="hiphop" >Hip Hop</option>
    <option value="ska" >Ska</option>
    <option value="jazz" >Jazz</option>
    <option value="country" >Country</option>
    <option value="classical" >Classical</option>
    <option value="alternative" >Alternative</option>
    <option value="oldies" >Oldies</option>
    <option value="techno" >Techno</option>
</select>
```

```
<input type="submit" value="Email Yourself">
</form>
```

6. HTML Text Areas

Text areas serve as an input field for viewers to place their own comments onto. Forums and the like use text areas to post what you type onto their site using scripts. For this form, the text area is used as a way to write comments to somebody.

Rows and columns need to be specified as attributes to the <textarea> tag. Rows are roughly 12pixels high, the same as in word programs and the value of the columns reflects how many characters wide the text area will be. i.e. The example below shows a text area 5 rows tall and 20 characters wide.

HTML Code:

```
<form method="post”>
        <textarea rows="5" cols="20" wrap="physical" name="comments">
                        Enter Comments Here
        </textarea>
        <input type="submit" value="Email Yourself">
</form>
```

Note that any text placed between the opening and closing textarea tags will show up inside the text area when the browser views it.

Image Map

A single graphic image containing more than one hot spot is known as Image Map. ***For example,*** imagine a graphic of a bowl of fruit. When you click on a banana, the system displays the number of calories in a banana and when you click on an apple, it displays the number of calories in an apple. Image maps are used extensively on the World Wide Web. Each hot spot in a Web image map takes you to a different Web page. *Image map* is sometimes spelled as one word: ***imagemap****.*

Image Map is defined by each of the sensitive areas in terms of their x and y coordinates (that is, a certain horizontal distance and a certain vertical distance from the left-hand corner of the image). With each set of coordinates, Uniform Resource Locator or Web address is specified and that will be linked to when the user clicks on that area.

The X and Y coordinates are expressed in pixel s either in a separate file called a map file or in the same HTML file that contains the link to the image map. Popular tools like **MapEdit** provide a graphical interface for creating an image map (so that you don't have to figure out the X and Y coordinate numbers yourself).

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas.

The name attribute of the <map> element is required and it is associated with the <img>'s usemap attribute and creates a relationship between the image and the map. The <map> element contains a number of <area> elements, which defines the clickable areas in the image map.

Before Creating Image Map…

- Image

First of all, you need a suitable graphic. You need to be a bit careful when choosing the right image. An image that cannot be divided in different sections is not your best choice, but an image that has clear cut-offs or lines would be better.

-Co-ordinates

When you've chosen or created your desired image, you then insert it into an image editor so that you can find its co-ordinates. Either write different co-ordinates of take them one by one and then insert them into your document. You'll need different co-ordinates depending on what sort of "hot spot" you'd like to use.

Coordinates are defined with ‘coords’ attribute of the <area> tag. ***<area … coords="245,30,10" … />***

- Hot Spot

You've got your image and co-ordinates; HTML comes to finalize your job. There are three different shapes (commonly termed "hot spots") that can be used in image maps. They are as follows:

-RECT
-CIRCLE
-POLYGON

-Rect

Rect is obviously short for rectangle. For this one, you'll need two different co-ordinates. Top right and bottom left.

-Circle

Many people get confused with this one and many don't even know that it exists. All you need to create circle hotspot is the center co-ordinate and it's radius size (Radius is HALF the diameter. Don't get diameter and radius mixed up!).

- Polygon

This is the most common hot spot and used when you can't use neither of the above hot spots. This one uses the co-ordinates of the points in order. So if you're going to have a pentagon (shape with five sides), then you list all five co-ordinates but make sure they're listed in ORDER. One after the other, otherwise you'll confuse some browsers.

- Map Name

Just like any person, document, image and country, image maps also require names. This is simply because if you are going to include more than one map on your page, then to identify which image goes with which map, you'll need to name the map.

```
<map name="apple"> … </map>

<map name="mango"> … </map>
```

- Area & Anchor Tags

<area *shape*="rect"> is a tag that identifies which hot spot is being used. The other is the (*href*) anchor tag, <… href="http://www.YourLink.com>. So in a completed code, they should both look like this:

```
    <area shape="circle" coords="245,30,10" href="http://www.YourLink.com">
```

- Final Code

```
<body>
<img src="trees.gif" usemap="#green" border="0">
  <map name="green">
```

```
        <area shape="polygon" coords="19,44,45,11,87,37,82,76,49,98" href=" save.html">
        <area shape="rect" coords="128,132,241,179" href=" furniture.html">
        <area shape="circle" coords="68,211,35" href=" plantations.html">
    </map>
</body>
```

Example

```
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap" />

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
  <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
  <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
```

Entities References used in HTML

In HTML we cannot directly use the special symbols so we use a technique called Entities References. With this we can keep any symbols in a web page. It takes a form: **&Entity_Name;**

Some of the mostly used symbols and their corresponding entities are as follows:

Result	Description	Entity Name
	non-breaking space	
<	less than	<
>	greater than	>
&	ampersand	&
"	quotation mark	"
'	apostrophe	' (does not work in IE)
¢	cent	¢
£	pound	£
¥	yen	¥
€	euro	€
§	section	§
©	copyright	©
®	registered trademark	®
×	multiplication	×
÷	division	÷

And more…

Note: This handout is for simple reference only. Do not completely depend on it.

Points for review:

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage.
- HTML file consist of tags also called markups to display information in an arranged way.
- Tags in HTML are predefined i.e. we cannot create our own tags.
- Web browser is application software that is used to view web pages created using HTML.
- HTML document is mainly divided into two parts: head and body.
- The current version of HTML that we are using now is HTML4.
- W3C stands for World Wide Web consortium is a body which looks after the standardization of HTML language.
- HTML file ends with .htm or .html extension.
- Every starting tag has its corresponding ending tag in HTML for e.g. …. .
- HTML also has empty tags like
 which is used to break row.
- The browser ignores comments. <!-- … … … … -->
- Tag can also have attributes, which are used to define the properties of a tag.

  For e.g. …
- Character entities are used to display some special characters, which cannot be typed from the keyboard. It is also used to display some of the characters, which are forbidden to be written as element content.

HTML Assistants

HTML Assistant is a software tool, which helps you to build web pages with HTML without a lot of extraneous features. HTML assistants are favorite among Web developers which makes HTML page design easy.

E.g.

BROOKLYN NORTH HTML Assistant Pro
Adobe Dreamweaver
Microsoft FrontPage

HTML Assistant Basic Features

- Text editor
- Color coding
- FTP or Site manager

- Search and replace / Extended search and replace (across multiple documents)
- Can edit JavaScript
- Additional CSS assistance
- Spell checking
- Form wizard
- Includes a built-in RTF to HTML converter.
    - Etc.

HTML Editors

A **HTML editor** is an authoring software program that is used to create content for web sites. HTML software is easy to use since it has a feature that is known as **WYSIWYG**.

It is a software application for creating web pages. Although the HTML markup of a web page can be written with any text editor, specialized HTML editors can offer convenience and added functionality. For example, many HTML editors work not only with HTML, but also with related technologies such as CSS, XML and JavaScript.

When you design web pages you want to use editor features that are simple to understand. You can buy and download HTML management software from the Internet as well as templates that will help you create web pages for your business or personal use.
HTML editors are also great for creating tables; building borders around images and changing background color in no time at all. You can easily change the design of your website in just a few minutes to reflect your business style.

Examples:

- Vim Editor, gEdit Editor, Emacs Editor, BlueFish Editor (Linux)
- Notepad, Wordpad, Notepad+ (Windows)
- BlueFish Editor, TextEdit, TextWrangler (Mac)

WYSIWYG - ***W****hat* ***Y****ou* ***S****ee* ***I****s* ***W****hat* ***Y****ou* ***G****et*. The term is used in computing to describe a system in which content (text and graphics) displayed onscreen during editing appears in a form exactly

corresponding to its appearance when printed or displayed as a finished product, which might be a printed document, web page, or slide presentation.

WYSIWYG implies a user interface that allows the user to view something very similar to the end result while the document is being created. In general WYSIWYG implies the ability to directly manipulate the layout of a document without having to type or remember names of layout commands. The actual meaning depends on the user's perspective.

Examples:

- Adobe Dreamweaver (With this software user can view the output as he writes the HTML code.)
- Microsoft Front Page

HTML Convertor

A software program that is used to convert basic text files to HTML code. Different software tools can be used for the conversion of HTML.

Developer use drag-and-drop and other graphical technique to create the elements like table, form etc. and corresponding HTML code will be automatically writer by the convertor. With this tool the developer do not need to know the detailed code.

In Adobe Dreamweaver, the user can make required design with Drag-and-Drop mouse event and the software will automatically write the corresponding HTML code. The Adobe Photoshop also provide this facility, The user can save the sliced image in web format which will automatically convert the image to HTML files that can be opened with the browsers. Microsoft FrontPage can also be used for the same purpose.

The office software packages also provide the facility to save the documents in web format, which means they are converting the normal documents in web format.

CSS

- CSS is an acronym for **C**ascading **S**tyle **S**heets.
- A CSS (Cascading Style Sheet) file allows to separate your web sites HTML content from it's style. HTML file is used to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on…) is accomplished with a CSS.
- Styles define **how to display** HTML elements
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in *.**CSS files**

What can be done with CSS?

CSS is a style language that defines layout of HTML documents. For example, CSS covers fonts, colors, margins, lines, height, width, background images, advanced positions and many other things.

HTML can be (mis-)used to add layout to websites. But CSS offers more options and is more accurate and sophisticated. All browsers support CSS.

What is the difference between CSS and HTML?

HTML is used to structure content. CSS is used for formatting structured content.

Back in the good old days when a guy called Tim Berners Lee invented the World Wide Web, the language HTML was only used to add structure to text. An author could mark his text by stating, "this is a headline" or "this is a paragraph" using HTML tags such as <h1> and <p>.

As the Web gained popularity, designers started looking for possibilities to add layout to online documents. To meet this demand, the browser producers (at that time Netscape and Microsoft) invented new HTML tags such as for example <font> which differed from the original HTML tags by defining layout - and not structure. This also led to a situation where original structure tags such as <table> were increasingly being misused to layout pages instead of adding structure to text. Many new layout tags such as <blink> were only supported by one type of browser. "You need browser X to view this page." became a common disclaimer on web sites.

CSS was invented to remedy this situation by providing web designers with sophisticated layout opportunities supported by all browsers. At the same time, separation of the presentation style of documents from the content of documents makes site maintenance a lot easier.

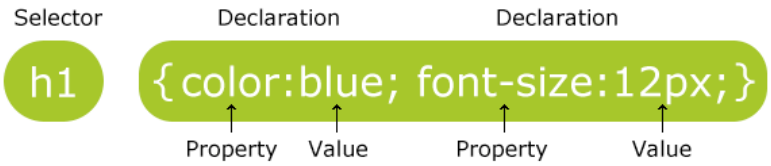
Which benefits will CSS give?

CSS was a revolution in the world of web design. The concrete benefits of CSS include:

- control layout of many documents from one single style sheet;

- more precise control of layout;
- apply different layout to different media-types (screen, print, etc.);
- numerous advanced and sophisticated techniques.

The basic CSS syntax



Let's say we want a nice red color as the background of a webpage:

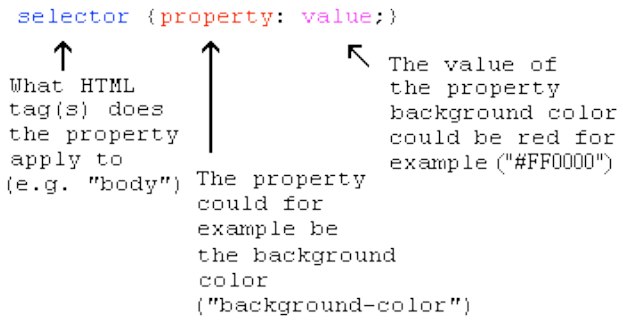
Using **HTML** we could have done it like this:

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

With **CSS** the same result can be achieved like this:

```
body {background-color: #FF0000;}
```

As you will note, the codes are more or less identical for HTML and CSS. The above example also shows you the fundamental CSS model:



CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers. A CSS comment begins with "/*", and ends with "*/", like this:

```
/* This is a comment */
p
{
text-align:center;

/* This is another comment */
}
```

Applying CSS to an HTML document

There are three ways you can apply CSS to an HTML document. These methods are all outlined below. We recommend that you focus on the third method i.e. external.

Method 1: In-line (the attribute style)

One way to apply CSS to HTML is by using the HTML attribute style. Building on the above example with the red background color, it can be applied like this:

```
<html>
<head> <title>Example</title></head>
<body style="background-color: #FF0000;">
        <p>This is a red page</p>
</body>
</html>
```

Method 2: Internal (the tag style)

Another way is to include the CSS codes using the HTML tag <style>. For example like this:

```
<html>
<head>
        <title>Example</title>
        <style type="text/css">
                body {background-color: #FF0000;}
        </style>
</head>
<body>
        <p>This is a red page</p>
</body>
</html>
```

Method 3: External (link to a style sheet)

The recommended method is to link to a so-called external style sheet. An external style sheet is simply a text file with the extension **.css**. Like any other file, you can place the style sheet on your web server or hard disk.

For example, let's say that your style sheet is named **style.css** and is located in a folder named **style**. The situation can be illustrated like this:

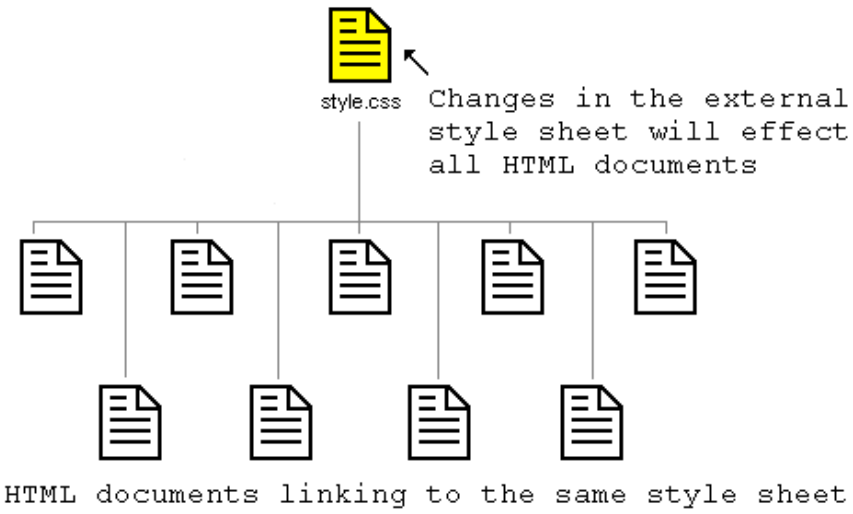
The trick is to create a link from the HTML document (default.htm) to the style sheet (style.css). Such link can be created with one line of HTML code:

```
<link rel="stylesheet" type="text/css" href="style/style.css" />
```

Notice how the path to our style sheet is indicated using the attribute 'href'. The line of code must be inserted in the header section of the HTML code i.e. between the <head> and </head> tags. Like this:

```
<html>
<head>
<title>My document</title>
        <link rel="stylesheet" type="text/css" href="style/style.css" />
</head>
<body>
        … … … ...
```

This link tells the browser that it should use the layout from the CSS file when displaying the HTML file. The really smart thing is that several HTML documents can be linked to the same style sheet. In other words, one CSS file can be used to control the layout of many HTML documents.



This technique can save you a lot of work. If you, for example, would like to change the background color of a website with 100 pages, a style sheet can save you from having to manually change all 100 HTML documents. Using CSS, the change can be made in a few seconds just by changing one code in the central style sheet.

CSS Classes

The class selector allows to style items within the same HTML element differently. With classes the style can be overwritten by changing out stylesheets. Same class selector can be used again and again within an HTML file.

This style will be applied to all tags.

```
p {
  font-size: small;
  color: #333333
}
```

But lets say that we wanted to change some words to green bold text within the paragraph, while leaving the rest of the sentence untouched. We would do the following:

```
.greenBoldText
{
  font-size: small;
  color: #008080;
  font-weight: bold;
}
```

 This is main paragraph, it contains Green Bold Text and it still continues with main formatting.

CSS IDs

IDs are similar to classes, except once a specific id has been declared it cannot be used again within the same HTML file.

Use IDs to style the layout elements of a page that will only be needed once and use classes to style text and such that may be declared multiple times.

The main container for this page is defined by the following.

```
<div id="container">
    Everything within my document is inside this division.
</div>
```

Here, id selector is chosen for the "container" division over a class, because it will be used one time only within the file.

CSS file looks following:

```
#container
{
  width: 80%;
  margin: auto;
  padding: 20px;
  border: 1px solid #666;
  background: #ffffff;
}
```

Note: The id selector begins with a (#) sign instead of a (.)

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

For example, an external style sheet has these properties for the h3 selector:

```
h3
{
color:red;
text-align:left;
font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
text-align:right;
font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Mostly Used CSS$^{2.0}$ Properties

Background Properties

Property	Description	Possible Values	Examples
background-attachment	Declares the attachment of a background image (to scroll with the page content or be in a fixed position).	*fixed* *scroll*	*div { background-attachment:fixed; }* *div { background-attachment:scroll; }*
background-color	Declares the background color.	Valid color names, RGB values, hexadecimal notation.	*div { background-color:green; }* *div { color:#00FF00; }*
background-image	Declares the background image of an element.	URL values.	*div { background-image:url(images/img.jpg); }* *body { background-image:url(img.jpg); }*
background-position	Declares the position of a background image.	*top left* *top center* *top right* *center left* *center center* *center right* *bottom left* *bottom center* *bottom right*	*div { background-position:10px 50px; }* *div { background-position:bottom right; }*
background-repeat	Declares how and/or if a background image repeats.	*repeat* *repeat-x* *repeat-y* *no-repeat*	*div { background-repeat:repeat-x; }* *div { background-repeat:no-repeat; }*
background	Used as a shorthand property to set all the background properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *background-color* *background-image* *background-repeat* *background-attachment* *background-position*	*div { background:green url(image.jpg) no-repeat fixed center center; }* *div { background:url(image.jpg) fixed; }*

Border Properties

Property	Description	Possible Values	Examples
border-top-color	Declares the color of the top border.	Valid color names, RGB values, hexidecimal notation, or the predefined value **transparent**.	*div { border-top-color:green; }* *div { border-top-color:#00FF00; }*
border-top-style	Declares the style of the top border.	*none* *hidden* *dotted* *dashed* *solid* *double* *groove* *ridge* *inset* *outset*	*div { border-top-style:solid; }* *div { border-top-style:inset; }*
border-top-width	Declares the width of the top border.	Lengths or the following predefined values: *thin* *medium* *thick*	*div { border-top-width:2px; }* *div { border-top-width:thin; }*
border-top	Used as a shorthand property to set all the border-top properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *border-top-width* *border-top-style* *border-top-color*	*div { border-top:2px solid green; }* *div { border-top:thick double #00FF00; }*
border-right-color	Declares the color of the right border.	Valid color names, RGB values, hexadecimal notation, or the predefined value **transparent**.	*div { border-right-color:green; }* *div { border-right-color:#00FF00; }*
border-right-style	Declares the style of the right border.	*none* *hidden* *dotted* *dashed* *solid* *double* *groove* *ridge* *inset* *outset*	*div { border-right-style:solid; }* *div { border-right-style:inset; }*
border-right-width	Declares the width of the right border.	**Lengths or the following predefined values:** *thin* *medium* *thick*	*div { border-right-width:2px; }* *div { border-right-width:thin; }*

border-right	Used as a shorthand property to set all the border-right properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *border-right-width* *border-right-style* *border-right-color*	*div { border-right:2px solid green; }* *div { border-right:thick double #00FF00; }*
border-bottom-color	Declares the color of the bottom border.	Valid color names, RGB values, hexidecimal notation, or the predefined value **transparent**.	*div { border-bottom-color:green; }* *div { border-bottom-color:#00FF00; }*
border-bottom-style	Declares the style of the bottom border.	*none* *hidden* *dotted* *dashed* *solid* *double* *groove* *ridge* *inset* *outset*	*div { border-bottom-style:solid; }* *div { border-bottom-style:inset; }*
border-bottom-width	Declares the width of the bottom border.	Lengths or the following predefined values: *thin* *medium* *thick*	*div { border-bottom-width:2px; }* *div { border-bottom-width:thin; }*
border-bottom	Used as a shorthand property to set all the border-bottom properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *border-bottom-width* *border-bottom-style* *border-bottom-color*	*div { border-bottom:2px solid green; }* *div { border-bottom:thick double #00FF00; }*
border-left-color	Declares the color of the left border.	Valid color names, RGB values, hexidecimal notation, or the predefined value **transparent**.	*div { border-left-color:green; }* *div { border-left-color:#00FF00; }*
border-left-style	Declares the style of the left border.	*none* *hidden* *dotted* *dashed* *solid* *double* *groove*	*div { border-left-style:solid; }* *div { border-left-style:inset; }*

		ridge *inset* *outset*	
border-left-width	Declares the width of the left border.	Lengths or the following predefined values: *thin* *medium* *thick*	*div { border-left-width:2px; }* *div { border-left-width:thin; }*
border-left	Used as a shorthand property to set all the border-left properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *border-left-width* *border-left-style* *border-left-color*	*div { border-left:2px solid green; }* *div { border-left:thick double #00FF00; }*
border-color	Declares the border color of all four borders at once.	Valid color names, RGB values, hexidecimal notation, or the predefined value **transparent**. Separate the color for each border by a space, declaring the colors for the borders in the following order: border-top-color border-right-color border-bottom-color border-left-color Undeclared values work as further shorthand notation. If only one color value is declared, all four borders will use that color. If two colors are declared, the top and bottom borders will use the first color while the right and left borders will use the second color. If three colors are declared, the top border will use the first color, the right and left borders will use the second color, and the bottom border will use the third color.	*div { border-color:green red blue olive; }* *div { border-color:green; }* *div { border-color:green red; }* *div { border-color:green red blue; }*
border-style	Declares the border style of all four borders at once.	*none* *hidden* *dotted* *dashed* *solid* *double* *groove* *ridge* *inset* *outset*	*div { border-style:solid dotted dashed double; }* *div { border-style:solid; }* *div { border-style:solid dotted; }* *div { border-style:solid dotted dashed; }*
border-width	Declares the width of all four borders at	Lengths or the following predefined values:	*div { border-width:1px 3px*

	once.	*thin* *medium* *thick*	*5px 2px; }* *div { border-width:thin; }*
border	Used as a shorthand to declare the border properties when all four borders will have the same appearance.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *border-width* *border-style* *border-color*	*div { border:1px double green; }* *div { border:thin solid #00FF00; }*

Classification and Positioning Properties

Property	Description	Possible Values	Examples
clear	Declares the side(s) of an element where no previous floating elements are allowed to be adjacent.	*left* *right* *both* *none*	*div { clear:right; }* *div { clear:both; }*
cursor	Declares the type of cursor to be displayed.	URL values, and the following prefefined values: *auto* *crosshair* *default* *pointer* *move* *e-resize* *ne-resize* *nw-resize* *n-resize* *se-resize* *sw-resize* *s-resize* *w-resize* *text* *wait* *help*	*div { cursor:crosshair; }* *div { cusrsor:url(image.csr); }* *div { cusrsor:url(image.csr), pointer; }*
display	Declares if/how the element displays.	*none* *inline* *block* *list-item* *run-in* *compact* *marker* *table* *inline-table* *table-row-group* *table-header-group* *table-footer-group* *table-row*	*div { display:none; }* *div { display:inline; }* *div { display:marker; }*

		table-column-group *table-column* *table-cell* *table-caption*	
float	Declares whether a box should float to the left or right of other content, or whether it should not be floated at all.	*left* *right* *none*	*div { float:left; }* *div { float:right; }*
visibility	Declares the visibility of boxes generated by an element.	visible hidden collapse	*div { visibility:visible; }* *div { visibility:hidden; }*
top	Declares the distance that the top content edge of the element is offset below the top edge of its containing block. The **position** property of the element must also be set to a value other than **static**.	Lengths, percentages, and the predefined value **auto**.	*div { top:15px; }* *div { top:2%; }*
right	Declares the distance that the right content edge of the element is offset to the left of the right edge of its containing block. The **position** property of the element must also be set to a value other than **static**.	Lengths, percentages, and the predefined value **auto**.	*div { right:15px; }* *div { right:2%; }*
bottom	Declares the distance that the bottom content edge of the element is offset above the bottom edge of its containing block. The **position** property of the element must also be set to a value other than **static**.	Lengths, percentages, and the predefined value **auto**.	*div { bottom:15px; }* *div { bottom:2%; }*
left	Declares the distance that the left content edge of the element is offset to the right of the left edge of its containing block. The **position** property of the element must also be set to a value other than **static**.	Lengths, percentages, and the predefined value **auto**.	*div { left:15px; }* *div { left:2%; }*
position	Declares the type of positioning of an element.	*static* *relative* *absolute* *fixed*	*div { position:absolute; }* *div { position:relative; }*
clip	Declares the shape of a clipped region when the value of the **overflow** property is set to a value other than **visible**.	Shapes, or the predefined value **auto**. *rect(top, right, bottom, left)*	*div { clip:auto; }* *div { clip:rect(2px, 4px, 7px, 5px); }*
overflow	Declares how content that overflows the element's box is handled.	*visible* *hidden* *scroll* *auto*	*div { overflow:hidden; }* *div { overflow:scroll; }*
vertical-align	Declares the vertical alignment of an inline-level element or a table cell.	Lengths, percentages, and the following predefined values: *baseline* *sub* *super*	*span { vertical-align:middle; }* *td { vertical-align:top; }*

		top *text-top* *middle* *bottom* *text-bottom*	
z-index	Declares the stack order of the element.	Integer values and the predefined value **auto**.	*div { z-index:2; }* *div { z-index:auto; }*

Dimension Properties

Property	Description	Possible Values	Examples
height	Declares the height of the element.	Lengths, percentages, and the predefined value **auto**.	*div { height:200px; }* *div { height:50%; }*
max-height	Declares the maximum height of the element.	Lengths, percentages, and the predefined value **auto**.	*div { max-height:200px; }* *div { max-height:50%; }*
min-height	Declares the minimum height of the element.	Lengths, percentages, and the predefined value **auto**.	*div { min-height:200px; }* *div { min-height:50%; }*
width	Declares the width of the element.	Lengths, percentages, and the predefined value **auto**.	*div { width:500px; }* *div { width:75%; }*
max-width	Declares the maximum width of the element.	Lengths, percentages, and the predefined value **auto**.	*div { max-width:500px; }* *div { max-width:75%; }*
min-width	Declares the minimum width of the element.	Lengths, percentages, and the predefined value **auto**.	*div { min-width:500px; }* *div { min-width:75%; }*

Font Properties

Property	Description	Possible Values	Examples
font-family	Declares the name of the font to be used. Previously set in HTML via the *face* attribute in a <font> tag.	Valid font family names or generic family names, i.e. *Arial*, *Verdana*, *sans-serif*,*"Times New Roman"*, *Times*, *serif*, etc. Font family names can be separated by a comma in the same declaration to allow additional and/or generic family names to be used if the preferred font is unable to be displayed.	*div { font-family:Arial; }* *div { font-family:Arial, Helvetica, sans-serif; }*

font-size	Declares the size of the font. Previously set in HTML via the *size* attribute in a <font> tag.	Lengths (number and unit type— i.e. *1em*,*12pt*, *10px*, *80%*) or one of the following predefined values: *xx-small* *x-small* *small* *medium* *large* *x-large* *xx-large* *smaller* *larger*	*div { font-size:70%; }* *div { font-size:0.85em; }* *div { font-size:medium; }*
font-style	Declares the font style.	*normal* *italic* *oblique*	*div { font-style:italic; }* *div { font-style:oblique; }*
font-variant	Declares the font variant.	*normal* *small-caps*	*div { font-variant:normal; }* *div { font-variant:small-caps; }*
font-weight	Declares the font weight (lightness or boldness)	*normal* *bold* *bolder* *lighter* *100* *200* *300* *400* *500* *600* *700* *800* *900*	*div { font-weight:bolder; }* *div { font-weight:200; }*
font	Used as a shorthand property to declare all of the font properties at once (except font-size-adjust and font-stretch).	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *font-style* *font-variant* *font-weight* *font-size* *line-height* *font-family*	*div { font:italic small-caps bold 1em 1.2em Arial }* *div { font:bold 0.8em Verdana }*

List Properties

Property	Description	Possible Values	Examples
list-style-type	Declares the type of list marker used.	*disc* *circle* *square* *decimal* *decimal-leading-zero* *lower-roman* *upper-roman* *lower-alpha* *upper-alpha* *lower-greek* *lower-latin* *upper-latin*	*ol { list-style-type:upper-roman; }* *ul { list-style-type:square; }*
list-style-position	Declares the position of the list marker.	*inside* *outside*	*ol { list-style-position:inside; }* *ul { list-style-position:outside; }*
list-style-image	Declares an image to be used as the list marker.	URL values.	*ul { list-style-image:url(image.jpg); }*
list-style	Shorthand property to declare three list properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *list-style-type* *list-style-position* *list-style-image*	*ul { list-style:disc inside url(image.gif); }* *ol { list-style:upper-roman outside; }*
marker-offset	Declares the marker offset for elements with a value of**marker** set for the **display**property.	Lengths and the predefined value**auto**.	*li:before { display:marker; marker-offset:5px; }*

Margin Properties

Property	Description	Possible Values	Examples
margin-top	Declares the top margin for the element.	Lengths, percentages, and the predefined value **auto**.	*div { margin-top:5px; }* *div { margin-top:15%; }*
margin-right	Declares the right margin for the element.	Lengths, percentages, and the predefined value **auto**.	*div { margin-right:5px; }* *div { margin-right:15%; }*
margin-bottom	Declares the bottom margin for the element.	Lengths, percentages, and the predefined value **auto**.	*div { margin-bottom:5px; }* *div { margin-bottom:15%; }*
margin-left	Declares the left margin for the	Lengths, percentages, and the predefined value **auto**.	*div { margin-left:5px; }*

	element.		*div { margin-left:15%; }*
margin	Shorthand property used to declare all the margin properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *margin-top* *margin-right* *margin-bottom* *margin-left*	*div { margin:5px 12px 4px 7px; }* *div { margin:5px; }* *div { margin:5px 10px; }* *div { margin:5px 7px 4px; }*

Padding Properties

Property	Description	Possible Values	Examples
padding-top	Declares the top padding for the element.	Lengths, percentages, and the predefined value **auto**.	*div { padding-top:5px; }* *div { padding-top:15%; }*
padding-right	Declares the right padding for the element.	Lengths, percentages, and the predefined value **auto**.	*div { padding-right:5px; }* *div { padding-right:15%; }*
padding-bottom	Declares the bottom padding for the element.	Lengths, percentages, and the predefined value **auto**.	*div { padding-bottom:5px; }* *div { padding-bottom:15%; }*
padding-left	Declares the left padding for the element.	Lengths, percentages, and the predefined value **auto**.	*div { padding-left:5px; }* *div { padding-left:15%; }*
padding	Shorthand property used to declare all the margin properties at once.	Separate values by a space in the following order (those that are not defined will use inherited or default initial values): *padding-top* *padding-right* *padding-bottom* *padding-left*	*div { padding:5px 12px 4px 7px; }* *div { padding:5px; }* *div { padding:5px 10px; }* *div { padding:5px 7px 4px; }*

Table Properties

Property	Description	Possible Values	Examples
border-collapse	Declares the way borders are displayed.	*collapse* *separate*	*table { border-collapse:collapse; }* *table { border-collapse:separate; }*
border-spacing	Declares the distance separating borders (if **border-collapse** is**separate**).	Lengths for the horizontal and vertical spacing, separated by a space. If one length is value is declared, that length is used for both the horizontal and vertical spacing. If two lengths are declared, the first one is used for horizontal spacing and the second one is used for vertical spacing.	*table { border-spacing:5px; }* *table { border-spacing:5px 10px; }*
caption-side	Declares where the table caption is displayed in relation to the table.	*top* *bottom* *left* *right*	*caption { caption-side:top; }* *caption { caption-side:right; }*
empty-cells	Declares the way empty cells are displayed (if**border-collapse**is **separate**).	*show* *hide*	*table { empty-cells:show; }* *table { empty-cells:hide; }*
table-layout	Declares the type of table layout.	*auto* *fixed*	*table { table-layout:auto; }* *table { table-layout:fixed; }*

Text Properties

Property	Description	Possible Values	Examples
color	Declares the color of the text.	Valid color names, RGB values, hexidecimal notation. *aqua* *black* *blue* *fuchsia* *gray* *green* *lime* *maroon* *navy* *olive* *purple* *red*	*div { color:green; }* *div {color:rgb(0,255,0); }* *div { color:#00FF00; }*

		silver *teal* *white* *yellow*	
direction	Declares the reading direction of the text.	ltr = left-to-right rtl = right-to-left	*div { direction:ltr; }* *div { direction:rtl; }*
line-height	Declares the distance between lines.	Numbers, percentages, lengths, and the predefined value of *normal*.	*div { line-height:normal; }* *div { line-height:2em; }* *div { line-height:125%; }*
letter-spacing	Declares the amount of space between text characters.	A length (in addition to the default space) or the predefined value of*normal*.	*div { letter-spacing:normal; }* *div { letter-spacing:5px; }* *div { letter-spacing:-1px; }*
text-align	Declares the horizontal alignment of inline content.	*left* *right* *center* *justify*	*div { text-align:center; }* *div { text-align:right; }*
text-decoration	Declares the text decoration.	*none* *underline* *overline* *line-through* *blink*	*div { text-decoration:none; }* *div { text-decoration:underline; }*
text-indent	Declares the indentation of the first line of text.	Lengths and percentages.	*div { text-indent:12px; }* *div { text-indent:2%; }*
text-shadow	Declares shadow effects on the text.	A list containg a color followed by numeric values (separated by spaces) that specify: 1. The color for the shadow effect 2. Horizontal distance to the right of the text 3. Vertical distance below the text 4. Blur radius	*div { text-shadow:green 2px 2px 7px; }* *div { text-shadow:olive -3px -4px 5px; }*
text-transform	Declares the capitalization effects on the letters in the text.	*none* *capitalize* *uppercase* *lowercase*	*div { text-transform:uppercase; }* *div { text-transform:lowercase; }*
word-spacing	Declares the space between words in the text.	A length (in addition to the default space) or the predefined value of*normal*.	*div { word-spacing:normal; }* *div { word-spacing:1.5em; }*

DHTML

Dynamic HTML is not really a new specification of HTML, but rather a new way of looking at and controlling the standard HTML codes and commands. When thinking of dynamic HTML, you need to remember the qualities of standard HTML, especially that once a page is loaded from the server, it will not change until another request comes to the server. Dynamic HTML gives you more control over the HTML elements and allows them to change at any time, without returning to the Web server.

There are four parts to DHTML:

- Document Object Model (DOM)
- Scripts
- Cascading Style Sheets (CSS)
- HTML

DOM

The DOM is what allows you to access any part of your Web page to change it with DHTML. The DOM specifies every part of a Web page and using its consistent naming conventions you can access them and change their properties.

Scripts

Scripts written in either JavaScript or VBScript.

Cascading Style Sheets (CSS)

CSS is used in DHTML to control the look and feel of the Web page. Style sheets define the colors and fonts of text, the background colors and images, and the placement of objects on the page. Using scripting and the DOM, you can change the style of various elements.

HTML

HTML 4.x is used to create the page itself and build the elements for the CSS and the DOM to work on. There is nothing special about HTML for DHTML - but having valid HTML is even more important.

These are most common features of DHTML:

- **Changing the tags and properties**
    - This is one of the most common uses of DHTML. It allows changing the qualities of an HTML tag depending on an event outside of the browser (such as a mouse click, time, or date, and so on). With this information can be preloaded but not displayed until the reader clicks on a specific link.
- **Real-time positioning**
    - When most people think of DHTML this is what they expect. Objects, images, and text moving around the Web page. This can allow you to play interactive games with your readers or animate portions of your screen.

Practical Exercises:

Type the following codes in a notepad and save it as a html page to see the output. Here in this example we have used tables to divide the content of web page into different sections.

Question 1:

```
<html>
<head>
<title>WEB PAGE TITLE GOES HERE</title>
</head>

<body style="margin: 0px; padding: 0px; font-family: 'Trebuchet MS',verdana;">
<table width="100%" style="height: 100%;" cellpadding="10" cellspacing="0"
border="0"><tr>
<!-- ============ LEFT COLUMN (MENU) ============== -->
<td width="20%" valign="top" bgcolor="#999f8e">
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a>
</td>
<!-- ============ RIGHT COLUMN (CONTENT) ============== -->
<td width="80%" valign="top" bgcolor="#d2d8c7">

<h1>Website Logo</h1>
<h2>Page heading</h2>
This is a basic two-column web page layout. The left column or the <i>menu
column</i> is a narrow band of space (usually between 15-25% of the page width)
and is reserved for a menu of hyperlinks leading to other pages on your
website. The table used to create this layout employs a single table row
containing two table cells.<br>
<br>
The right column or the <i>content column</i> takes up the lion's share of the
web page width and contains the actual content of each particular page. In a
basic two column layout like this, it is common to place the website logo at
the top of the content column on each page.</td></tr></table>
</body>
</html>
```

Question 2:

```
<html>
<head>
<title>WEB PAGE TITLE GOES HERE</title>
</head>
<body style="margin: 0px; padding: 0px; font-family: 'Trebuchet MS',verdana;">
```

```
<table width="100%" style="height: 100%;" cellpadding="10" cellspacing="0"
border="0">
<tr>
<!-- ============ HEADER SECTION ============== -->
<td colspan="2" style="height: 100px;" bgcolor="#777d6a"><h1>Website
Logo</h1></td></tr>

<tr>
<!-- ============ LEFT COLUMN (MENU) ============== -->
<td width="20%" valign="top" bgcolor="#999f8e">
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a>
</td>

<!-- ============ RIGHT COLUMN (CONTENT) ============== -->
<td width="80%" valign="top" bgcolor="#d2d8c7">
<h2>Page heading</h2>

Here's a two column layout with a header section that spans the width of both
columns. The first table row creates the header and contains a single table
cell which uses the colspan="2" attribute-value pair. The website logo
typically goes in the header section.<br>
<br>
The second table row contains two table cells which create the menu column
(left) and the content column (right). The colspan attribute is not set in
either so they default to colspan="1".</td></tr></table>
</body>
</html>
```

Question 3:

```
<html>
<head>
<title>WEB PAGE TITLE GOES HERE</title>
</head>
<body style="margin: 0px; padding: 0px; font-family: 'Trebuchet MS',verdana;">
<table width="100%" style="height: 100%;" cellpadding="10" cellspacing="0"
border="0">
<tr>
<!-- ============ HEADER SECTION ============== -->
<td colspan="2" style="height: 100px;" bgcolor="#777d6a"><h1>Website
Logo</h1></td></tr>

<tr>
<!-- ============ LEFT COLUMN (MENU) ============== -->
```

```
<td width="20%" valign="top" bgcolor="#999f8e">
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a><br>
<a href="#">Menu link</a>
</td>

<!-- ============ RIGHT COLUMN (CONTENT) ============== -->
<td width="80%" valign="top" bgcolor="#d2d8c7">
<h2>Page heading</h2>

Here's a two column layout with header and footer sections that span the width
of both columns. The first table row creates the header and contains a single
table cell which uses the colspan="2" attribute-value pair.<br>
<br>
The second table row contains two table cells which create the menu column
(left) and the content column (right). The colspan attribute is not set in
either so they default to colspan="1".<br>
<br>
The third table row creates the footer. Like the header, it contains a single
table cell which uses the colspan="2" attribute-value pair.</td></tr>

<!-- ============ FOOTER SECTION ============== -->
<tr><td colspan="2" align="center" height="20" bgcolor="#777d6a">Copyright
©</td></tr>
</table>
</body>
</html>
```

Question 4

```
<!-- save the file as frame.html-->
<frameset rows="75%, *" cols="*, 40%">
  <frame src="framea.html">
  <frame src="frameb.html">
  <frame src="framec.html">
  <frame src="framed.html">

  <noframes>
       <h1>Your browser does not supports frames</h1>
       Click the below link to continue your visit.
       <a href="noframes.html">no-frames</a>
  </noframes>

</frameset>
</html>
```

```
<!-- save the file as framea.html-->
<html>
<head>
<title> framea</title>
</head>
<body>
<h1>this is framea</h1>
</body>
</html>

<!-- save the file as frameb.html-->
<html>
<head>
<title> frameb</title>
</head>
<body>
<h1>this is frameb</h1>
</body>
</html>

<!-- save the file as framec.html-->
<html>
<head>
<title> framec</title>
</head>
<body>
<h1>this is framec</h1>
</body>
</html>

<!-- save the file as framed.html-->
<html>
<head>
<title> framed</title>
</head>
<body>
<h1>this is framed</h1>
</body>
</html>
```

~