

CHAPTER 1 : INTRODUCTION TO INTERNET

COMPUTER NETWORK

A computer network is a group of computer systems and other computing hardware devices that are linked together through communication channels to facilitate communication and resource-sharing among a wide range of users. Networks are commonly categorized based on their characteristics.

TYPE OF COMPUTER NETWORK

LAN(Local Area Network) - A LAN connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs (perhaps one per room), and occasionally a LAN will span a group of nearby buildings. In TCP/IP networking, a LAN is often but not always implemented as a single IP subnet.

In addition to operating in a limited space, LANs are also typically owned, controlled, and managed by a single person or organization. They also tend to use certain connectivity technologies, primarily Ethernet and Token Ring.

MAN(Metropolitan Area Network) - A network spanning a physical area larger than a LAN but smaller than a WAN, such as a city. A MAN is typically owned by an operated by a single entity such as a government body or large corporation.

WAN(Wide Area Network) - As the term implies, a WAN spans a large physical distance. The Internet is the largest WAN, spanning the Earth.

A WAN is a geographically-dispersed collection of LANs. A network device called a router connects LANs to a WAN. In IP networking, the router maintains both a LAN address and a WAN address.

NETWORK TOPOLOGY

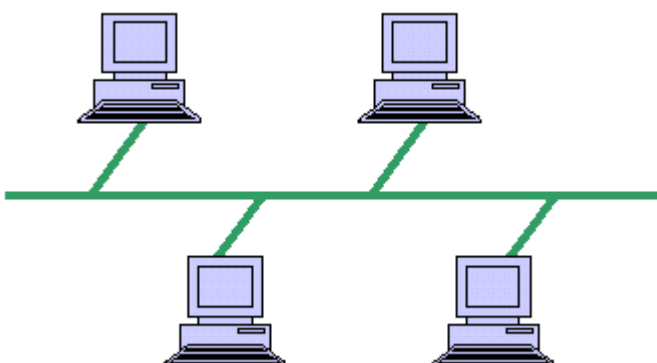
In computer networking, topology refers to the layout of connected devices.

Network topologies are categorized into the following basic types:

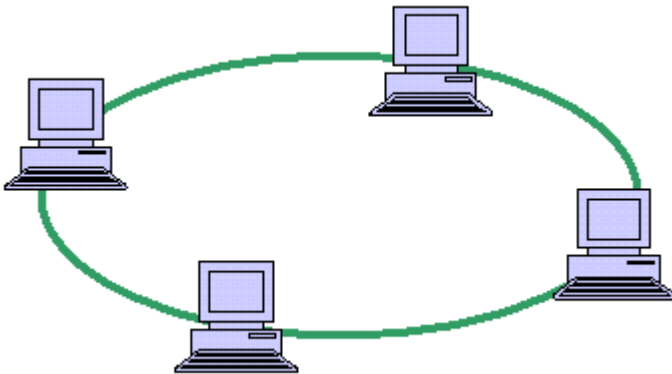
Bus, Ring, Star, Tree, Mesh

More complex networks can be built as hybrids of two or more of the above basic topologies.

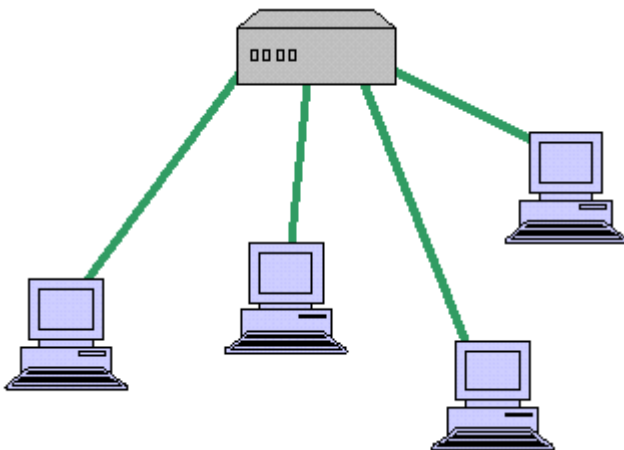
Bus Topology - Bus networks use a common backbone to connect all devices. A single cable, the backbone functions as a shared communication medium that devices attach or tap into with an interface connector. A device wanting to communicate with another device on the network sends a broadcast message onto the wire that all other devices see, but only the intended recipient actually accepts and processes the message.



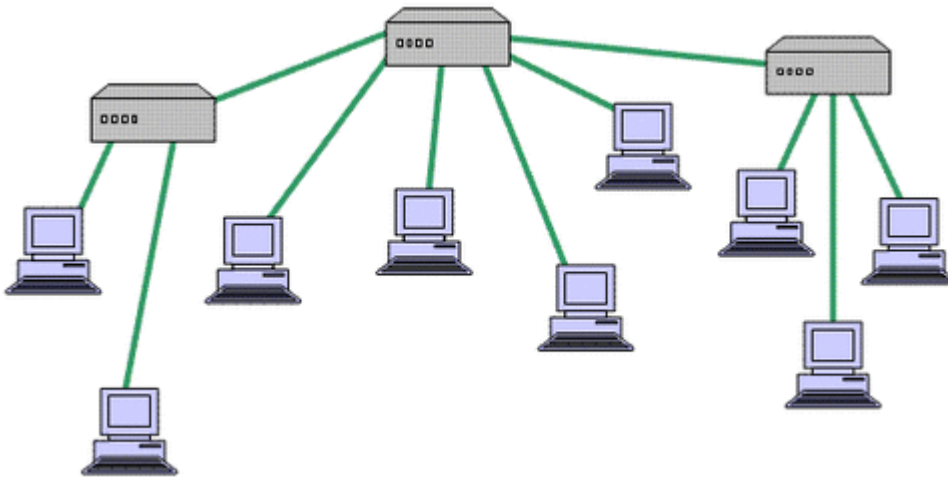
Ring Topology - In a ring network, every device has exactly two neighbors for communication purposes. All messages travel through a ring in the same direction (either "clockwise" or "counterclockwise"). A failure in any cable or device breaks the loop and can take down the entire network.



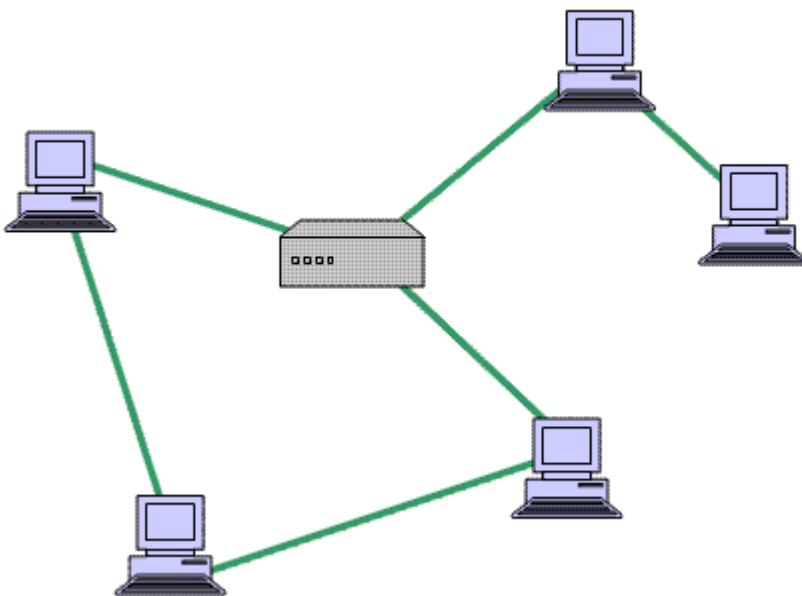
Star Topology - Many home networks use the star topology. A star network features a central connection point called a "hub node" that may be a network hub, switch or router. Devices typically connect to the hub with Unshielded Twisted Pair (UTP) Ethernet.



Tree Topology - Tree topologies integrate multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus, and each hub functions as the root of a tree of devices. This bus/star hybrid approach supports future expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub connection points) alone.



Mesh Topology - Mesh topologies involve the concept of routes. Unlike each of the previous topologies, messages sent on a mesh network can take any of several possible paths from source to destination. (Recall that even in a ring, although two cable paths exist, messages can only travel in one direction.) Some WANs, most notably the Internet, employ mesh routing.

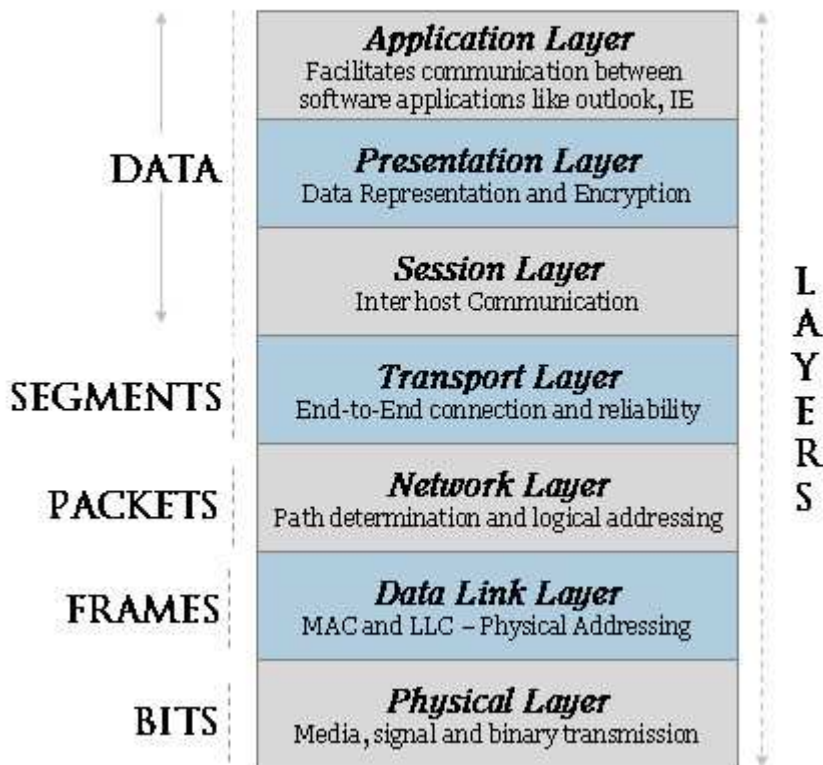


OSI REFERENCE MODEL

The OSI, or Open System Interconnection, model defines a networking framework to implement protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, and proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

There's really nothing to the OSI model. In fact, it's not even tangible. The OSI model doesn't do any functions in the networking process, It is a conceptual framework so we can better understand complex interactions that are happening. The OSI model takes the task of internetworking and divides that up into what is referred to as a vertical stack that consists of the following layers:

OSI MODEL



Open Systems Interconnection (OSI) is a standard reference model for communication between two end users in a network. The model is used in developing products and understanding networks.

Layer 1: The physical layer ...This layer conveys the bit stream through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier.

Layer 2: The data-link layer ...This layer provides synchronization for the physical level and does bit-stuffing for strings of 1's in excess of 5. It furnishes transmission protocol knowledge and management. At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sub layers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sub layer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.

Layer 3: The network layer ...This layer handles the routing of the data (sending it in the right direction to the right destination on outgoing transmissions and receiving incoming transmissions at the packet level). The network layer does routing and forwarding. It provides the functional & procedure means of transferring variable length data sequences (called datagrams) from one node to other.

Layer 4: The transport layer ...This layer manages the end-to-end control (for example, determining whether all packets have arrived) and error-checking. It ensures complete data transfer.

Layer 5: The session layer ...This layer sets up, coordinates, and terminates conversations, exchanges, and dialogs between the applications at each end. It deals with session and connection coordination.

Layer 6: The presentation layer ...This is a layer, usually part of an operating system, that converts incoming and outgoing data from one presentation format to another (for example, from a text stream into a popup window with the newly arrived text). Sometimes called the syntax layer.

Layer 7: The application layer ...This is the layer at which communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. (This layer is not the application itself, although some applications may perform application layer functions). This layer provides application services for file transfers, e-mail, and other network software services.

TCP/IP

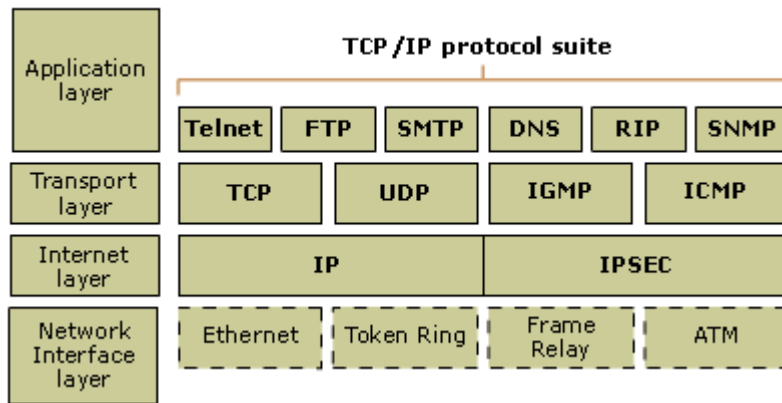
TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). When you are set up with direct access to the Internet, your computer is provided with a copy of the TCP/IP program just as every other computer that you may send messages to or get information from also has a copy of TCP/IP.

TCP/IP is a two-layer program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.

TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one (unlike ordinary phone conversations that require a dedicated connection for the call duration). Being stateless frees network paths so that everyone can use them continuously. (Note that the TCP layer itself is not stateless as far as any one message is concerned. Its connection remains in place until all packets in a message have been received.)

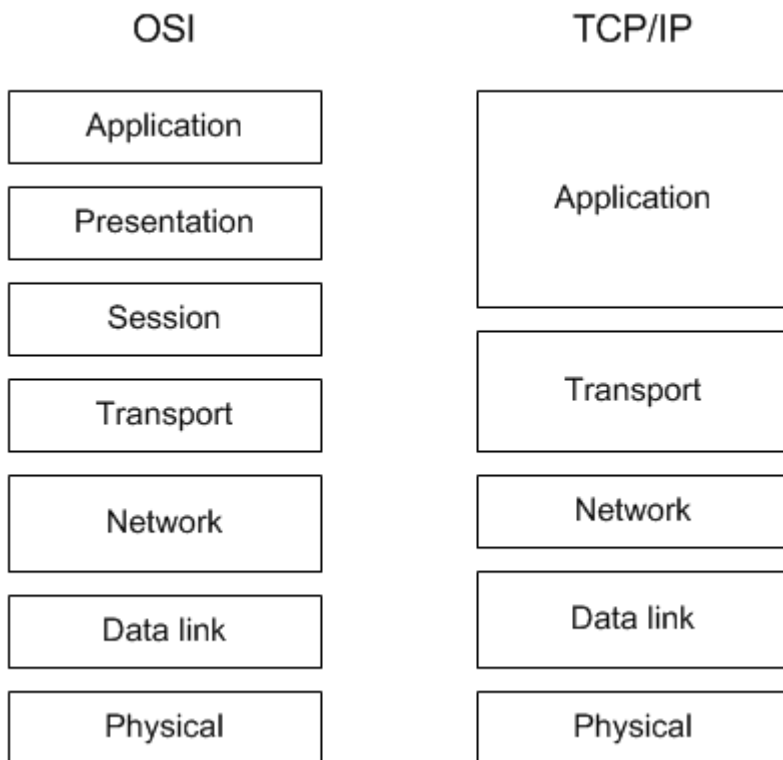
***(Stateless :** In computing, a stateless protocol is a communications protocol that treats each request as an independent transaction that is unrelated to any previous request so that the communication consists of independent pairs of request and response. A stateless protocol does not require the server to retain session information or status about each communications partner for the duration of multiple requests.)*

TCP/IP model



Layer	Description	Protocols
Application	Defines TCP/IP application protocols and how host programs interface with transport layer services to use the network.	HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP, X Windows, other application protocols
Transport	Provides communication session management between host computers. Defines the level of service and status of the connection used when transporting data.	TCP, UDP, RTP
Internet	Packages data into IP datagrams, which contain source and destination address information that is used to forward the datagrams between hosts and across networks. Performs routing of IP datagrams.	IP, ICMP, ARP, RARP
Network interface	Specifies details of how data is physically sent through the network, including how bits are electrically signaled by hardware devices that interface directly with a network medium, such as coaxial cable, optical fiber, or twisted-pair copper wire.	Ethernet, Token Ring, FDDI, X.25, Frame Relay, RS-232, v.35

TCP V/s. OSI



OSI

- 1) It has 7 layers
- 2) Transport layer guarantees delivery of packets
- 3) Separate presentation layer
- 4) Separate session layer
- 5) It defines the services, interfaces and protocols very clearly and makes a clear distinction between them
- 6) The protocols are better hidden and can be easily replaced as the technology changes
- 7) OSI truly is a general model

TCP/IP

- 1) Has 4 layers
- 2) Transport layer does not guarantee delivery of packets
- 3) No presentation layer, characteristics are provided by application layer
- 4) No session layer, characteristics are provided by transport layer
- 5) It does not clearly distinguish between service interface and protocols
- 6) It is not easy to replace the protocols
- 7) TCP/IP cannot be used for any other application

INTERNET TERMINOLOGY

The Internet is a global system of interconnected computer networks that use the standard Internet protocol suite (TCP/IP) to link several billion devices worldwide. It is a network of networks that consists of millions of private, public, academic, business, and government networks, of local to global scope, that are linked by a broad array of electronic, wireless, and optical networking technologies.

History of Internet

Until the early 1980s, what is now called the Internet was a relatively small network called ARPAnet (The Advanced Research Projects Agency Network). This small network was mainly used as a research tool for about 15 years. After the Internet was created many universities and government organizations got connected to it to exchange and distribute information. Although at first the Internet was used exclusively for educational purposes, commercial organizations realized the potential of the Internet and connected to it, as well.

ISP

An ISP (Internet Service Provider) is a company that supplies Internet connectivity to home and business customers. For a monthly fee, the service provider usually provides a software package, username, password and access phone number. Equipped with a modem, you can then log on to the Internet and browse the World Wide Web, send and receive e-mail. For broadband access you typically receive the broadband modem hardware or pay a monthly fee for this equipment that is added to your ISP account billing.

In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company's networks to the Internet. ISPs themselves are connected to one another through Network Access Points (NAPs). ISPs may also be called IAPs (Internet Access Providers).

INTRANET

Intranet is the generic term for a collection of private computer networks within an organization. An intranet uses network technologies as a tool to facilitate communication between people or work groups to improve the data sharing capability and overall knowledge base of an organization's employees.

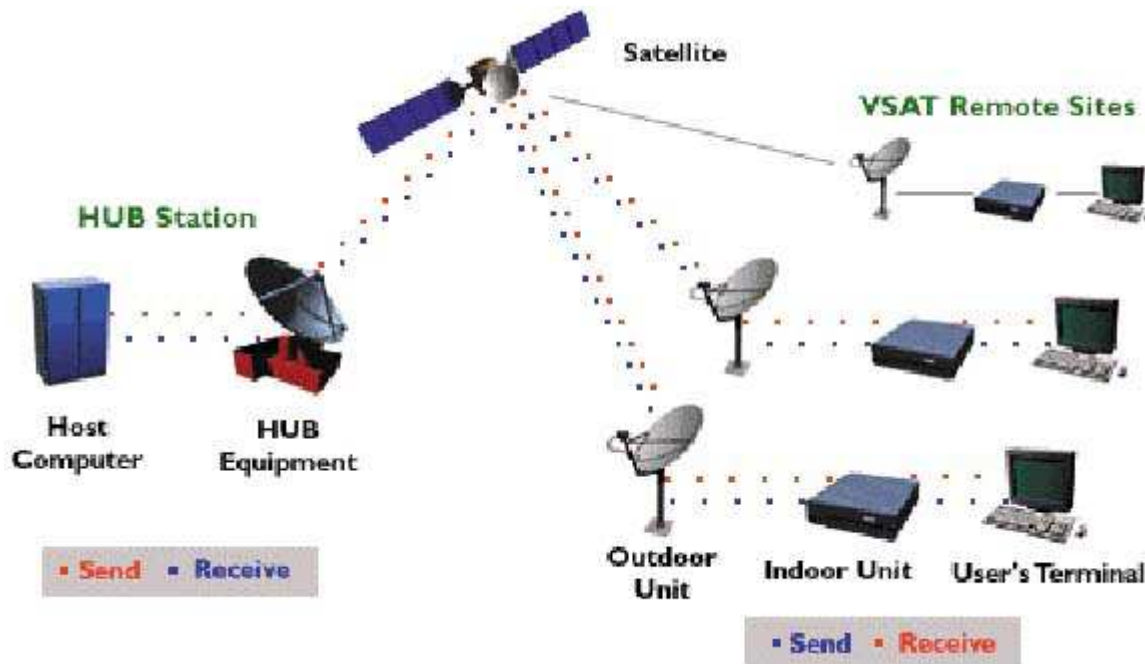
Intranets utilize standard network hardware and software technologies like Ethernet, Wi-Fi, TCP/IP, Web browsers and Web servers. An organization's intranet typically includes Internet access but is firewalled so that its computers cannot be reached directly from the outside.

A common extension to intranets, called extranets, opens this firewall to provide controlled access to outsiders.

Many schools and non-profit groups have deployed them, but an intranet is still seen primarily as a corporate productivity tool. A simple intranet consists of an internal email system and perhaps a message board service. More sophisticated intranets include Web sites and databases containing company news, forms, and personnel information. Besides company email and Internet access, an intranet generally incorporates internal Web sites, documents, and/or databases.

VSAT (VERY SMALL APERTURE TERMINAL)

VSAT (Very Small Aperture Terminal) is a satellite communications system that serves home and business users. A VSAT end user needs a box that interfaces between the user's computer and an outside antenna with a transceiver. The transceiver receives or sends a signal to a satellite transponder in the sky. The satellite sends and receives signals from an earth station computer that acts as a hub for the system. Each end user is interconnected with the hub station via the satellite in a star topology. For one end user to communicate with another, each transmission has to first go to the hub station which retransmits it via the satellite to the other end user's VSAT. VSAT handles data, voice, and video signals.

**URL (UNIFORM RESOURCE LOCATOR)**

A URL (Uniform Resource Locator, previously Universal Resource Locator). A URL is a formatted text string used by Web browsers, email clients and other software to identify a network resource on the Internet. Network resources are files that can be plain Web pages, other text documents, graphics, or programs.

URL strings consist of three parts (substrings):

- 1. network protocol**
- 2. host name or address**
- 3. file or resource location**

These substrings are separated by special characters as follows:

protocol **://** **host** **/** **location**

example : <http://www.sample.com/images/img1.htm>

URL Protocol

The 'protocol' substring defines a network protocol to be used to access a resource. These strings are short names followed by the three characters '://' (a simple naming convention to denote a protocol definition). Typical URL protocols include http://, ftp://, and mailto://.

URL Host

The 'host' substring identifies a computer or other network device. Hosts come from standard Internet databases such as DNS and can be names or IP addresses. For example, sample.com.

URL Location

The 'location' substring contains a path to one specific network resource on the host. Resources are normally located in a host directory or folder. For example, /images/img1.htm is the location of this Web page including a subdirectory and the file name.

PORTAL

A portal is a kind of Web site. The term originated with large, well-known Internet search engine sites that expanded their features to include email, news, stock quotes, and an array of other functionality. Some corporations took a similar approach in implementing their intranet sites, that then became known as enterprise information or corporate portals.

Technically speaking, a portal site includes a start page with rich navigation, a collection of loosely-integrated features. E.g. : www.yahoo.com, www.msn.com etc

DOMAIN NAME SERVER

The Domain Name System (DNS) is a standard technology for managing the names of Web sites and other Internet domains. DNS technology allows you to type names into your Web browser like sample.com and your computer to automatically find that address on the Internet.

DNS Root Servers

DNS servers communicate with each other using private network protocols. All DNS servers are organized in a hierarchy. At the top level of the hierarchy, so-called root servers store the complete database of Internet domain names and their corresponding IP addresses. The Internet employs 13 root servers that have become somewhat famous for their special role. Maintained by various independent agencies, the servers are aptly named A, B, C and so on up to M. Ten of these servers reside in the United States, one in Japan, one in London, UK and one in Stockholm, Sweden.

CHAPTER 2 : APPLICATION OF INTERNET

WORLD WIDE WEB(WWW)

The term WWW refers to the World Wide Web or simply the Web. The World Wide Web consists of all the public Web sites connected to the Internet worldwide, including the client devices (such as computers, Laptops and Mobile phones) that access Web content. The WWW is just one of many applications of the Internet and computer networks.

The World Web is based on these technologies:

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

Web servers and Web browsers

Researcher Tim Berners-Lee led the development of the original World Wide Web in the late 1980s and early 1990s. He helped build prototypes of the above Web technologies and coined the term "WWW." Web sites and Web browsing exploded in popularity during the mid-1990s.

SEARCH ENGINE

Search engines are programs that search documents for specified keywords and returns a list of the documents where the keywords were found. A search engine is really a general class of programs, however, the term is often used to specifically describe systems like Google, Bing and Yahoo! Search that enable users to search for documents on the World Wide Web.

Typically, Web search engines work by sending out a spider to fetch as many documents as possible. Another program, called an indexer, then reads these documents and creates an index based on the words contained in each document. Each search engine uses a proprietary algorithm to create its indices such that, ideally, only meaningful results are returned for each query.

REMOTE LOGIN

In Remote login there are two separate parties: the host computer and the remote user. To share a desktop, the host computer allows a remote user to view the contents of the host computer's desktop over the Internet. The host computer can also hand over keyboard and mouse controls to the remote user. With remote log-in, your home or work computer is the host and you (in this case) are the remote user.

Remote login requires three basic components:

Software download

Internet connection


Secure desktop sharing network

e.g. : goToMyPc, TeamViewer

TELNET

Telnet is a simple, text-based program that allows you to connect to another computer by using the Internet. If you've been granted the right to connect to that computer by that computer's owner or administrator, Telnet will allow you to enter commands used to access programs and services that are on the remote computer, as if you were sitting right in front of it. Telnet can be used for many things, including accessing e-mail, databases, or files.

To install Telnet Client

Click the Start button  Picture of the Start button, click Control Panel, and then click Programs.

Under Programs and Features, click Turn Windows features on or off.

In the Windows Features dialog box, select the Telnet Client check box.

Click OK. The installation might take several minutes.

FTP

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet. Like the Hypertext Transfer Protocol (HTTP), which transfers displayable Web pages and related files, and the Simple Mail Transfer Protocol (SMTP), which transfers e-mail, FTP is an application protocol that uses the Internet's TCP/IP protocols. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet. It's also commonly used to download programs and other files to your computer from other servers.

As a user, you can use FTP with a simple command line interface (for example, from the Windows MS-DOS Prompt window) or with a commercial program that offers a graphical user interface. Your Web browser can also make FTP requests to download programs you select from a Web page. Using FTP, you can also update (delete, rename, move, and copy) files at a server. You need to logon to an FTP server. However, publicly available files are easily accessed using anonymous FTP.

EMAIL

An application that runs on a personal computer or workstation and enables you to send, receive and organize e-mail. It's called a client because e-mail systems are based on a client-server architecture. Mail is sent from many clients to a central server, which re-routes the mail to its intended destination.

E-COMMERCE & E-BUSINESS

E-commerce (electronic commerce or EC) is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the Internet. These business transactions occur either business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business. The terms e-commerce and e-business are often used interchangeably.

E-business (electronic business) is the conduct of business processes on the Internet. These electronic business processes include buying and selling products, supplies and services; servicing customers; processing payments; managing production control; collaborating with business partners; sharing information; running automated employee services; recruiting; and more.

E-business can comprise a range of functions and services, ranging from the development of intranets and extranets to e-service, the provision of services and tasks over the Internet by application service providers.

E-GOVERNANCE

E-government is a generic term for web-based services from agencies of local, state and central governments. In e-government, the government uses information technology and particularly the Internet to support government operations, engage citizens, and provide government services. The interaction may be in the form of obtaining information, filings, or making payments and a host of other activities via the World Wide Web.

According to World Bank

E-Government refers to the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government. These technologies can serve a variety of different ends: better delivery of government services to citizens, improved interactions with business and industry, citizen empowerment through access to information, or more efficient government management. The resulting benefits can be less corruption, increased transparency, greater convenience, revenue growth, and/or cost reductions."

- G2C (government to citizens)
- G2B (government to businesses)
- G2E (government to employees)
- G2G (government to governments)

CHAPTER 3 : Basic of HTML & Advance HTML

FUNDAMENTALS OF HTML

HTML is a language for describing web pages.

HTML stands for Hyper Text Markup Language

- HTML is a markup language which is a set of markup tags that describes document content
- HTML documents contain HTML tags and plain text
- HTML documents are also called web pages
- HTML is a language for describing web pages.

USE OF HTML DOCUMENTS

The World Wide Web commonly known as the Web is a system of interlinked hypertext documents that are accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.

HTML TAGS

- HTML markup tags are usually called HTML tags.
- HTML tags are keywords (tag names) surrounded by angle brackets like <html>
- HTML tags normally come in pairs like <p> and </p>
- The first tag in a pair is the start tag, the second tag is the end tag
- The end tag is written like the start tag, with a slash before the tag name
- Start and end tags are also called opening tags and closing tags
<tagname>content</tagname>

HTML ELEMENTS

- In HTML, most elements are written with a start tag (e.g. <p>) and an end tag (e.g. </p>), with the content in between: <p>This is a paragraph.</p>

Advantages of HTML

- Easy to use
- Support on almost every browser.
- Widely used on almost every website.
- Free of cost.
- Easy to learn and code even novice programmer.

HTML Versions

Version	Year
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999

XHTML	2000
HTML5	2012

HTML Document Structure

```
<HTML>
  <HEAD>
    <TITLE>My first HTML document</TITLE>
  </HEAD>
  <BODY>
    Hello world!
  </BODY>
</HTML>
```

BASIC TAGS & ATTRIBUTES

TAGS

Tags are instructions that are embedded into the text of the documents. An HTML tag is a signal to a browser that it should do something other than just through text up on the screen.

Syntax :

```
< TAG > ... .. </TAG >
```

Note: Tags START with an open angle bracket (<) and END with close angle bracket (>).

Types of TAGS

TAGS are basically divided into two parts

- 1. Paired Tags :** The Tags written along with a companion tag is called PAIRED TAG.

For example,

The tag with its companion tag .

The text between this both tag are written in BOLD. The effect of paired tag is applied only to the text they contain. The OPENING tag activates the effect & the CLOSING tag closes the effect.

- 2. Singular Tags:** This tag is also known as STAND-ALONE tags. A singular tags does not have a companion tag. Some HTML tags require additional information to be supplied to them.

For example...

when a picture is placed on the screen, information like the HEIGHT and WIDTH of the Picture can be specified.

ATTRIBUTES

Additional Information supplied to an HTML Tag is called as ATTRIBUTES of Tags. Attributes are written immediately following the tag separated by a Space. Multiple Attributes can be associated with a Tag. Attribute are used with tag that provides the extract information and actions.

For example :

Here tag tells a web browser to display a graphical image but which image? Then src attribute give picture file path information and display that picture in the browser.

DOCUMENT TAG

<html> </html>

The <html> tag tells the browser that this is an HTML document. The <html> tag represents the root of an HTML document. The <html> tag is the container for all other HTML elements

<head> </head>

The <head> element is a container for all the head elements. The <head> element must include a title for the document, and can include scripts, styles, meta information, and more.

The following elements can go inside the <head> element: <title> (this element is required in the head section), <style>, <base>, <link>, <meta>, <script>, <noscript>

<body> </body>

The <body> tag defines the document's body. The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

Attribute	Value	Description
<u>alink</u>	color	Specifies the color of an active link in a document
<u>background</u>	URL	Specifies a background image for a document
<u>bgcolor</u>	color	Specifies the background color of a document
<u>link</u>	color	Specifies the color of unvisited links in a document
<u>text</u>	color	Specifies the color of the text in a document
<u>vlink</u>	color	Specifies the color of visited links in a document

<title> </title>

The <title> tag is required in all HTML documents and it defines the title of the document. The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search-engine results

<!-- -->

The comment tag is used to insert comments in the source code. Comments are not displayed in the browsers. You can use comments to explain your code, which can help you when you edit the source code at a later date. This is especially useful if you have a lot of code.

E.g. : <!--This is a comment. Comments are not displayed in the browser-->

TEXT FORMATTING TAG**<h1></h6>**

The <h1> to <h6> tags are used to define HTML headings. <h1> defines the most important heading. <h6> defines the least important heading.

E.g. : <h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>

<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

Attributes

Attribute	Value	Description
<u>align</u>	left center right justify	Specifies the alignment of a

<p>

The <p> tag defines a paragraph. Browsers automatically add some space (margin) before and after each <p> element.

Attributes

Attribute	Value	Description
<u>align</u>	left center right justify	Specifies the alignment of a

<Pre> </Pre>

The <pre> tag defines preformatted text. Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

** **

The tag specifies bold text.

E. g. : <p>This is normal text - and this is bold text.</p>

<U> </U>

The <u> tag represents some text that should be stylistically different from normal text, such as misspelled words or proper nouns in Chinese.

E. g. : <p>This is a <u>paragraph</u>.</p>

<I> </I>

The <i> tag defines a part of text in an alternate voice or mood. The content of the <i> tag is usually displayed in italic. The <i> tag can be used to indicate a technical term, a phrase from another language, a thought, or a ship name, etc.

E. g. : <p>He named his car <i>The lightning</i>, because it was very fast.</p>

<tt> </tt>

The <tt> tag defines teletype text.

E. g. : <p><tt>Teletype text</tt> </p>

The <sub> tag defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H₂O.

E. g. : <p>This text contains _{subscript} text.</p>

Output : This text contains _{subscript} text.

<Strike> </Strike>

The <strike> tag defines strikethrough text.

E. g. : <p>Version 2.0 is <strike>not yet available!</strike> now available!</p>

The <sup> tag defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font.

E. g. : <p>This text contains ^{superscript} text.</p>

<Big> </Big>

The <big> tag defines bigger text.

E. g. : <p><big>Bigger text</big> </p>

** **

The tag is a phrase tag. It defines important text.

E. g. : Strong text

<Small> </Small>

The <small> tag defines smaller text (and other side comments).

E. g. : Shri V. J. Modha College<small>of Information & Technology </small>

** **

The tag specifies the font face, font size, and color of text.

Optional Attributes

Attribute	Value	Description
<u>color</u>	<i>rgb(x,x,x)</i> <i>#xxxxxx</i> <i>colorname</i>	Specifies the color of text
<u>face</u>	<i>font_family</i>	Specifies the font of text

<u>size</u>	<i>number</i>	Specifies the size of text
-------------	---------------	----------------------------

E. g. : `This is some text!`

`This is some text!`

`This is some text!`

<blink> </blink>

The HTML Blink Element (`<blink>`) is a non-standard element causing the enclosed text to flash slowly.

e.g. : `<blink> Example of blink tag.</blink>`

<marquee> </marquee>

The HTML `<marquee>` element is used to insert a scrolling area of text.

Attribute	Value	Description
behavior	<i>Scroll</i> <i>Slide</i> <i>Alternate</i>	Sets how the text is scrolled within the marquee. Possible values are scroll, slide and alternate. If no value is specified, the default value is scroll.
bgcolor		Sets the background color through color name or hexadecimal value.
direction	Left Right Up Down	Sets the direction of the scrolling within the marquee. Possible values are left, right, up and down. If no value is specified, the default value is left.
Height		Sets the height in pixels or percentage value.
Hspace		Sets the horizontal margin
Loop		Sets the number of times the marquee will scroll. If no value is specified, the default value is -1, which means the marquee will scroll continuously.

Scrollamount		Sets the amount of scrolling at each interval in pixels. The default value is 6.
Scrolldelay		Sets the interval between each scroll movement in milliseconds. The default value is 85. Note that any value smaller than 60 is ignored and the value 60 is used instead, unless truespeed is specified.
Truespeed		By default, scrolldelay values lower than 60 are ignored. If truespeed is present, those values are not ignored.
Vspace		Sets the vertical margin in pixels or percentage value.
Width		Sets the width in pixels or percentage value.

e.g. : `<marquee>This text will scroll from right to left</marquee>`

`<marquee direction="up">This text will scroll from bottom to top</marquee>`

`<marquee direction="down" width="250" height="200" behavior="alternate" style="border:solid">`

`<marquee behavior="alternate">`

This text will bounce

`</marquee>`

`</marquee>`

List Creation

** **

The `` tag defines an ordered list. An ordered list can be numerical or alphabetical. Use the `` tag to define list items.

Attribute	Value	Description
<u>reversed</u>	reversed	Specifies that the list order should be descending (9,8,7...)
<u>start</u>	<i>number</i>	Specifies the start value of an ordered list
<u>type</u>	1 A a I	Specifies the kind of marker to use in the list

	i	
--	---	--

e.g. `<ol type=i>`
`Coffee`
`Tea`
`Milk`
``

`<ol start="50">`
`Coffee`
`Tea`
`Milk`
``

** **

The `` tag defines an unordered (bulleted) list. Use the `` tag together with the `` tag to create unordered lists.

Attribute	Value	Description
<u>type</u>	disc square circle	Specifies the kind of marker to use in the list

**<il> **

The `` tag defines a list item. The `` tag is used in ordered lists(``), unordered lists (``), and in menu lists (`<menu>`).

CREATING LINK WITH OTHER PAGES**<A> **

The `<a>` tag defines a hyperlink, which is used to link from one page to another. The most important attribute of the `<a>` element is the href attribute, which indicates the link's destination.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Attribute	Value	Description
<u>download</u>	<i>filename</i>	Specifies that the target will be downloaded when a user clicks on the hyperlink

<u>href</u>	URL	Specifies the URL of the page the link goes to
<u>shape</u>	default rect circle poly	Specifies the shape of a link

e.g. : `Visit google.com!`

<Link>

The `<link>` tag defines the relationship between a document and an external resource.

The `<link>` tag is most used to link to style sheets.

Attribute	Value	Description
<u>href</u>	URL	Specifies the location of the linked document
<u>target</u>	_blank _self _top _parent frame_name	Specifies where the linked document is to be loaded

e.g. : `<head>`

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

`</head>`

INSERTING SPECIAL CHARACTERS, ADDING IMAGES AND SOUND

INSERTING SPECIAL CHARACTERS

Result	Description	Entity Name	Entity Number
	non-breaking space	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥

€	euro	€	€
©	copyright	©	©
®	registered trademark	®	®
•	Rupee Symbol		₹

INSERTING IMAGES

The tag defines an image in an HTML page. It has two required attributes: src and alt.

Attribute	Value	Description
<u>align</u>	top bottom middle left right	Specifies the alignment of an image according to surrounding elements
<u>alt</u>	<i>text</i>	Specifies an alternate text for an image
<u>border</u>	<i>pixels</i>	Specifies the width of the border around an image
<u>height</u>	<i>pixels</i>	Specifies the height of an image
<u>hspace</u>	<i>pixels</i>	Specifies the whitespace on left and right side of an image
<u>src</u>	<i>URL</i>	Specifies the URL of an image
<u>vspace</u>	<i>pixels</i>	Specifies the whitespace on top and bottom of an image
<u>width</u>	<i>pixels</i>	Specifies the width of an image

e.g. :

INSERTING SOUND

Using Embed

The <embed> tag defines a container for external (non-HTML) content.

e.g.: <embed height="50" width="300" src=" Song1.mp3">

Using Object

The <object> tag can also define a container for external (non-HTML) content.

e.g.: <object height="50" width="100" data=" Song1.mp3"> </object>

Using Hyperlink

e.g.: `Play the sound`

Using HTML Audio

The HTML5 `<audio>` tag defines sound, such as music or other audio streams. The `<audio>` element works in all modern browsers. The following example uses the HTML5 `<audio>` tag, which specifies one MP3 file (for Internet Explorer, Chrome, Firefox 21+, and Safari), and one OGG file (for older Firefox and Opera). If something fails, it will display a text:

e.g.:

```
<audio controls>
  <source src="Song1.mp3" type="audio/mpeg">
  <source src="Song1.ogg" type="audio/ogg">
  Your browser does not support this audio format.
</audio>
```

LISTS TYPES OF LISTS

<dl> </dl>

The `<dl>` tag defines a description list. The `<dl>` tag is used in conjunction with `<dt>` (defines terms/names) and `<dd>` (describes each term/name).

<dt> </dt>

The `<dt>` tag defines a term/name in a description list.

<dd> </dd>

The `<dd>` tag is used to describe a term/name in a description list.

e.g. : `<dl>`

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

TABLE CREATION

<table> </table>

The `<table>` tag defines an HTML table. An HTML table consists of the `<table>` element and one or more `<tr>`, `<th>`, and `<td>` elements. The `<tr>` element defines a table row, the `<th>` element defines a table header, and the `<td>` element defines a table cell.

Attribute	Value	Description
<u>align</u>	left center right	Specifies the alignment of a table according to surrounding text
<u>bgcolor</u>	<i>rgb(x,x,x)</i> <i>#xxxxxx</i> <i>colorname</i>	Specifies the background color for a table
<u>border</u>	1 ""	Specifies whether the table cells should have borders or not
<u>cellpadding</u>	<i>pixels</i>	Specifies the space between the cell wall and the cell content
<u>cellspacing</u>	<i>pixels</i>	Specifies the space between cells
<u>frame</u>	void above below hsides lhs rhs vsides box border	Specifies which parts of the outside borders that should be visible
<u>rules</u>	none groups rows cols all	Specifies which parts of the inside borders that should be visible
<u>width</u>	<i>pixels</i> %	Specifies the width of a table

<tr> </tr>

The <tr> tag defines a row in an HTML table. A <tr> element contains one or more <th> or <td> elements.

Attribute	Value	Description
<u>align</u>	right left center justify char	Aligns the content in a table row

<u>bgcolor</u>	<i>rgb(x,x,x)</i> <i>#xxxxxx</i> <i>colorname</i>	Specifies a background color for a table row
<u>valign</u>	top middle bottom baseline	Not supported in HTML5. Vertical aligns the content in a table row

<th></th> & <td></td>

An HTML table has two kinds of cells:

- Header cells - contains header information (created with the <th> element)
- Standard cells - contains data (created with the <td> element)

The <th> tag defines a header cell in an HTML table. The text in <th> elements are bold and centered by default.

The <td> tag defines a standard cell in an HTML table. The text in <td> elements are regular and left-aligned by default.

Attribute	Value	Description
<u>align</u>	left right center justify char	Aligns the content in a header cell
<u>bgcolor</u>	<i>rgb(x,x,x)</i> <i>#xxxxxx</i> <i>colorname</i>	Specifies the background color of a header cell
<u>colspan</u>	<i>number</i>	Specifies the number of columns a header cell should span
<u>height</u>	<i>pixels</i> <i>%</i>	Sets the height of a header cell
<u>rowspan</u>	<i>number</i>	Specifies the number of rows a header cell should span
<u>valign</u>	top middle bottom baseline	Vertical aligns the content in a header cell
<u>width</u>	<i>pixels</i>	Specifies the width of a header cell

	%	
--	---	--

e.g.: `<table>`

```

<tr>
  <th>Sr.</th>  <th>Name</th>
</tr>
<tr>
  <td>1</td>  <td>ABC</td>
</tr>
</table>

```

<caption>

The `<caption>` tag defines a table caption. The `<caption>` tag must be inserted immediately after the `<table>` tag.

e.g. : `<table>`

```

<caption>Monthly savings</caption>

<tr>
  <th>Sr.</th>  <th>Name</th>
</tr>
<tr>
  <td>1</td>  <td>ABC</td>
  <td>2</td>  <td>XYZ</td>
</tr>
</table>

```

FRAMES

<frameset> </frameset>

The `<frameset>` element holds one or more `<frame>` elements. Each `<frame>` element can hold a separate document. The `<frameset>` element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

Attribute	Value	Description
<u>cols</u>	<i>pixels</i> % *	Specifies the number and size of columns in a frameset
<u>rows</u>	<i>pixels</i> %	Specifies the number and size of rows in a frameset

*

<frame> </frame>

The <frame> tag defines one particular window (frame) within a <frameset>. Each <frame> in a <frameset> can have different attributes, such as border, scrolling, the ability to resize, etc.

Attribute	Value	Description
<u>frameborder</u>	0 1	Specifies whether or not to display a border around a frame
<u>marginheight</u>	<i>pixels</i>	Specifies the top and bottom margins of a frame
<u>marginwidth</u>	<i>pixels</i>	Specifies the left and right margins of a frame
<u>scrolling</u>	yes no auto	Specifies whether or not to display scrollbars in a frame
<u>src</u>	<i>URL</i>	Specifies the URL of the document to show in a frame

e.g. : <frameset cols="25%,*,25%">

<frame src="frame_a.htm">

<frame src="frame_b.htm">

<frame src="frame_c.htm">

</frameset>

FORM**<form> </form>**

HTML forms are used to pass data to a server. An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements. The <form> tag is used to create an HTML form:

<form>

.

input elements

.

</form>

INPUT ELEMENTS

The most important form element is the <input> element. The <input> element is used to select user information. An <input> element can vary in many ways, depending on the type attribute. An <input> element can be of type text field, checkbox, password, radio button, submit button, and more.

1. Text Fields

<input type="text"> defines a one-line input field that a user can enter text into:

e.g.:

<form>

First name: <input type="text" name="firstname">

Last name: <input type="text" name="lastname">

</form>

2. Password Field

<input type="password"> defines an input field in which text is covered by bullets.

e.g.:

<form>

Password: <input type="password" name="pwd">

</form>

3. Radio Buttons

<input type="radio"> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

e.g.:

<form>

<input type="radio" name="gender" value="male">Male

<input type="radio" name="gender" value="female">Female

</form>

4. Checkboxes

<input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

e.g.:

<form>

<input type="checkbox" name="vehicle" value="Bike">I have a bike

<input type="checkbox" name="vehicle" value="Car">I have a car

</form>

5. Submit Button

`<input type="submit">` defines a submit button. A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

e.g.:

```
<form name="input" action="test2.html" method="get">
```

```
Username: <input type="text" name="user">
```

```
<input type="submit" value="Submit">
```

```
</form>
```

6. Button

The button INPUT tag gives you a way to create simple input buttons with custom text. It is similar to the INPUT submit tag but is slightly more flexible.

e.g.:

```
<form>
```

```
<input type="button" value="this is a text button">
```

```
</form>
```

7. File

The file INPUT tag provides a file upload form box for HTML forms. It allows you to solicit files from your readers such as images or video.

e.g.:

```
<form enctype="multipart/form-data">
```

```
<input type="file" accept="image/jpg,image/gif">
```

```
</form>
```

8. Hidden

The hidden INPUT tag allows HTML form developers to store information for delivery with the form data that is invisible to the form users. The hidden form fields are used to save state in multi-page forms, collect cookie data to deliver to the form, and just store data beyond what the customer fills in.

e.g.:

```
<form
```

```
<input type="hidden" name="date-submitted" value="2010-12-10">
```

```
</form>
```

9. Reset

The reset INPUT tag sets the form back to its initial state when the form was loaded. In most forms this means that it clears the contents of the form, but if the form started out with pre-loaded contents, that is what will display when the reset button is clicked.

e.g.:

```
<form>
<input type="reset">
</form>
```

<select>

The <select> element is used to create a drop-down list. The <option> tags inside the <select> element define the available options in the list.

<option>

The <option> tag defines an option in a select list. <option> elements go inside a <select> or <datalist> element.

e.g. :

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>
```

<textarea>

The <textarea> tag defines a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). The size of a text area can be specified by the cols and rows attributes

Attribute	Value	Description
<u>autofocus</u>	autofocus	Specifies that a text area should automatically get focus when the page loads
<u>cols</u>	<i>number</i>	Specifies the visible width of a text area
<u>disabled</u>	disabled	Specifies that a text area should be disabled
<u>form</u>	<i>form_id</i>	Specifies one or more forms the text area belongs to

<u>maxlength</u>	<i>number</i>	Specifies the maximum number of characters allowed in the text area
<u>name</u>	<i>text</i>	Specifies a name for a text area
<u>placeholder</u>	<i>text</i>	Specifies a short hint that describes the expected value of a text area
<u>readonly</u>	readonly	Specifies that a text area should be read-only
<u>required</u>	required	Specifies that a text area is required/must be filled out
<u>rows</u>	<i>number</i>	Specifies the visible number of lines in a text area
<u>wrap</u>	hard soft	Specifies how the text in a text area is to be wrapped when submitted in a form

e.g. :

```
<textarea rows="4" cols="50">
```

Welcome to our College. It provides education and placement facilities.

```
</textarea>
```

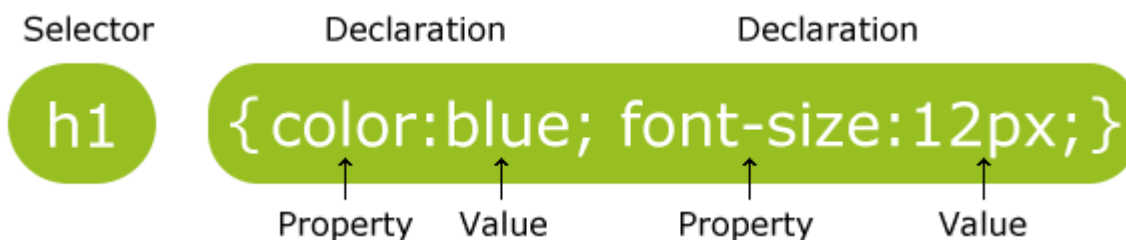
CHAPTER 4 : CASCADING STYLE SHEET

INTRODUCTION TO CSS

A CSS (cascading style sheet) file allows you to separate your web sites (X)HTML content from it's style Use your (X)HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are formatted within a CSS. CSS defines HOW HTML elements are to be displayed. External style sheets enable you to change the appearance and layout of all web pages just by editing one single file!. Styles are normally saved in external .css files. Styles were added to HTML 4.0 and helps user to save a lot of work.

Syntax

A CSS consists of a **selector** and a **declaration** block:



1. The **selector** points to the HTML element you want to style.
2. The **declaration** block contains one or more declarations separated by semicolons. Each declaration includes a property name and a value, separated by a colon.

TYPES OF STYLESHEETS

There are three ways of inserting a style sheet:

- Inline style
- Internal style sheet
- External style sheet

Inline style sheet

To use inline styles you just have to add the style attribute to the relevant tag.

e.g. : `<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>`

Internal style sheet

An internal style sheet is used when you want to specify unique style to a single web page. It must be defined in the head section of an HTML page, inside the `<style>` tag, like this:

e.g. :

```
<head>
```

```
<style>
```

```
body {
```

```
    background-color: linen;
```

```
}
```

```
h1 {  
    color: maroon;  
    margin-left: 40px;  
}  
</style>  
</head>
```

External style sheet

With an external style sheet, you can change the look of an entire Web site by changing just one file. An external style sheet is ideal when the style is applied to many pages.

Each page must include a link to the style sheet with the <link> tag which goes inside the head section:

```
e.g.:  
<head>  
<link rel="stylesheet" type="text/css" href="SampleSheet.css">  
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

" SampleSheet.css":

```
body {  
    background-color: lightblue;  
}  
h1 {  
    color: navy;  
    margin-left: 20px;  
}
```

If multiple style sheets are used...

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, assume that an external style sheet has the following properties for the h1 selector:

```
h1 {  
    color: navy;  
    margin-left: 20px;  
}
```

then, assume that an internal style sheet also has the following property for the h1 selector:

```
h1 {  
    color: orange;  
}
```

If the page with the internal style sheet also links to the external style sheet the properties for the h1 element will be:

```
color: orange;  
margin-left: 20px;
```

Cascading order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

- Browser default
- External style sheet
- Internal style sheet (in the head section)
- Inline style (inside an HTML element)

CSS SELECTORS

CSS selectors allow you to select and manipulate HTML element(s). CSS selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more.

The element Selector

The element selector selects elements based on the element name. You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

E.g. :

```
p {
  text-align: center;
  color: red;
}
```

The id Selector

It uses the id attribute to find the specific element. An id should be unique within a page, so only a single & unique element is found. To find an element write a hash character before the id of the element. (#idName)

E.g. :

```
#para1 {
  text-align: center;
  color: red;
}
<body>    <p id=para1>Sample</p>    </body>
```

The class Selector

The class selector finds elements with the specific class. It uses the HTML class attribute. To find elements with a specific class, write a period character, followed by the name of the class.

E.g. :

```
<style>
```

```
.center {
  text-align: center;
  color: red;
}
</style>
<body>    <h1 class="center">Red and center-aligned heading</h1>    </body>
```

You can also specify that only specific HTML elements should be affected by a class. In the example below, all p elements with class="center" will be center-aligned:

```
p.center {
  text-align: center;
  color: red;
}
<body>    <p class="center">This paragraph will be red and center-aligned.</p>    </body>
```

MINIMISING GROUP

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

CSS FONT PROPERTIES

CSS font properties define the font family, boldness, size, and the style of a text.

1. FONT FAMILY

It is used to define fonts. If more than one font is specified and the browser does not support the first font, it tries the next font.

```
p {
  font-family: "Times New Roman", Arial;
}
```

2. FONT STYLE

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {
  font-style: normal;
}
```

```
p.italic {
  font-style: italic;
}
```

3. FONT SIZE

The font-size property sets the size of the text. You can set size with either Pixels or Em. To allow users to resize the text (in the browser menu), many developers use em instead of pixels. The em size unit is recommended by the W3C. 1em is equal to the 16px.

e.g.:

```
body {  
  font-size: 100%;  
}
```

```
h1 {  
  font-size: 2.5em;  
}
```

```
h2 {  
  font-size: 1.875em;  
}
```

4. FONT-WEIGHT :

Used to make fonts lighter or Bolder.

e.g.:

```
h1 {  
  font-weight: normal;  
}  
H2{  
  font-weight: lighter;  
}
```

CSS TEXT PROPERTIES

1. TEXT COLOR

It is used to set the color of the text. It can be specified by

a HEX value - like "#ff0000"

an RGB value - like "rgb(255,0,0)"

a color name - like "red"

The default color for a page is defined in the body

e.g. : body {

```
  color: blue;
```

```
}
```

```
h1 {
```

```
  color: #00ff00;
```

```
}
```

```
h2 {
```

```
  color: rgb(255,0,0);
```

```
}
```

2. TEXT ALIGNMENTS

Used to align text. Possible values are Center, Right and Justify.

```
e.g.: h1 {  
    text-align: center;  
}
```

3. TEXT TRANSFORMATION

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

```
e.g.: h1{  
    text-transform: uppercase;  
}
```

4. TEXT INDENTATION

The text-indent property is used to specify the indentation of the first line of a text.

```
p {  
    text-indent: 50px;  
}
```

5. TEXT DECORATION

The text-decoration property is used to Create Overline, Line-through or Underline text.

```
Eg. h1 {  
    text-decoration: overline;  
}  
h2 {  
    text-decoration: line-through;  
}  
h3 {  
    text-decoration: underline;  
}
```

CSS BACKGROUND PROPERTIES

CSS background properties are used to define the background effects of an element. CSS properties used for background effects:

1. BACKGROUND COLOR

It specifies the background color of an element & it is defined in the body selector

```
e.g. :  
body {  
    background-color: #3b5998;  
}
```

2. BACKGROUND IMAGE

It specifies the background Image for a page.

```
e.g. :  
body {  
    background-image: url("gradient_bg.png");  
}
```

3. BACKGROUND REPEAT

It is used to specify how and when the image will be repeated.

```
e.g. : body {  
    background-image: url("gradient_bg.png");  
    background-repeat: repeat-x; /* repeat horizontally */  
    background-repeat: repeat-y; /* repeat Vertically */  
    background-repeat: no-repeat-y; /* repeat once */  
}
```

4. BACKGROUND ATTACHMENT

Sets whether a background image is fixed or scrolls with the rest of the page

```
e.g. : body {  
    background-image: url('w3css.gif');  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
}
```

5. BACKGROUND POSITION

```
e.g. : body {  
    background-position: right top;  
  
}
```

6. BACKGROUND Size

Sets the background Size

```
e.g. : body {  
    background-size: 800px 600px;  
    background-size: 100%; /* Fit to Page */  
}
```

CSS LIST PROPERTIES

The CSS list properties allow you to:

1. Set different list item markers for ordered lists
2. Set different list item markers for unordered lists
3. Set an image as the list item marker

1. Different List Item Markers

The type of list item marker is specified with the list-style-type property:

Possible Values :

Ordered List : Circle, Square, Disc.

Unordered List : Lower-alpha, Upper-alpha, Lower-roman, Upper-roman, Number.

```
<style>  
/* ordered */ ul { list-style-type: circle; }
```



```

/* Unordered */ ol                                { list-style-type: upper-roman; }
</style>
<body>
<p>Example of unordered lists:</p>                <p>Example of ordered lists:</p>
<ul>                                              <ol>
  <li>Coffee</li>                                <li>Coffee</li>
  <li>Tea</li>                                   <li>Tea</li>
</ul>                                             </ol>
</body>

```

2. Different Image as Marker

To specify an image as the list item marker, use the list-style-image property:

```

<style> ul{ list-style-image: url('1.jpg'); } </style>
<body> <ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
</body>

```

CSS MARGIN PROPERTIES

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Value	Description
<i>length</i>	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element

E.G.:

```

p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 50px;
}

```

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 starts with /* and ends with */.

CSS BORDER PROPERTIES

The CSS border properties allow you to specify the style, size, and color of an element's border.

1. Border Style

The border-style property specifies what kind of border to display.

e.g. :

```
p {
```

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

```
}
```

2. Border Width

The border-style property specifies what kind of border to display. The border-width property is used to set the width of the border. The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

e.g.:

```
{
```

```
border-style: solid;
```

```
border-width: 1px;
```

```
}
```

3. Border Color

The border-color property is used to set the color of the border

e.g.:

```
{
```

```
border-style: solid;
```

```
border-color: red;
```

```
}
```

4. Border - Individual sides

In CSS it is possible to specify different borders for different sides:

e.g.:

```
{
```

```
border-top-style: dotted;
```

```
border-right-style: solid;
```

```
border-bottom-style: dotted;
```

```
border-left-style: solid;
```

```
}
```

Padding

The padding clears an area around the content (inside the border) of an element. The top, right, bottom, and left padding can be changed independently.

Possible Values

Value	Description
<i>length</i>	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

e.g. :

```
p {  
  padding-top: 25px;  
  padding-right: 50px;  
  padding-bottom: 25px;  
  padding-left: 50px;  
}
```

CHAPTER 6 : INTRODUCTION TO JAVASCRIPT

WHAT IS SCRIPTING LANGUAGE

Scripting languages are programming languages that are interpreted or compiled each time they run. Scripts are executed directly from their source code, which are generally text files containing language specific markup.

Scripts are typically compiled from source code into binary executable files then script will run from these binary files without needing the source code.

INTRODUCTION TO JAVASCRIPT

JavaScript is a web programming language, that enables you to control how a web page behaves. This makes JavaScript different from HTML which gives structure to your web pages and CSS which controls the appearance of web pages.

JavaScript gives you the freedom to add interactivity and responsiveness to your web pages. JavaScript is a lightweight, easy to learn, scripting language. It's used on almost every website to respond to user actions, validate web forms, detect browser support, and much more.

JavaScript is a scripting language that will allow you to add real programming to your webpages. You can create small application type processes with javascript, like a calculator.

ADVANTAGES OF JAVASCRIPT

- **JavaScript gives HTML designers a programming tool by putting small "snippets" of code into their HTML pages**
- **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page
- **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

Where to Put the JavaScript

Scripts in the head section

Scripts to be executed when the page loads is written in the head section.

```
<html>
<head>
  <script type="text/javascript">
    document.write("Hello");
```

```
</script>
</head>
<body>
World!
</body>
</html>
```

Output : Hello World!

Scripts in the body section

Scripts to be executed when they are called, or when an event is triggered, is written in the body section.

```
<html>
<head>
</head>
<body>
Hello
    <script type="text/javascript">
        document.write("Students");
    </script>
</body>
</html>
```

Output : Hello Students

Scripts in both the body and the head section

You can place an unlimited number of scripts in your document, you can insert scripts in the body and the head section together

```
<html>
<head>
    <script type="text/javascript">
        document.write("Students");
    </script>
</head>
<body>
    <script type="text/javascript">
        document.write("of Shri V J Modha College");
    </script>
</body>
```

Output: Students of Shri V J Modha College

Using an External JavaScript

Sometimes you might want to run the same JavaScript on several pages, You can do it by writing a JavaScript in an external file. It must be saved with .js file extension.

Note: The external script cannot contain the <script> tag!

To use the external script, point to the .js file in the "src" attribute of the <script> tag:

```
<html>
```

```
<head>
```

```
    <script src="script.js"> </script>
```

```
</head>
```

```
<body>
```

```
    Students
```

```
</body>
```

```
</html>
```

script.js

```
document.write("Good Morning");
```

Output: Good Morning Students

JAVASCRIPT VARIABLES

JavaScript uses the var keyword to define variables. An equal sign is used to assign values to variables

RULES FOR VARIABLE NAME

- Variable names can contain letters, digits, underscores, and dollar signs.
- Variable names must begin with a letter or \$ or _(underscore)
- Variable names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as variable names

JAVASCRIPT DATA TYPES

- When you assign a string value to a variable, you put double or single quotes around the value.
- When you assign a numeric value to a variable, you do not put quotes around the value. If you put quotes around a numeric value, it will be treated as a text string.

```
var pi = 3.14;
```

```
var person = "ABC";
```

```
var answer = 'XYZ';
```

DECLARING (CREATING) JAVASCRIPT VARIABLES

In JavaScript, the equal sign (=) is an "assignment" operator, is not an "equal to" operator.

You declare JavaScript variables with the var keyword:

```
var carName;
```

After the declaration, the variable is empty (it has no value).

To assign a value to the variable, use the equal sign:

```
carName = "Maruti Suzuki";
```

You can also assign a value to the variable when you declare it:

```
var carName = "Volvo";
```

You can declare many variables in one statement. Start the statement with var and separate the variables by comma:

```
var lastName = "Sharma", age = 25, job = "CEO";
```


JAVASCRIPT OPERATORS

The assignment operator

= is used to assign values,

+ is used to add values,

++ is used to increment values.

The arithmetic operator + is used to add values together.

```
<html>
```

```
<body>
```

```
<input type=button onclick="myFunction()" value=try>
```

```
<p id="demo"></p>
```

```
<script>
```

```
function myFunction() {
```

```
  y = 5;
```

```
  z = 2;
```

```
  x = y + z;
```

```
  document.write(x) }
```

```
</script>
```

```
</body>
```

```
</html>
```

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values. Given that $y = 5$, the table below explains the arithmetic operators:

Operator	Description	Example	Result	Result
+	Addition	$x = y + 2$	$y = 5$	$x = 7$
-	Subtraction	$x = y - 2$	$y = 5$	$x = 3$
*	Multiplication	$x = y * 2$	$y = 5$	$x = 10$
/	Division	$x = y / 2$	$y = 5$	$x = 2.5$
%	Modulus (division remainder)	$x = y \% 2$	$y = 5$	$x = 1$
++	Increment	$x = ++y$	$y = 6$	$x = 6$
		$x = y++$	$y = 6$	$x = 5$
--	Decrement	$x = --y$	$y = 4$	$x = 4$

		<code>x = y--</code>	<code>y = 4</code>	<code>x = 5</code>
--	--	----------------------	--------------------	--------------------

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that `x = 10` and `y = 5`, the table below explains the assignment operators:

Operator	Example	Same As	Result
<code>=</code>	<code>x = y</code>	<code>x = y</code>	<code>x = 5</code>
<code>+=</code>	<code>x += y</code>	<code>x = x + y</code>	<code>x = 15</code>
<code>-=</code>	<code>x -= y</code>	<code>x = x - y</code>	<code>x = 5</code>
<code>*=</code>	<code>x *= y</code>	<code>x = x * y</code>	<code>x = 50</code>
<code>/=</code>	<code>x /= y</code>	<code>x = x / y</code>	<code>x = 2</code>
<code>%=</code>	<code>x %= y</code>	<code>x = x % y</code>	<code>x = 0</code>

COMPARISON OPERATORS

Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that `x=5`, the table below explains the comparison operators:

Operator	Description	Comparing	Returns
<code>==</code>	equal to	<code>x == 8</code>	false
		<code>x == 5</code>	true
<code>===</code>	equal value and equal type	<code>x === "5"</code>	false
		<code>x === 5</code>	true
<code>!=</code>	not equal	<code>x != 8</code>	true
<code>!==</code>	not equal value or not equal type	<code>x !== "5"</code>	true
		<code>x !== 5</code>	false

>	greater than	$x > 8$	false
<	less than	$x < 8$	true
>=	greater than or equal to	$x \geq 8$	false
<=	less than or equal to	$x \leq 8$	true

LOGICAL OPERATORS

Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that $x=5$, the table below explains the comparison operators:

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x == 5 \ \ y == 5)$ is false
!	not	$!(x == y)$ is true

CONDITIONAL OPERATOR

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename = (condition) ? value1:value2

Example

voteable = (age < 18) ? "Too young":"Old enough";

CONDITIONAL STATEMENTS

Conditional statements are used to perform different actions based on different conditions.

In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

THE IF STATEMENT

Use the if statement to specify a block of JavaScript code to be executed if a condition is true. Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

Syntax: if (condition) {
 block of code to be executed if the condition is true
 }

Example: <script type="text/javascript">
 var no = 20;
 if(no>5)
 {
 document.write("Number is greater than 5");
 }
 </script>

THE IF ELSE STATEMENT

Use the else statement to specify a block of code to be executed if the condition is false.

Syntax: if (condition) {
 block of code to be executed if the condition is true
 }
 else {
 block of code to be executed if the condition is false
 }

Example: <script type="text/javascript">
 var no = 20;
 if(no>5)
 {
 document.write("Number is greater than 5");
 }
 else
 {
 document.write("Number is Less than 5");
 }
 </script>

THE ELSE IF STATEMENT

Use the else if statement to specify a new condition if the first condition is false

Syntax: if (condition) {
 block of code to be executed if the condition is true
 }

```
else if (condition2) {  
    block of code to be executed if the condition1 is false and condition2 is true  
} else {  
    block of code to be executed if the condition1 is false and condition2 is false  
}
```

Example: `<script type="text/javascript">`

```
var no = 0;  
if(no>5)  
{  
document.write("Number is greater than 0");  
}  
else if(no==0)  
{  
document.write("Number is 0");  
}  
else  
{  
document.write("Number is Less than 0");  
}  
</script>
```

THE JAVASCRIPT SWITCH STATEMENT

Use the switch statement to select one of many blocks of code to be executed.

Syntax: `witch(expression) {`

```
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        default code block  
}
```

Example: `switch (new Date().getDay()) {`

```
    case 0:  
        day = "Sunday";  
        break;  
    case 1:  
        day = "Monday";  
        break;
```

```
    case 2:
      day = "Tuesday";
      break;
    case 3:
      day = "Wednesday";
      break;
    case 4:
      day = "Thursday";
      break;
    case 5:
      day = "Friday";
      break;
    case 6:
      day = "Saturday";
      break;
  }
  document.write(day);
```

THE BREAK KEYWORD

When the JavaScript code interpreter reaches a break keyword, it breaks out of the switch block. This will stop the execution of more execution of code and/or case testing inside the block.

THE DEFAULT KEYWORD

The default keyword specifies the code to run if there is no case match:

Example

If today is neither Saturday nor Sunday, write a default message:

```
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

JAVASCRIPT LOOPS

Loops are handy, if you want to run the same code over and over again, each time with a different value.

JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times

- for/in - loops through the properties of an object
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

THE FOR LOOP

The for loop is often the tool you will use when you want to create a loop.

Syntax: for (statement 1; statement 2; statement 3) {
 code block to be executed
 }

Statement 1 is executed before the loop (the code block) starts. Statement 2 defines the condition for running the loop (the code block). Statement 3 is executed each time after the loop (the code block) has been executed.

Example: <script type="text/javascript">
 for (i = 1; i <= 5; i++) {
 document.write(i);
 }
 document.write(day);
 </script>

Note: *An expression is like a phrase; a statement is like a complete sentence; and a statement block is like a paragraph. For example, the following is an expression:*

2+3

This code does not do anything other than calculates a result. The following is a statement:

x=2+3;

A simple expression is nothing more than a function call. This can be two data items joined by an operator;

THE FOR/IN LOOP

The JavaScript for/in statement loops through the properties of an object: The for...in statement is used to loop (iterate) through the elements of an array or through the properties of an object.

Example: for (n in array)
 {
 }
 }

Example: <script type="text/javascript">
 var n;
 var cities = new Array();

```
cities[0] = "Porbandar";
cities[1] = "Rajkot";
cities[2] = "Ahmedabad";

for (n in cities)
{
    document.write(cities[n] + "<br />");
}
</script>
```

THE WHILE LOOP

The while loop loops through a block of code as long as a specified condition is true.

Syntax: while (condition) {
 code block to be executed
 }

Example: <script type="text/javascript">
 var i=0;
 while (i <= 10) {
 document.write(i);
 i++;
 }
 </script>

THE DO/WHILE LOOP

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax: do {
 code block to be executed
 }
 while (condition);

Example: <script type="text/javascript">
 var i=0;
 do{
 document.write(i);
 i++;
 }while (i<2)
 </script>

JAVASCRIPT BREAK & CONTINUE STATEMENT

THE BREAK STATEMENT

The break statement "jumps out" of a loop.

Example : <script type="text/javascript">
 for (i = 1; i < 10; i++) {


```
    if (i == 3)
    {
    break
    }
    document.write(i);
    }
</script>
```

Output: 12

THE BREAK STATEMENT

The continue statement "jumps over" one iteration in the loop.

Example :

```
<script type="text/javascript">
for (i = 1; i < 10; i++) {
    if (i == 3)
    {
    continue
    }
    document.write(i);
    }
</script>
```

Output: 12456789

JAVASCRIPT DIALOG BOXES

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

ALERT BOX

An alert box is used to display some information to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax : `window.alert("text");`

Example : `window.alert("This is Alert Box");`

CONFIRM BOX

A confirm box is used to verify or accept something. It will display "OK" or "Cancel" Options. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax : `window.confirm("text");`

Example :

```
var ans;
ans = window.confirm("This is Alert Box");
if(ans==true){
document.write("You pressed OK"); }
else{
document.write("You pressed CANCEL"); }
```

PROMPT BOX

A prompt box is used to input a value before entering a page. It will display "OK" or "Cancel" Options. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax : `window.prompt("text","defaultText");`

Example :

```
var ans;
ans = window.prompt("Enter Your Name","Name1");
document.write(ans);
```

SPECIAL CHARACTERS IN JAVASCRIPT

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Code	Outputs
<code>\'</code>	single quote
<code>\"</code>	double quote
<code>\&</code>	ampersand
<code>\\</code>	backslash
<code>\n</code>	new line
<code>\r</code>	carriage return
<code>\t</code>	tab
<code>\b</code>	backspace
<code>\f</code>	form feed

JAVASCRIPT ARRAYS

An array is a special variable, which can hold more than one value at a time. If you have a list of items, you can store it into a single variable using this

CREATING AN ARRAY

Syntax:

```
var array-name = [item1, item2, ...];
```

Example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var fruits = new Array("Banana", "Orange", "Apple", "Mango");  
var person = {firstName:"AAA", lastName:"BBB", age:26};
```

Access the Elements of an Array

You refer to an array element by referring to the index number. This statement access the value of the first element in myCars:

```
Ex.    var fruits = ["Banana", "Orange", "Apple", "Mango"];  
        var selected = fruits[0];  
        document.write(selected);  
Ex.    var person = {firstName:"AAA", lastName:"BBB", age:26};  
        document.write(person["lastName"]);
```

ADDING ELEMENTS INTO ARRAY

```
Ex.    var fruits = ["Banana", "Orange", "Apple", "Mango"];  
        fruits[fruits.length]="Pineapple";  
        document.write(fruits);  
O/p.   Banana,Orange,Apple,Mango,Pineapple
```

DELETING ELEMENTS INTO ARRAY

```
Ex.    var fruits = ["Banana", "Orange", "Apple", "Mango"];  
        delete fruits[3];  
        document.write(fruits);  
O/p.   Banana,Orange,Apple,
```

JOINING TWO ARRAYS

```
Ex.    var fruits = ["Banana", "Orange"];  
        var colors = ["Red", "White"];  
        var mix = fruits.concat(colors);  
        document.write(mix);  
O/p.   Banana,Orange,Red,White
```

JAVA SCRIPT USER DEFINE FUNCTIONS

A function is a block of code that will be executed when "someone" calls it. In JavaScript, we can define our own functions, called user-defined functions, or we can use built-in functions already defined in the JavaScript language.

JavaScript Function Syntax

A function is written as a code block (inside curly { } braces), preceded by the function keyword:

```
function functionname()
{
some code to be executed
}
```

e.g. :

```
<html>
  <head> </head>
  <body>

    <script type="text/javascript">
      function hello( )
      {
        document.write("Hello World!");
      }
    </script>

    <input type="button" name="print" value="Print" onClick="hello()">

  </body>
</html>
```

FUNCTION WITH PARAMETERS

When you call a function, you can pass along some values to it, these values are called parameters or arguments.

Parameters are the variables we specify when we define the function. When the function is later called somewhere else in the code, arguments are the values passed as parameters.

You can specify as many parameters as you like, separated by commas (.). The parameters then serve as variables that can be used inside the function.

```
function functionName( parameter1 , parameter2 , parameter3 )
{
some code to be executed
}
```

Ex.

```
<html>
  <head> </head>
  <body>
    <script type="text/javascript">
      function welcome(user)
```

```
        { document.write("Welcome " + user + " Good Morning"); }  
  
</script>  
  
<input type="button" name="print" value="Print" onClick=welcome("Students")>  
  
</body>  
  
</html>
```

FUNCTIONS WITH A RETURN VALUE

Sometimes you want your function to return a value back to where the call was made.

This is possible by using the **return** statement. When using the return statement, the function will stop executing, and return the specified value.

```
Ex. <html>  
  
<head> </head>  
  
<body>  
  
<script type="text/javascript">  
  
    function welcome(n1,n2,n3)  
  
    {      var avg=(n1+n2+n3)/3;  
  
          return avg;  
  
    }  
  
var returned = welcome(10,20,35);  
  
document.write(returned);  
  
</script>  
  
</body>  
  
</html>
```

EVENT HANDLERS

Event handlers are JavaScript code that executes when the user performs some action, or triggers an event.

The onclick Event

```
<h1> onclick="alert('you clicked me ...') " > Click on this text!</h1>
```

The onload and onunload Events

```
<body onload="checkCookies()">
```

The onchange Event

```
<input type="text" id="fname" onChange="makeUpper()" >
```

The onmouseover, onmouseout, onmousedown, onmouseup and onclick Events

BUILT IN FUNCTIONS**How to Use Built in Functions(A Sample Program)**

```

<html>
<head></head>
<body>
  <script type="text/javascript">
    var str = "COMPUTER";
    document.write(str.charAt(2));
  </script>
</body>
</html>

```

} **Function**

STRING FUNCTIONS**1. charat()**

The charAt() method returns the character at a specified index (position) in a string:

e.g. : `var str = "COMPUTER";`
`document.write(str.charAt(2));`
 o.p. : **M**

2. concat()

The concat() method can be used instead of the plus operator.

e.g. : `var str1 = "Learn ";`
`var str2 = "JavaScript";`
`document.write(str1.concat(str2));`
 o.p. : **Learn JavaScript**

3. indexOf()

Returns the position of the first found occurrence of a specified value in a string

e.g. : `var str = "Computers, Mobiles, Tablets";`
`document.write(str.indexOf("s"));`
 o.p. : **8**

4. lastIndexOf()

Returns the position of the Last found occurrence of a specified value in a string

e.g. : `var str = "Computers, Mobiles, Tablets";`
`document.write(str.lastIndexOf("s"));`
 o.p. : **26**

5. replace()

Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced

e.g. : `var str = "Computers, Mobiles, Tablets";`
`var str2 = str.replace("Mobiles","Cell Phones");`
`document.write(str2);`
 o.p. : **Computers, Cell Phones, Tablets**

6. search()

Searches a string for a specified value, or regular expression, and returns the position of the match.

e.g. : `var str = "Computers, Mobiles, Tablets";`
`document.write(str.search("m"));`
 o.p. : **2**

7. substr()

Extracts the characters from a string, beginning at a specified start position, and through the specified number of character (2nd argument is for length)

e.g. : `var str = "JavaScript";
document.write(str.substr(4,6));`

o.p. : **Script**

8. substring()

Extracts the characters from a string, between two specified indices (Till 2nd Option's Character)

e.g. : `var str = "JavaScript";
document.write(str.substring(4,6));`

o.p. : **Sc**

9. toLowerCase()

Converts a string to lowercase letters

e.g. : `var str = "JavaScript";
document.write(str.toLowerCase());`

o.p. : **javascript**

10. toUpperCase()

Converts a string to Uppercase letters

e.g. : `var str = "JavaScript";
document.write(str.toUpperCase());`

o.p. : **JAVASCRIPT**

MATH FUNCTIONS**1. abs()**

Returns the absolute value of variable(Positive Value)

e.g. : `var x = -2.15;
document.write(Math.abs(x));`

o.p. : **2.15**

2. ceil()

Returns the number rounded upwards to the nearest integer.

e.g. : `var x = 2.15;
document.write(Math.ceil(x));`

o.p. : **3**

3. floor()

Returns the number rounded downwards to the nearest integer.

e.g. : `var x = 2.15;
document.write(Math.floor(x));`

o.p. : **2**

4. pow(x,y)

Returns the value of x to the power of y

e.g. : `document.write(Math.pow(2,3));`
o.p. : **8**

5. random()

Returns a random number between 0 and 1

e.g. : `document.write(Math.random());`

o.p. : **0.5918758015614003 (ANY RANDOM VALUE LIKE THIS)**

6. round()

Rounds no to the nearest integer

e.g. : `var x = 2.15, y = 2.85;`
`document.write(Math.round(x));`
`document.write(Math.round(y));`

o.p. : **2 3**

7. max()

Returns the number with the highest value

e.g. : `document.write(Math.max(5,10,80,19));`

o.p. : **80**

8. min()

Returns the number with the lowest value

e.g. : `document.write(Math.min(5,10,80,19));`

o.p. : **5**

DATE FUNCTIONS

The below outputs are as per **01st November 2014 at 05:15 PM (Saturday)**

1. Date()

The Date object is used to work with dates and times. Date objects are created with `new Date()`.

e.g. : `var d = new Date();`
`document.write(d);`

o.p. : **Sat Nov 01 2014 17:06:06 GMT+0530 (India Standard Time)**

2. getDate()

Returns the day of the month (from 1-31)

e.g. : `var d = new Date();`
`var n = d.getDate();`
`document.write(n);`

o.p. : **1**

3. getDay()

Returns the day of the week (from 0-6)(Sunday = 0, Monday = 1, Tuesday = 2,...)

e.g. : `var d = new Date();`
`var n = d.getDay();`
`document.write(n);`

o.p. : **6**

4. getMonth()

Returns the month (from 0-11)

e.g. : `var d = new Date();`
`var n = d.getMonth();`
`document.write(n);`

o.p. : **10**

5. getYear()

Returns the Year

```
e.g. : var d = new Date();  
      var n = d.getYear();  
      document.write(n);
```

o.p. : **114**

getFullYear()

Returns the Year in Four Digits.

```
e.g. : var d = new Date();  
      var n = d.getFullYear();  
      document.write(n);
```

o.p. : **2014**

6. getHours()

Returns the hour (from 0-23)

```
e.g. : var d = new Date();  
      var n = d.getHours();  
      document.write(n);
```

o.p. : **17**

7. getMinutes()

Returns the Minutes(from 0-23)

```
e.g. : var d = new Date();  
      var n = d.getMinutes();  
      document.write(n);
```

o.p. : **15**

8. getSeconds()

Returns the seconds (from 0-59)

```
e.g. : var d = new Date();  
      var n = d.getSeconds();  
      document.write(n);
```

o.p. : **48**

9. getMilliseconds()

Returns the milliseconds (from 0-999)

```
e.g. : var d = new Date();  
      var n = d.getMilliseconds();  
      document.write(n);
```

o.p. : **845**

10. setDate()

Sets the day of the month of a date object

```
e.g. : var d = new Date();  
      d.setDate(18);  
      document.write(d);
```

o.p. : **Tue Nov 18 2014 17:21:54 GMT+0530 (India Standard Time)**

11. setMonth()

Sets the month of a date object

```
e.g. : var d = new Date();  
       d.setMonth(4);  
       document.write(d);
```

o.p. : **Thu May 01 2014 17:25:47 GMT+0530 (India Standard Time)**

12. setYear()

Sets the Year

```
e.g. : var d = new Date();  
       d.setYear(094);  
       document.write(d);
```

o.p. : **Tue Nov 01 1994 17:27:22 GMT+0530 (India Standard Time)**

13. setFullYear()

Sets the year (four digits) of a date object

```
e.g. : var d = new Date();  
       d.setFullYear(2000);  
       document.write(d);
```

o.p. : **Wed Nov 01 2000 17:28:22 GMT+0530 (India Standard Time)**

14. setHours()

Sets the hour of a date object

```
e.g. : var d = new Date();  
       d.setHours(2);  
       document.write(d);
```

o.p. : **Sat Nov 01 2014 02:29:41 GMT+0530 (India Standard Time)**

15. setMinutes()

Set the minutes of a date object

```
e.g. : var d = new Date();  
       d.setMinutes(01);  
       document.write(d);
```

o.p. : **Sat Nov 01 2014 17:01:30 GMT+0530 (India Standard Time)**

16. setSeconds()

Sets the seconds of a date object

```
e.g. : var d = new Date();  
       d.setSeconds(20);  
       document.write(d);
```

o.p. : **Sat Nov 01 2014 17:34:20 GMT+0530 (India Standard Time)**

ARRAY PROPERTIES AND METHODS

1. length()

The length() is used to find the length of array.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];  
       document.write(fruits.length);
```

o.p. **4**

2. sort()

The sort() is used to sort the array in ascending order.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.sort();
      document.write(fruits);
```

o.p. **Apple,Banana,Mango,Orange**

3. reverse()

The reverse() is used to sort the array in descending order.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.sort();
      fruits.reverse();
      document.write(fruits);
```

o.p. **Orange,Mango,Banana,Apple**

4. pop()

The pop() method removes the last element from an array

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.pop();
      document.write(fruits);
```

o.p. **Banana,Orange,Apple**

5. push()

The push() method adds a new element to an array (at the end):

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.push("Pineapple");
      document.write(fruits);
```

o.p. **Banana,Orange,Apple,Mango,Pineapple**

6. shift()

Shifting is equivalent to popping, working on the first element instead of the last.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.shift();
      document.write(fruits);
```

o.p. **Orange,Apple,Mango**

7. unshift()

Shifting is equivalent to popping, working on the first element instead of the last.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      fruits.unshift("Pineapple");
      document.write(fruits);
```

o.p. **Pineapple,Banana,Orange,Apple,Mango**

8. join()

The join() method joins the elements of an array into a string, and returns the string.

```
e.g. : var fruits = ["Banana", "Orange", "Apple", "Mango"];
      var all = fruits.join();
      document.write(all);
```

o.p. **Banana,Orange,Apple,Mango**

USER DEFINE OBJECT

HISTORY OBJECT**1. back()**

Loads the previous URL in the history list

e.g. : `<button onclick="goBack()">Go Back</button>`
`<script>`
`function goBack() {`
`window.history.back()`
`}`
`</script>`

2. forward()

Loads the next URL in the history list

e.g. : `<button onclick="goForward()">Go Forward</button>`
`<script>`
`function goForward() {`
`window.history.forward()`
`}`
`</script>`

3. go()

Loads the next URL in the history list

e.g. : `<button onclick="goBack()">Go Back 2 Pages</button>`
`<script>`
`function goBack() {`
`window.history.go(-2)`
`}`
`</script>`

NAVIGATOR OBJECT

The navigator object contains information about the browser.

1. appName()

Returns the name of the browser

e.g. : `var x = "Browser Name: " + navigator.appName;`
`document.write(x);`
o.p. : **Browser Name: Netscape**

2. appVersion()

Returns the version information of the browser

e.g. : `var x = "Version Info: " + navigator.appVersion;`
`document.write(x);`
o.p. : **Version info: 5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36**

3. cookieEnabled()

Determines whether cookies are enabled in the browser

e.g. : `var x = "Cookies Enabled: " + navigator.cookieEnabled;`
`document.write(x);`
o.p. : **Cookies Enabled: true**

4. platform()

Returns for which platform the browser is compiled

e.g. : `var x = "Platform: " + navigator.platform;`
`document.write(x);`

o.p. : **Platform: Win32**

5. **javaEnabled()**

Specifies whether or not the browser has Java enabled

e.g. : `var x = "Java Enabled: " + navigator.javaEnabled();`
`document.write(x);`

o.p. : **Java Enabled: true**

EVENTS

1. **onClick()**

The event occurs when the user clicks on an element

2. **ondblclick()**

The event occurs when the user double-clicks on an element

3. **onblur()**

The event occurs when an element loses focus

4. **onfocus()**

The event occurs when an element gets focus

5. **onChange()**

The event occurs when the content of a form element, the selection, or the checked state have changed (for `<input>`, `<keygen>`, `<select>`, and `<textarea>`)

6. **onkeypress()**

The event occurs when the user presses a key

7. **onkeydown()**

The event occurs when the user is pressing a key

8. **onkeyup()**

The event occurs when the user releases a key

9. **onmousemove()**

The event occurs when the pointer is moving while it is over an element

10. **onmouseup()**

The event occurs when a user releases a mouse button over an element

11. **onsubmit()**

The event occurs when a form is submitted

12. **onreset()**

The event occurs when a form is reset

13. **onselect()**

The event occurs after the user selects some text (for <input> and <textarea>)

14. onLoad()

The event occurs when an object has loaded

15. onUnload()

The event occurs once a page has unloaded (for <body>)

16. timer()

The event occurs when the user clicks on an element

CHAPTER 5 : MACROMEDIA DREAMWEAVER

GETTING STARTED WITH DREAMWEAVER MX

OPENING DREAMWEAVER MX

Go to the **Start** button and click **Programs**, another list will then appear with the programmers that there are installed in your computer, look for **Macromedia > Macromedia Dreamweaver MX 2004**, and the program will be opened.

If you primarily want to work in a design mode, select Designer; if you like to jump into the code right away, select Code. Whatever you select here can be easily changed later on. If you are new to Web design, you will probably find Designer a bit easier to work with initially.

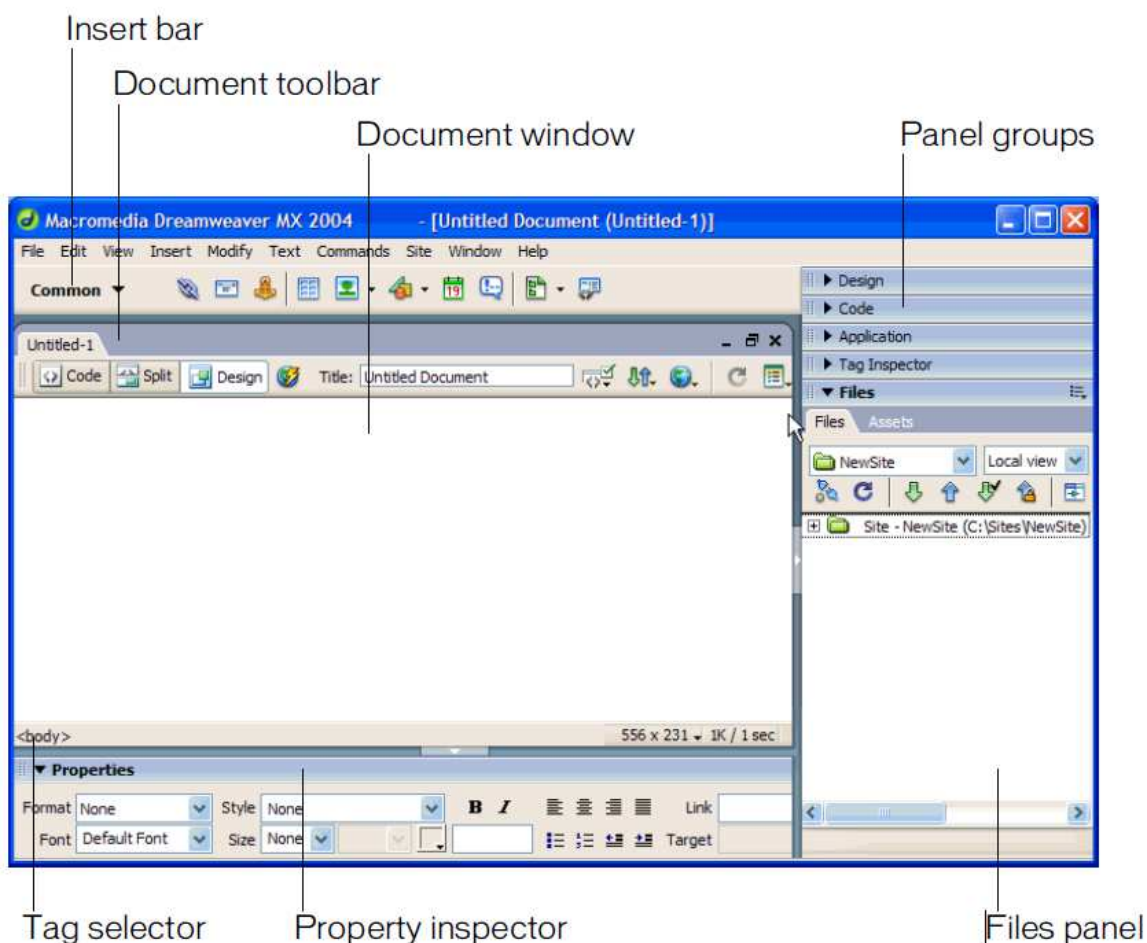
DIFFERENT VIEWERS IN DREAMWEAVER

There are three types of views in Dreamweaver – **DESIGN VIEW, CODE VIEW & SPLIT VIEW**. You can change the view of the document through the document toolbar.



- The **Design view** allows you to work with the visual editor.
- The **Code view** is used to be able to work totally in code programming mode.
- The **Split view** allows you to divide the window into two zones. The superior zone shows the source code, and inferior the visual editor. When a change is made in one of the zones, this change is applied directly on the other.

PROGRAM LAYOUT



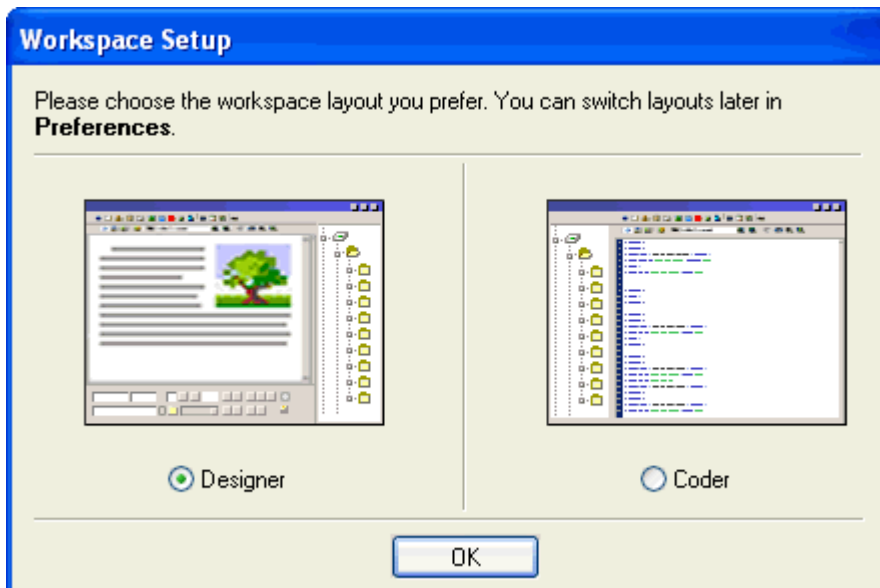
- The **Start page** enables you to open a recent document or create a new document. From the Start page you can also learn more about Dreamweaver by taking a product tour or a tutorial.

- The **Insert bar** contains buttons for inserting various types of “objects,” such as images, tables, and layers, into a document. Each object is a piece of HTML code that allows you to set various attributes as you insert it. For example, you can insert a table by clicking the Table button in the Insert bar. If you prefer, you can insert objects using the Insert menu instead of the Insert bar.
- The **Document toolbar** contains buttons and pop-up menus that provide different views of the Document window (such as Design view and Code view), various viewing options, and some common operations such as previewing in a browser.
- The **Document window** displays the current document as you create and edit it.
- **Panel groups** are sets of related panels grouped together under one heading. To expand a panel group, click the expander arrow at the left of the group’s name; to undock a panel group, drag the gripper at the left edge of the group’s title bar.
- The **Property inspector** lets you view and change a variety of properties for the selected object or text. Each kind of object has different properties.
- The **Files panel** enables you to manage your files and folders, whether they are part of a Dreamweaver site or on a remote server. The Files panel also enables you to access all the files on your local disk, much like Windows Explorer (Windows) or the Finder (Macintosh).

CHANGING WORKSPACE

To switch between the workspace

- 1 Select Edit > Preferences, then select General from the list of categories on the left.
2. Click the **Change Workspace** button.
3. Select the **Designer** or **Coder** option.
- 4 Click **OK** as necessary to close the Workspace Setup dialog box and the Preferences dialog box.
- 5 Close Dreamweaver and restart it.

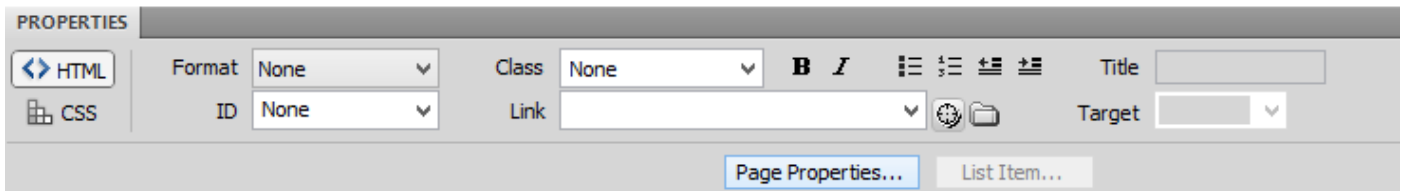


You can also change it by clicking
Window > Workspace

PANELS, MANAGING PANELS & INSERT BAR

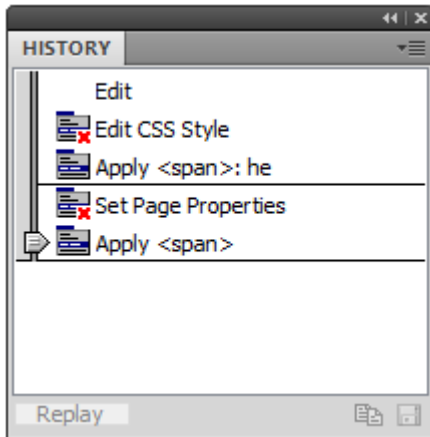
Properties Window/Panel

The Properties panel lets you adjust the properties of whatever page element you have selected. Most of the time it looks like this:



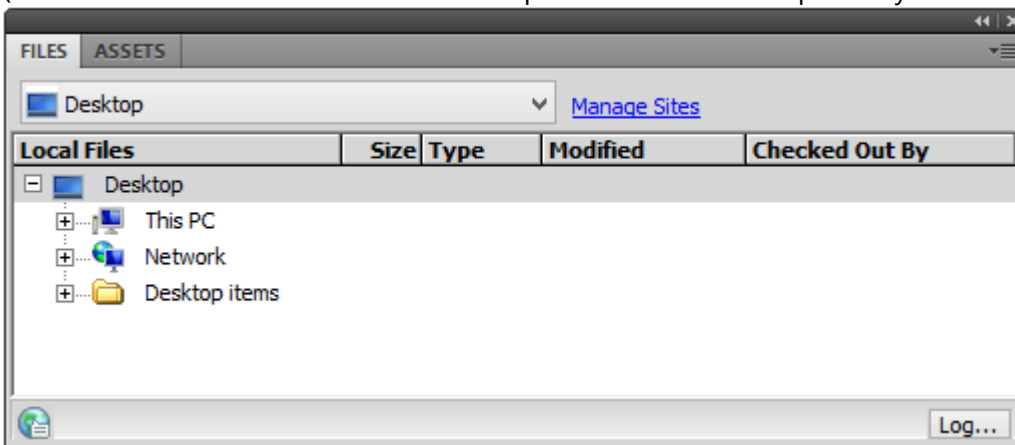
History Window/Panel

The History panel, in the list under 'Others', displays a list of all the actions you've taken in this screen in this session.



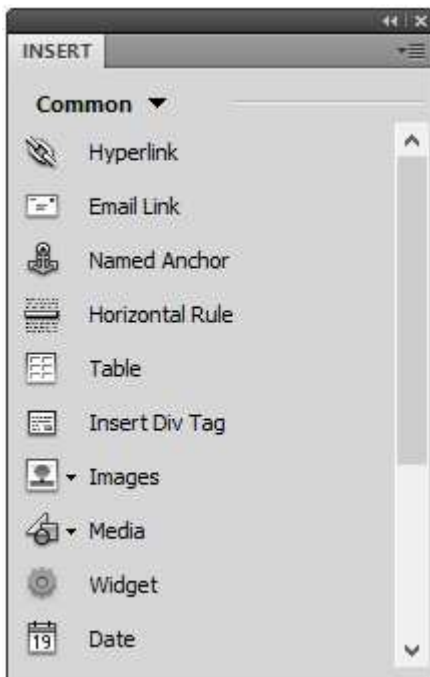
Files Panel

The Files panel generally appears in the right side of the window, and lists the files that are part of the site (if a site has been defined. Files can be opened from the Files panel by double clicking.)



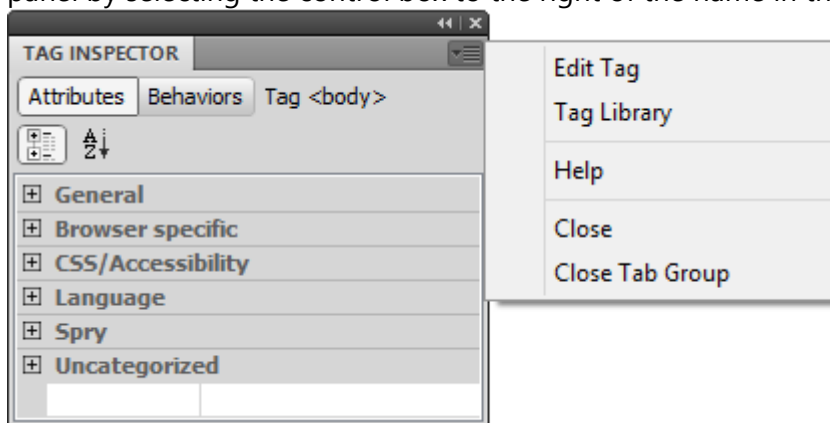
Insert panel

The Insert panel lets you insert various things, like links, images, tables, email links, dates, internal anchors, and other special objects, by clicking on them rather than choosing them from the Insert menu. You can choose one of eleven different object palettes from the drop-down menu at the top of the, depending on whether you are working with text, tables, special characters, forms, frames, media, script elements, etc.



MANAGING PANEL

You can close, regroup or rename panel groups to suit your demands. You may decide to customize your panel by selecting the control box to the right of the name in the panel group title bar.



You can collapse a panel group by clicking on the arrow mark to the left or right of the top corner.

To close a panel or window, either click the close box (marked x) in the upper right corner of the panel, or select it twice from the Windows menu, it will then be unchecked and the window will disappear.

To move a panel and dock/undock it you can do it by dragging it from Top Corner of the Panel.

MAKING A PAGE

WEB PAGES AND THEIR RELATION TO EACH OTHER

1. Open Dreamweaver
2. Create a new page.
3. Add content to a page.
4. Set the page title
5. Adjust the page properties
6. Save your work
7. Save a copy of your page
8. Preview the page in a browser
9. Close the file.

MULTIPLE PAGES WITH SIMILAR STYLE

1. From the Document window menu bar, Select File->New or Press Ctrl + N The new document dialog box will appear.

2. To create a regular HTML file, Click on Basic Page in the General category List and HTML in the basic Page list.
3. Click create. A new, Blank document will appear.
4. Then put Style in document.
5. After that, create new document again and put this style in that.

You can also create similar pages with similar styles by using the Cascading Style Sheet and to do that you just have to add the below line to your page and link a stylesheet file to your page.

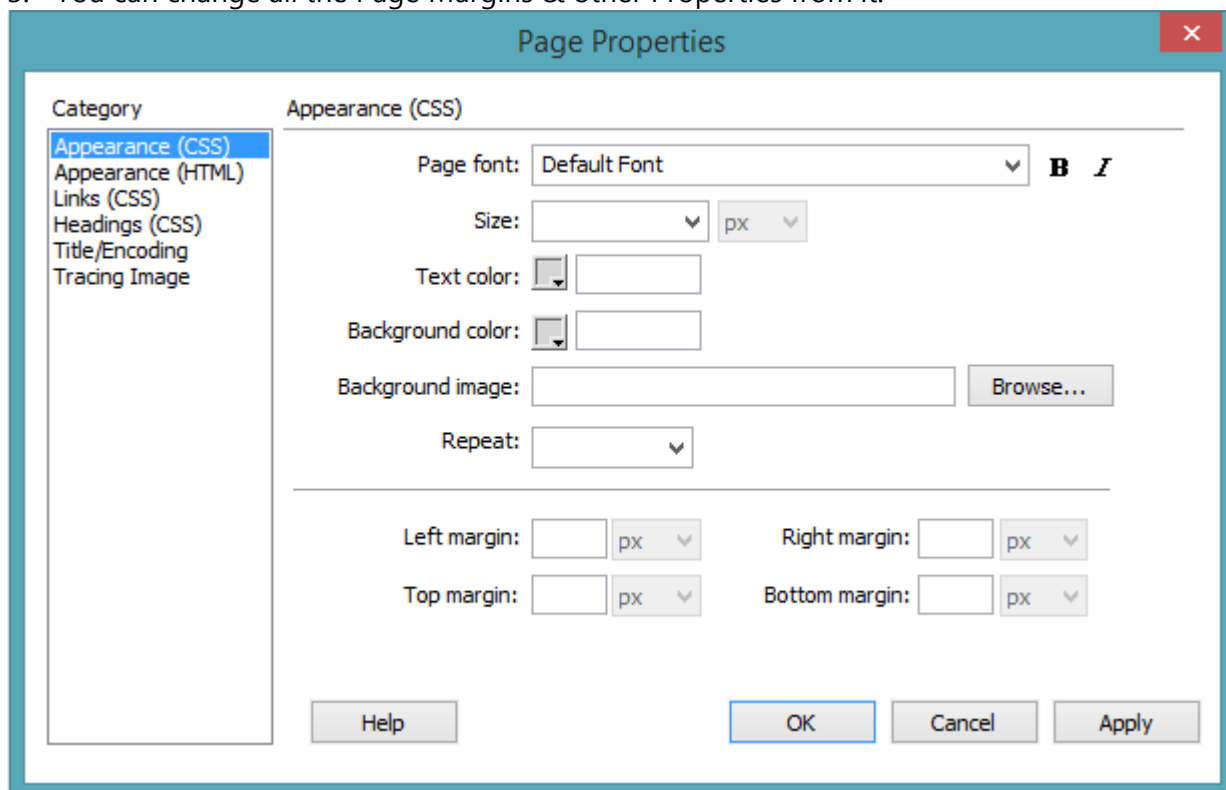
```
<link rel="stylesheet" href="site_styles.css">
```

PAGE PROPERTIES

Page properties are elements that apply to an entire page, rather than a single object on the page. Visual properties include the page's title, a background color or image and the text and link colors. Other page properties include the document encoding and the site folders, if any..

To set Page Margins, Colors, Background, etc.

1. Go to Properties bar(Window>Properties)
2. In the Properties bar look for Page Properties
3. You can change all the Page Margins & other Properties from it.



To Change the page title

1. In the Document toolbar, Click within the Title text box.
2. Type a new title and press Enter.
3. Type your page title in the Title text box.
4. Choose a good title for page, always try to write some descriptive title as some search engine uses it for indexing purposes.



TEXT AND TEXT PROPERTIES

You can type text into the Document window, or copy and paste it from another source (such as a Microsoft Word file). Then you can format the text using CSS styles.

You can use the text Property inspector to apply HTML formatting or Cascading Style Sheet (CSS) formatting. When you apply HTML formatting, Dreamweaver adds properties to the HTML code in the body

of your page. When you apply CSS formatting, Dreamweaver writes properties to the head of the document or to a separate style sheet.

1. Open the Property inspector (Window > Properties), if it isn't already open and click the CSS button.
2. Do one of the following:
 - Place the insertion point inside a block of text that's been formatted by a rule you want to edit. The rule appears in the Targeted Rule pop-up menu.
 - Select a rule from the Targeted Rule pop-up menu.
3. Make changes to the rule by using the various options in the CSS Property inspector.

Header Text

There are total 6 types of header text in the HTML i.e. h1 to h6. To define the text as a header you have to use Properties Panel(Window>Property), under the Format option you can select your text type.

Change Font Size and Font Face

To change font properties in Dreamweaver

1. Go to Properties bar(Window>Properties)
2. In the Properties bar look for Page Properties
3. You can change all the Properties from it.

Additional Formatting related to text in Dreamweaver(using CSS)

Targeted Rule Is the rule you are editing in the CSS Property inspector. When you have an existing style applied to text, the rule affecting the text's format appears when you click inside the text on the page. You can also use the Targeted Rule pop-up menu to create new CSS rules, new in-line styles, or apply existing classes to selected text. If you're creating a new rule, you'll need to complete the New CSS Rule dialog box. For more information, see the links at the end of this topic.

Edit Rule Opens the CSS Rule Definition dialog box for the targeted rule. If you select New CSS Rule from the Targeted Rule pop-up menu and click the Edit Rule button, Dreamweaver opens the New CSS Rule definition dialog box instead.

Font Changes the font of the targeted rule.

Size Sets the font size for the targeted rule.

Text Color Sets the selected color as the font color in the targeted rule. Select a web-safe color by clicking the color box, or enter a hexadecimal value (for example, #FF0000) in the adjacent text field.

Bold Adds the bold property to the targeted rule.

Italic Adds the italics property to the targeted rule.

Left, Center, and Right Align Adds the respective alignment properties to the targeted rule.

Note : Some options are available inside the Page Properties in newer version of Dreamweaver

LINKS

There are two types of links

1. Absolute paths

Absolute paths provide the complete URL of the linked document, including the protocol to use (usually http:// for web pages), for example, <http://www.adobe.com/support/dreamweaver/contents.html>.

2. Document-relative paths

Document-relative paths are usually best for local links in most websites. They're particularly useful when the current document and the linked document are in the same folder and are likely to remain together.

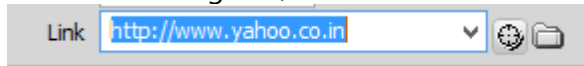
LINK PROPERTIES

Use the link tag to define a relationship between the current document and another file.

CREATING A LINK TO ANOTHER PAGE OR SITE

In Dreamweaver it's easy to create links to other websites, other pages in your own site, or to specific sections of a particular page. You can assign a hyperlink to text or images. In order to test the link, preview the page in a web browser – links aren't active in the Dreamweaver workspace.

Select the text or an image you want to be the link and enter the URL in the Link text box of the Property Inspector. You can type the URL directly in the box, browse for the file using the folder icon to open the Select File dialog box, or use the Point-to-File icon to drag to an existing site or page.



When you're linking to an external web page, you have to include its entire address. i.e.: <http://www.yahoo.co.in>. This kind of link is also referred to as an **absolute reference**.

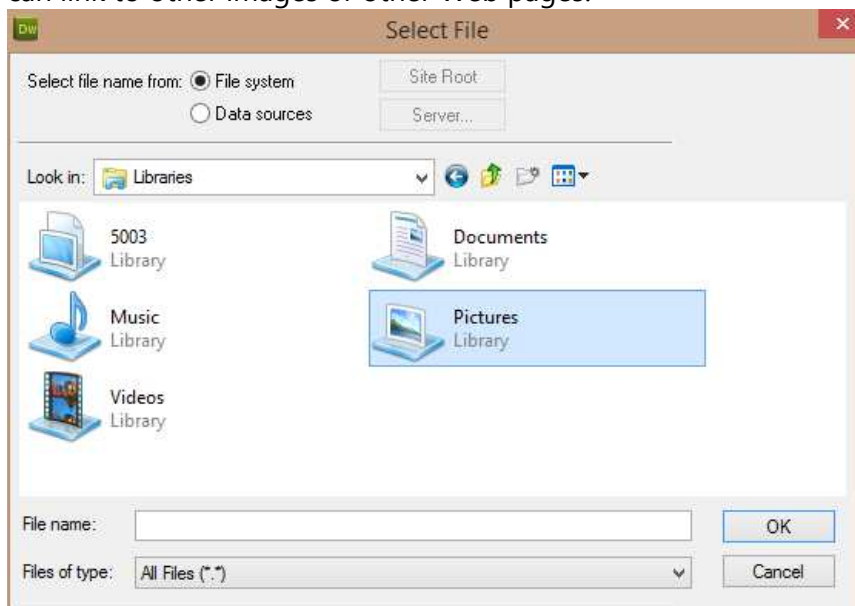
When you link to another page on your own site, you have to specify the entire URL. If all your web pages are in the same folder, you can just enter the name of the page you're linking to in the Link box. If you're linking to files stored in separate folders on your website, then the link should include the folder name, a slash, and the name of the page you want to open, i.e. "images/pic.jpg". This kind of link is also referred to as a **relative reference**.

MAKING AN IMAGE A LINK

In the properties Inspector (Ctrl+F3 if it's hidden) fill in the Link box with the URL you want to point to. You can either point to links on your current site or put full URLs into the box to point to pages on other sites. Click on any image on the page to get the properties to display. There will be black squares on the edges of the image to show that it is selected.

In the Properties Inspector for the image, there will be a Link box, just like for the text link. For this link, you should click on the yellow file icon to link to a page on your site.

A new window will open listing all the pages on your current site. Select the page you want to link to. You can link to other images or other Web pages.



1. Add an image to your Web page.
2. Click on the image to highlight it.
3. From the properties menu, click on one of three hotspot drawing to select rectangle, circle or polygon.
4. Draw the shape you want on your image.
5. In the properties window for the hotspot, type in or browse to the page hotspot should link to.
6. Continue adding hotspots until your map is complete
7. Click on the arrow icon to edit other properties of the image.
8. Preview your image map in a browser to make sure it works correctly.

LINKING TO OTHER MEDIA

1. Add a Text to your Web page
2. Click on the Insert Menu.

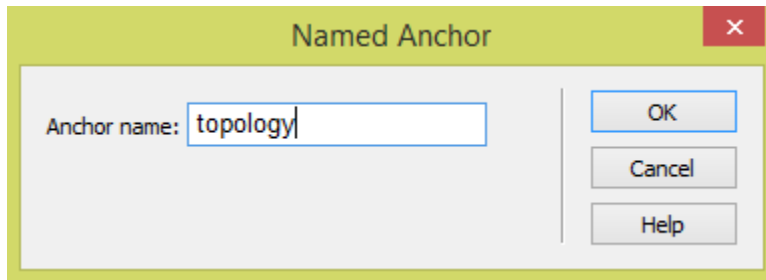
3. Select Media -> Flash tool.
4. Enter Text and Select Link for that.
5. Click Apply and Save Document.

MAKING ANCHORS

Named Anchors are essentially bookmarks within a page. You can create the anchor specific locations on the page (usually subheadings) and then you create links to anchors that allow users to jump to these places in the page.

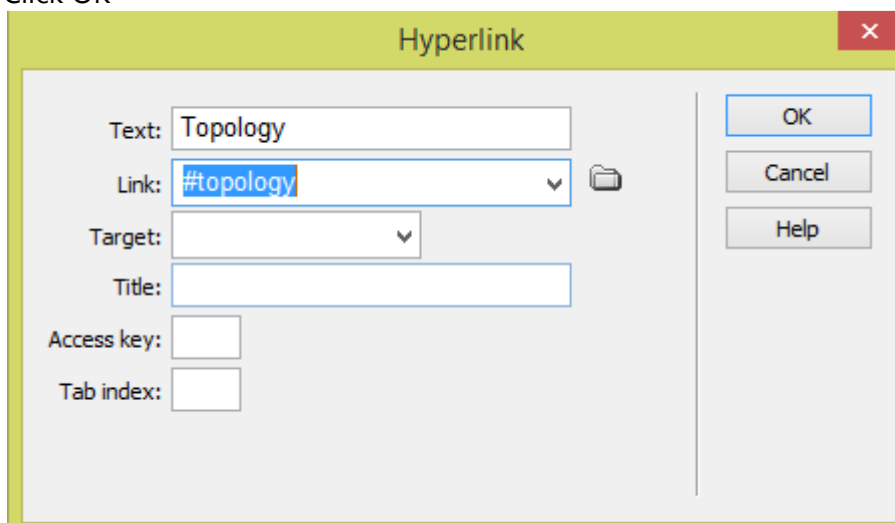
To create a named anchor.

1. Position the insertion point at the location where you want to create an often, this is at the beginning of the text of a subheading.
2. On the Insert bar, make sure Common is selected and then click the Named Anchor button.
3. Enter a name for the anchor and click OK



To Create a link to the anchor

1. With the cursor positioned where you want the link to appear, click the Hyperlink on the Insert bar.
2. In the Hyperlink dialog, enter the text for the link.
3. Select the named anchor from the Link menu
4. Click OK



PUBLISHING

MANAGING YOUR WORKSPACE

In Windows, Dreamweaver MX 2004 provides an all-in-one-window integrated workspace. In the integrated workspace, all windows and panels are integrated into a single larger application window. You can choose between a designer-oriented layout and a layout oriented toward the needs of hand-coders. Dreamweaver provides a floating workspace layout, in which each document is in its own individual window. Panel groups are initially docked together, but can be undocked into their own windows. Windows "snap" automatically to each other, to the sides of the screen, and to the Document window as you drag or resize them.

CREATING A NEW SITE

To define a site:

- 1 Start Dreamweaver.
- 2 Select Site > Manage Sites.The Manage Sites dialog box appears.

3 In the Manage Sites dialog box, click New, and select Site from the pop-up menu.

4 If the dialog box is showing the Advanced tab, click Basic. The first screen of the Site Definition Wizard appears, asking you to enter a name for your site.

5 In the text box, enter a name to identify the site within Dreamweaver. The name can be anything you want. For example, you could name the site Trio Motors.



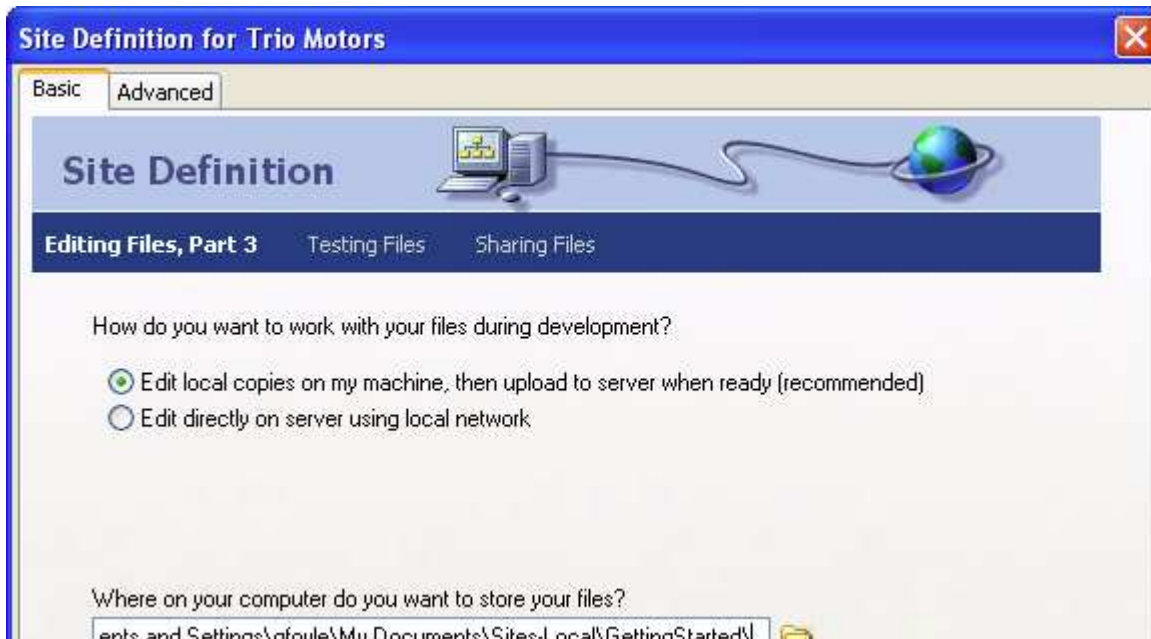
6 Click Next to proceed to the next step. The next screen of the wizard appears, asking if you want to work with a server technology.

7 Select the No option to indicate that for now, this site is a static site, with no dynamic pages. To set up a site to create a web application, you would need to select a dynamic document type—such as Macromedia ColdFusion, Microsoft Active Server Pages (ASP), Microsoft ASP.NET, Sun JavaServer Pages (JSP), or PHP: Hypertext Preprocessor (PHP)—and then supply information about your application server.



8 Click Next to proceed to the next step. The next screen of the wizard appears, asking how you want to work with your files.

9 Select the option labeled "Edit local copies on my machine, then upload to server when ready (recommended)." There are a variety of ways that you can work with files during site development, but for the purposes of this lesson, select this option.



10 Click the folder icon next to the text box. The text box allows you to specify a folder on your local disk where Dreamweaver should store the local version of the site's files, but it's easier to specify an accurate folder name if you browse to the folder rather than typing the path. The Choose Local Root Folder for Site dialog box appears.

11 In the Choose Local Root Folder for Site dialog box, start by navigating to the Sites-Local folder on your local disk, the folder you copied the sample files into in "Copy the sample files". Select the GettingStarted folder inside the Sites-Local folder. Open the GettingStarted folder, and click Select (Windows)

12 Click Next to proceed to the next step. The next screen of the wizard appears, asking how you connect to your remote server.

13 For now, select None from the pop-up menu.

You can set up information about your remote site later; for now, the local site information is all you need to start creating a page.

14 Click Next to proceed to the next step.

The next screen of the wizard appears, showing a summary of your settings.

15 Click Done to finish. The Manage Sites dialog box appears, showing your new site.

16 Click Done to close the Manage Sites dialog box.

After you create a website, next step is to publish it by uploading the files to a remote web server.

1 In your remote site (on the server), create an empty folder inside the web root folder for the server. Name this new empty folder with the same name as your local root folder; for example, for the tutorial site, you might name the remote empty folder GettingStarted to match the name of the local root folder.

2 In Dreamweaver, select Site > Manage Sites.

3 Select a site (such as Trio Motors) and click Edit.

4 Click the Basic tab at the top of the dialog box.

5 You've already filled in the first few steps in the Basic tab, when you set up your local site, so click Next a few times, until the Sharing Files step is highlighted at the top of the wizard.

6 In the pop-up menu labeled "How do you connect to your remote server?" select a method for connecting to the remote site. The most common methods for connecting to a server on the Internet are FTP and SFTP; the most common method for connecting a server on your intranet, or to your local computer if you're using that as a web server, is Local/Network. If you aren't sure what to select here, ask the server's system administrator.

7 If you selected FTP, enter the following options:

- Enter the hostname of the server (such as ftp.macromedia.com).

- In the text box that asks what folder contains your files, enter the path on the server from the FTP root folder to the remote site's root folder. If you're not sure, consult your system administrator. In many cases, this text box should be left blank.
- Enter your user name and password in the appropriate text boxes.
- If your server supports SFTP, select the Use Secure FTP (SFTP) option.
- Click Test Connection.
- If the connection is unsuccessful, consult your system administrator.

8 If you selected Local/Network, click the folder icon next to the text box and browse to the remote site's root folder. You may want to deselect the Refresh Remote File List Automatically option for improved speed.

9 After you've entered the appropriate information, click Next.

10 Don't enable file check-in and check-out for the Trio Motors site. If you and your co-workers are working together on a larger site, the file check-in and checkout feature helps to prevent you from overwriting each others' files. Also, if you or your coworkers use Macromedia Contribute, you must enable file check-in and check-out. For the Trio Motors sample site, though, you don't need this feature.

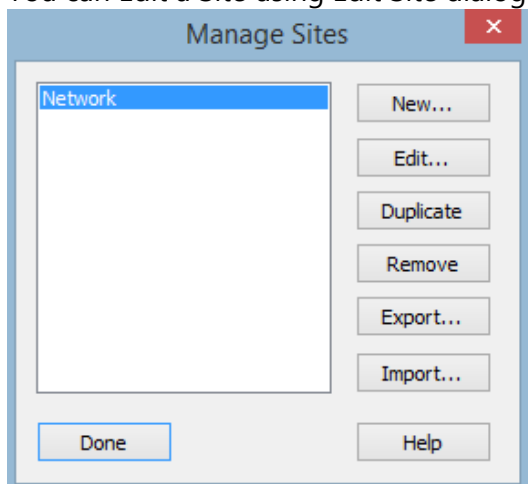
11 Click Next.

12 Click Done to finish setting up the remote site.

13 Click Done again to dismiss the Manage Sites dialog box.

EDITING A SITE

You can Edit a Site using Edit Site dialog box from Site Menu by selecting the appropriate site name.



TEMPLATE

CREATING A NEW TEMPLATE

Method 1

1. File > Save as Template.

Method 2

1. File > New > Blank Page > HTML Template
2. Save it.

Method 3

1. File > New > Blank Template or Template Page > HTML Template
2. Save it.

UNEDITABLE AND EDITABLE REGIONS

To insert Editable Regions

1. Select the text or content that you want to set as an editable region.
2. Place the insertion point where you want to insert an editable region.

3. Do one of the following to insert an editable region:
4. Select Insert > Template Objects > Editable Region.
5. Right-click (Windows), then select Templates > New Editable Region.
6. In the Common category of the Insert panel, click the Templates button, then select Editable Region from the pop-up menu.
7. In the Name box, enter a unique name for the region. (You cannot use the same name for more than one editable region in a particular template.)
8. Click OK. The editable region is enclosed in a highlighted rectangular outline in the template, using the highlighting color that is set in preferences.

To Change Editable Regions Content

1. Open the existing template from your site panel or Assets panel.
2. Make changes to areas that are not editable regions.
3. Save the Page using CTRL + S or File>Save.
4. Update associated pages.
5. Close the template instance Using CTRL + W or File>Close.

SAVING YOUR TEMPLATE

1. Open the template file that you just created, you may use any convenient method to do this.
2. Give the template a title ({sitename} Template).
3. Add your non-editable content (Site structural element, graphics and navigational elements).
4. Add your template region(s)
5. Save the Page using CTRL + S or File>Save.
6. Close the template instance Using CTRL + W or File>Close.

CREATING A NEW PAGE FROM A TEMPLATE

1. Go to File New to create a New from template file using the newly created templates.
2. Select your site in the Template for: column
3. Select your template from the site "{sitename}": column.
4. Make sure the Update pages when template changes is checked.
5. Click Create button & Immediately save the page in your site using CTRL + S or Save as.

CHANGES TO A TEMPLATE

To change the template that a page uses, or apply a template to an existing page, to Modify menu, choose templates then apply template to page. Choose template to apply.

If the region names in the new template don't match the region names in the template or the page didn't previously use a template- you will be asked what regions to put the content in.