INTERNET TECHNOLOGIES AND WEB DESIGNING

Author:
D.G.Ramnath

Edited by:
Dr. P.Premchand

Department of CSE,
UCOE,
Osmania University

To My Son

D.KIRTHI KUMAR

Contents:

1. Introduction

1.1 What is the Internet?

The Internet is a loose interconnection of thousands of computer networks reaching millions of people all over the world. Although its original purpose was to provide researchers with access to expensive hardware resources, the Internet has demonstrated such speed and effectiveness as a communications medium that it has gone beyond the original mission. The original mission was actually meant for scientists to share information among themselves. It has, in recent years, grown so large and powerful that it is now an information and communication tool that anybody cannot afford to ignore.

1.2 When it came?

The Internet Ocean was created in 1969 with the birth of ARPANET, an experimental project of the U.S. Department of Defense Advanced Research Projects Agency (DARPA). It had a simple vision, to explore experimental networking technologies that would link researchers with remote resources such as large computer systems and databases. The success of ARPANET helped cultivate numerous other networking initiatives, which grew up intertwined; 25 years later, these have evolved into an ever expanding, complex organism comprising tens of millions of people and tens of thousands of networks.

Most users describe the Internet (or the "Net") as a "network of networks"; it appears to stretch forever. It doesn't just connect you and another computer; it connects you and all other Internet-connected computers. Don't think of the Internet as just a bunch of computers, though. It is a perpetually expanding universe with its own geography, "weather," and dynamic cultures. In this cyber-sphere, people in geographically distant lands communicate across time zones without ever seeing each other, and information is available 24 hours a day from thousands of places.

As a result, the Internet is overflowing with useful resources and with people climbing aboard.

1.3 How large is the Internet?

According to the Internet Society (ISOC), a professional organization of Internet developers, influencers, and users, as of in early 2005, the Internet reached more than 100 countries consisting of 1 million networks and 325 million computers. Now the figures are even more!

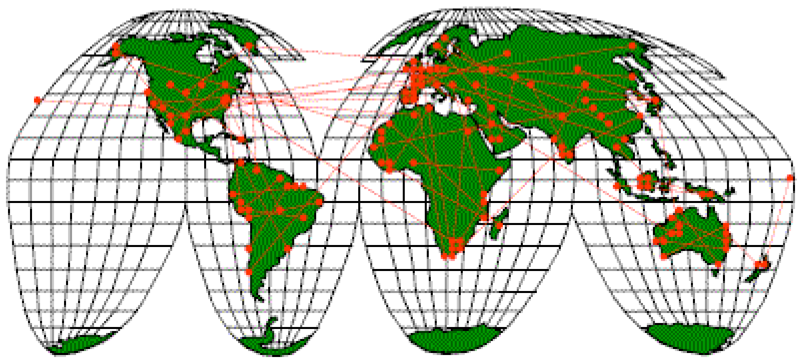


Fig.1.3.1 Worldwide distribution of net

1.4 To whom it belongs?

The bulk of Internet computers and networks still belong to the research and education communities. This is not surprising, because the Internet arose from a physics laboratory. However, many universities are teaming up with businesses to develop online catalogs and archives. And, according to a journal survey, 31 percent of the networks belong to businesses. Of the number of registered—but not necessarily connected—networks, 51 percent were commercial. There's definitely a rising trend in commercial activity. Many businesses have realized that they can link their enterprise networks to the Internet and gain instant access to their customers. Some market research indicates that online services in general make up almost a billion-dollar industry, with an estimated 25 percent per year growth.

The types of resources accessible via the Internet are growing at an astounding rate. The term resource describes anything you can access on the Internet, no matter where it's physically located.

Examples of some Internet resources are:

• An online newspaper	www.thehindu.com
• Weather information	www.weather.com
• Radio	www.broadcast.com
• Online Book store	www.amazon.com
• Music clips	www.mp3.com
• Search by topic	www.google.com
• Exam results	www.ouresults.com

…….. and many more

1.5 Growth of the Internet

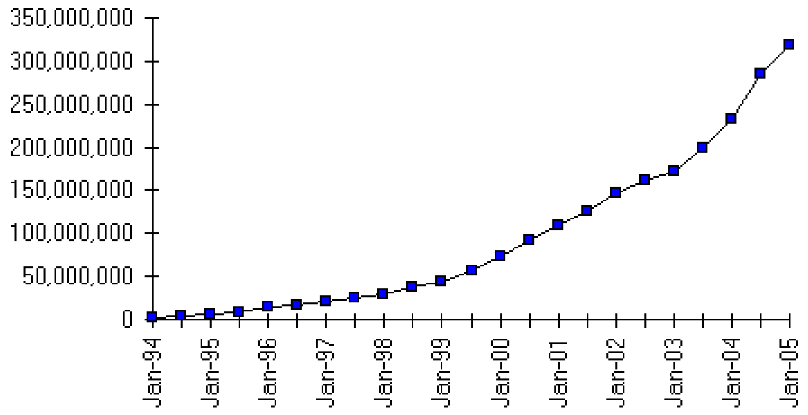


Fig.1.5.1. Growth of Internet

It's hard to imagine how the Internet has grown so fast and been so successful without some ambitious organization or individual managing the project. Yet no one has a monopoly on access to or use of the Internet; there's no monolithic empire called Internet, Inc., controlling accounts and application development or roping off the backstage parts of cyberspace.

One of the reasons for the Internet is so successful is the commitment of its developers to make Internet technology an "open system". That is virtually any machine can access the Internet. It is a machine independent technology. Standards that the Internet uses are known as the TCP/IP protocol suite.

Standards play a big part in everyday life. Camera film always fits in your camera, and loose paper bought at the stationary shop fits in your folder. Libraries catalog books according to a universal standard system, so that once you learn it, you can walk into any library in the world and find the books you need.

On the contrary, things that don't conform to standards can make your life miserable. Standards are just as important in the computer and networking world. Without open standards, only computers from the same vendor could talk to one another. Computers and networks that conform to the same communications standards are able to converse with each other regardless of brand.

The Internet system does not have a central system. Rather, all the networks and computers act as one-to-one in the exchange of information and communication among themselves. The technology that makes it happen is known as “internetworking”; it creates universality among heterogeneous systems, enabling the networks and computers to communicate. It is an example of distributed system.

Fundamentally, the exchange of information in a network revolves around the concept of a packet, a basic building block or a digital brick. All information and communication transmitted on the Internet are broken into packets, each of which is considered an independent entity. The packets are then individually routed from network to network until they reach their destination, where they are reassembled and presented to the user or computer process.

This method of networking is very flexible and robust. It allows diverse computers and systems to communicate by means of networking software, not by hardware. If a network goes "down"—meaning it isn't available to transfer information—the packets can be rerouted to other networks in many cases. This dynamic alternate routing of information creates a very reliable means of communication. Indeed, that was the intention of the network engineers developing this technology. They wanted a network that would continue to function even if parts of it were damaged.

While most beginners probably don't care about these standards and technical details, an understanding of the underlying infrastructure will help in learning to use the Internet properly and in taking full advantage of its powerful capabilities. It is very fascinating how separate computers and networks fit together will give the net culture—the sharing, cooperative spirit that is inherent in the Internet.

Open standards enable businesses and individuals to compete on a level playing field in developing networking software and products. But "open networking" extends beyond the development of networking protocols and products. Once you, an Internet user, you have access to the same resources as the rest of the millions of Internet users, wherever you are located.

The phrase “Right to know” often comes up in discussions about the Internet, which is, indeed, a truly democratic forum. The network doesn't care if you're a crorepathi or a lakhpathi, a small farmer or a big scientist - your requests and comments are handled in the same way, not on your social or financial status!

It's also never been so easy to be both a consumer and a producer of services. If you're ambitious enough and aspire to be an entrepreneur who provides commercial services or Internet access, there's nothing to prevent you (of course you have to fulfill some formalities). Once your network is directly hooked into the Internet, all the computers on that network are accessible from every other Internet-connected computer.

This environment empowers the individual; it encourages and stimulates participation, imagination, and innovation. There are many successful Indian names are also there who have leveraged the Internet to do great heights.

If you do not have access to a high-speed Internet connection or to a large multi-user computer, that's not a problem. Already there are people offering rental space on their Internet-connected computers and disks. You can lease "office space" from "office parks" in cyberspace and set up shop. Your virtual storefront may be thousands of miles and two countries away, but it's probably a few seconds hyper drive from every location. Such a convenience is a given on the Internet.

The Internet has had a positive effect on the democratization of society, making it more difficult to bottle up and suppress information. Network support of global freedom of information contributes to freedom of thought and to social and political freedom.

The advantages of Internet in a nutshell:

Communication tool: Data, voice, video can be instantly exchanged- thanks to high-speed networks
Teaching aid: Many complex concepts were easily explained using graphics. Wealth of knowledge.
Business tool: E-commerce has totally changed the way we were doing business
Information tool: Ocean full of information on any particular topic- thanks to the powerful search engines.
E-governance: Many government organizations already started using for collection of revenue, tax etc; Imagine earlier- even to pay electricity bill it was total one day's job sometimes!
Travel: Internet is useful for advance reservation, tour planning etc..
Medical & health: A specialist doctor's services can be availed over net

The possibilities with the Internet are endless.

Some believe that it is only a matter of time before the Internet displaces the telephone and television as the preeminent media for long distance communication.

With all goodies about the Internet there is a need for regulation of the Internet for a number of reasons:

- To provide common standards and security for the global use of community.
- To provide protection (copyright) for knowledge providers.
- To balance freedom of speech against offensive and libelous use.
- To encourage educational and social beneficial uses.
- To protect against anti-social uses by criminals and by terrorists.
- To protect countries, languages and cultures, ensuring access for rich and poor individuals and countries.

The Internet in its current state has certain shortcomings also including:

- Insecure transactions though secure protocols are being considered

- Protocols that provide minimal or non-existent guarantees of service quality.
- No mechanisms for protecting the intellectual property
- No support for interpolation or data interchange standard.

1.6 Safeguards

As you use the Internet, sometimes you may run into problems. Popup windows may appear on your screen. Strange email messages might come asking for bank information. You wonder if it is OK to order things with your credit card. You might begin wonder if it is safe to use the Internet at all.

The truth is that it is not safe to use the Internet if you are not aware of potential problems, don't take preventative measures, don't use caution, and don't take responsibility for your actions.

The following are few tips:

Protect yourself from security issues--keep your computer's operating system updated. For example, if you are using Microsoft Windows, visit windowsupdate.microsoft.com and follow the instructions. You can also keep the automatic update feature of your computer turned on so that security updates are downloaded automatically.

Install or use Spam protection. Visit the Web site of your Internet Service Provider (the one who gives you Internet connection) and look for information they have about preventing spam and viruses from reaching your email.

Enable popup blocking features of your Web browser (see the Help section of your Web browser) or install popup blocking software. Popups can be annoying and can also lure you into scams or other problems. Your Internet Service provider may also have special assistance or software for you to use to block popup windows.

Enable cookie protection in your Web browser. (Cookie is a piece of information that is automatically conversed with webserver whenever some one visits a web site. Some times it may carry very sensitive information about your system!) You shouldn't be afraid of cookies--they are useful for many ways but consider accepting cookies from only known, trusted Web sites.

Protect yourself from potential loss of data and create backup copies of your data regularly. If your computer crashes or if there is a fire or flood in your home, you don't want to lose important personal information. Copy information you don't want to lose onto a disk and store the disk in a secure place away from your computer. Be sure to password protect the disk if it contains sensitive information or keep the disk in a safe place.

Summary:

The Internet is a loose interconnection of thousands of computer networks reaching millions of people all over the world. Although its original purpose was to provide researchers with access to expensive hardware resources, the Internet has demonstrated such speed and effectiveness as a communications medium that it has gone beyond the original mission. The Internet has demonstrated such speed and effectiveness as a communications medium that it has gone beyond the original mission.

With all goodies about the Internet there is also a need for regulation of the Internet regarding security and protection.

The shortcomings of Internet include no mechanism for intellectual property protection, no data interchange standard.

Short questions:

1). How big is the Internet.
2). For What reason initially the Internet was invented?
3). In which year the Internet was evolved?
4). What is ARPANET?
5). List atleast three resourses of net.
6). On which protocol Internet runs?
7). Who provides the Internet?
8). List atleast two major organizations who contributed for the Internet.
9). What is a packet?
10). What are popup messages?

Long questions:

1). Describe the growth of Internet chronologically
2). Discuss the functioning of Internet
3). State the advantages of Internet
4). Why there is a need for regulation of Internet?
5). What are cookies? Why they are dangerous?
6). What do you mean by open system?
7). Describe the role of the net in a common man’s life
8). What are the setbacks of Internet?
9). What are the safeguards to be observed while on net?
10). What are the shortcomings of Internet and how to overcome?

2. Internet Connectivity

2.1 Client server system

The Internet is based on client server model. To accomplish a given task there must be a client server pair. There are many ways in which the client server model can operate. The most common being socket interface.

Before the advent of networking concepts, mainframe computers extended their power to individual computers through terminal connections and they were sharing the mainframe on time slot basis. A typical multi-user operating system on the mainframe was taking care of all this. Those were days when all the computer power in one central location called mainframe that catered the needs of every other through dumb terminal. There wasn't any computing power in the terminal systems except displaying the output.

Eventually, however, some of the hardware personnel discovered that personal computers (PCs) provided more power at substantially less cost than the mainframe, and the actual computing was shifted down to the user terminal. The powerful mainframe computer was left to do nothing but supply data (and sometimes program) files to the PCs. Thus earlier dumb terminals have become intelligent terminals. PCs that could manage the shared resources required by the system replaced the costly mainframe. Because the special PC served the needs of the other PCs, it was called a **SERVER**. The corresponding term for the desktop PC is client. This form of network link is, consequently, called a client-server model. The server in a client-server network runs special software (the network operating system NOS).

2.1.1 Servers should be powerful and reliable

Typically, the server is a special PC more powerful than the rest in the network. The most important feature of a server is storage. Because its file space is shared by many perhaps hundreds of PCs, it requires huge amounts of mass storage. In addition, the server is designed to be more reliable because all the PCs in the network depend on its proper functioning. If it fails, the entire network suffers.

Servers are designed to be fault-tolerant. That is, they will continue to run without interruption despite a fault, such as the failure of a hardware subsystem. Most servers use the special powerful microprocessors, not from need but because the price difference is tiny once the additional ruggedness and storage are factored in—and because most hardware personnel think that the single most important PC in a network should be the most powerful in all aspects.

There are several ways that a computer can ask for the services of another computer. A computer runs a program to either request a service from another computer or provide a service to another computer. This means the two computers, connected by an Internet, must each run a program, one to provide a service and the other to request a service. Thus if we want to use the services available on the Internet, application programs, running at both end computers and communicating with each other are needed. In an Internet it is the application programs that communicate with each other, not computers or users.

2.1.2 Intricacies involved

Enabling communications between two application programs, one running at local site and other running at remote site seem simple but there are many intricacies involved:

Should both applications provide services and request services?
Normally client program running on local machines request services, server program at the remote machine provides the service.

- Should application program provide services only to specific clients or all?
  Normally server provides service for any client not a particular client
- When should a client program run and when should a server program run?
  Generally the server program, which provides service, should run all the time because it doesn't know when its services will be requested.
- Should there be one application program for each type of service or one universal application program that can cater to different requests?
  In the Internet there are different types of services. All standardized services will have different application programs where as services which are customized should have one generic application program.

2.1.3 Active close and active open

A client program is initiated by the user (or it can be another application program) and terminates when the service is complete. A client opens the communication channel using the IP address of the remote host and the well-known port address of the specific server program running on the machine. Thus the channel becomes **active open**. After becoming active open client requests and subsequently receives the needed response from the server. The request-response may be repeated several times but still it is called a finite process and eventually comes to an end. At that instant the client closes the communication channel becomes **active close**.

A server program on the other hand is an infinite program. When it starts, it runs infinitely unless a problem arises. It waits for incoming requests from clients, when a request arrives, it responds to the request in two different ways namely iteratively or concurrently.

Running clients iteratively means running them one by one. One client must start, run, and terminate before the machine can start another client. Most computers today allow concurrent clients i.e. two or more clients can run at the same time.

The server can use connectionless or connection-oriented services. An iterative server normally uses UDP connectionless service. The server uses one single port (generally a well-known port). All the packets arriving at this port wait in line to be served.

In connection oriented concurrent server TCP is used. In this situation many ports are needed but a server can use only one well-known port and several ephemeral port. A client can make its initial approach to this port via well-known port. Once a connection is established, the data transfer takes place via ephemeral port. The well-known port is always reserved for making a new connection with the client.

2.1.4 Sockets

When two processes communicate with each other, they need a socket at both ends. A socket is nothing but a special type of structure. A typical example has five fields:

- **<u>Family</u>:** This field defines the protocol group IP4, IP6, UNIX protocols etc.
- **<u>Type</u>:** This field defines the type of socket. There are three types of sockets. One is stream socket other is pocket socket and a raw socket.
- **<u>Protocol</u>:** This field is set to zero for TCP and UDP.
- **<u>Local Socket Address:</u>** It defines a combination of IP address and port address.
- **<u>Remote Socket Address:</u>** This is combination of remote IP address and port address of remote application.

2.2 Internet domains and addresses name system

The Domain Name Service (DNS) is a network service that translates domain names to IP addresses. It helps Internet users to use the net more easily by allowing them to specify meaningful names to web sites and/or other users they want to communicate with.

When computers talk to each other via the Internet, they use the IP protocol. The **Internet Protocol (IP)** distinguish hosts from each other by an IP address which is a string of numbers appended to each other and separated by periods. An example of such a string might be 197.15.3.2.

The IP addresses are unique, which means that each host has its own IP address, which is different from all other IP addresses in the world. However, these IP address numbers are hard to remember and we prefer to use meaningful names instead, which we are used to in the everyday life.

The DNS is needed by applications to convert humans' meaningful names into computer meaningful names (IP addresses) and provide the final user an easier way to communicate via the Internet. Humans prefer names while computer prefer numbers. Each computer name consists of a sequence of alphanumeric segments separated by periods. For example, a computer name might be:

<u>www.yahoo.com</u>

A computer name is also called a 'Domain Name'. Domain names are hierarchical, with the most significant part of the name on the right. The left-most segment of a name (www in the example) is the name of an individual computer. Other segments of the full name identify the group that owns the individual name. In the above example, the individual name World Wide Web is owned by the yahoo group of names, which is itself owned by the com group (which holds a lot of company names).

The Domain Name System does not specify the exact number of segments in a name. Each organization can choose its own names hierarchy and specify as many segments as it needs. However, all names should be under familiar top-level domains.

www.osmania.ac.in

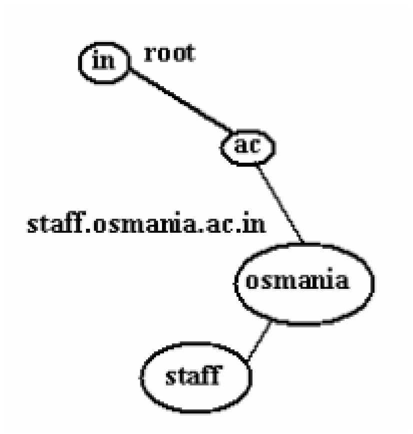


Fig 2.2.1 A hierarchical illustration of the example

2.3 TCP/IP Internet Domain Names

The mechanism that implements a machine name hierarchy for TCP/IP Internet is called the Domain Name System (DNS).

DNS has two independent aspects:

- It specifies the name syntax and rules for delegating authority over names
- It specifies the implementation of a distributed computing system that efficiently maps names to addresses.

The domain name system uses a hierarchical naming scheme known as domain names. A domain name consists of a sequence of sub-names separated by a delimiter character, the period. The domain calls each section a label.

The domain name mail.yahoo.com

contains three labels: mail, yahoo, and com. Any suffix of a label in a domain name is also called a domain. In the above example the lowest level domain is yahoo.com, the second level domain is mail.yahoo.com and the top level domain is com.

Writing domain names with the local label first and the top domain last makes it possible to compress messages that contain multiple domain names.

2.3.1 Official And Unofficial Internet Domain Names

Most users of the domain technology follow the hierarchical labels used by the official Internet domain system. There are two reasons:

The Internet scheme can accommodate a wide variety of organizations, and allows each group to choose between geographical or organizational naming hierarchies.

Most sites follow the Internet scheme so they can attach their TCP/IP installations to the global Internet without changing names.
The Internet authority has chosen to partition its top level into the following domains:

Domain Name	Meaning
.COM	Commercial organizations
.EDU	Educational institutions
.GOV	Government institutions
.MIL	Military groups
.NET	Major network support centers
.ORG	Organizations other than those above
.ARPA	Temporary ARPANET domain
.INT	International organizations
Country code	Each country (.in for India, .au for Australia etc)

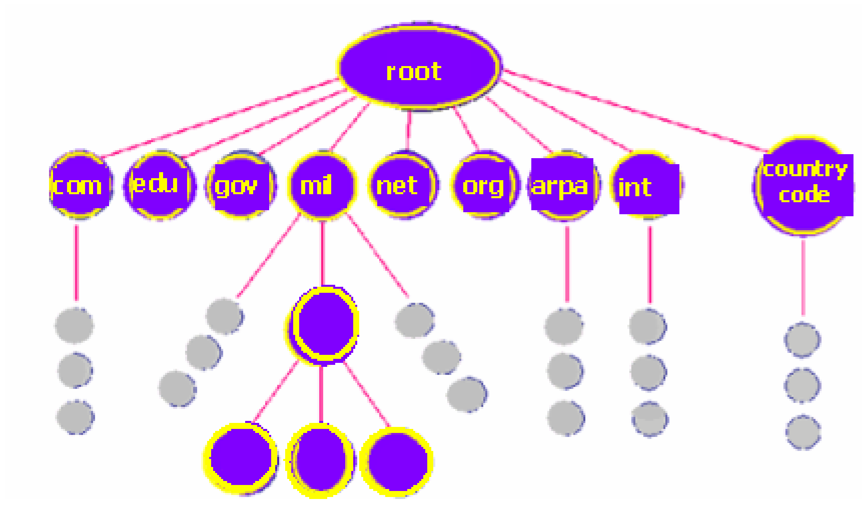


Fig.2.3.1 The DNS top level hierarchy:

The top-level names permit two completely different naming hierarchies: geographic and organizational. The geographic scheme divides the universe of machines by country. Each country is assigned its own top-level domain with the country's international 2-letter identifier as its label.

As an alternative to the geographic hierarchy, the top-level domain also allows organizations to be grouped by organizational type. An organization chooses how it wishes to be registered and request approval. The central authority reviews the application and assigns the organization a sub domain under one of the existing top-level domains.

The domain name system is quite general because it allows multiple naming hierarchies to be embedded in one system. In order to allow clients to distinguish among multiple kinds of entries, each named item stored in the system is assigned a type the specifies whether it is the address of a machine, a mailbox, a user, and so on.

When a client asks the domain system to resolve a name, it must specify the type of answer desired. For example, when an electronic mail application uses the domain system to resolve a name, it specifies that the answer should be the address of a mail exchanger.

In addition to specifying the type of answer sought, the domain system allows the client to specify the protocol family to use.

The domain system partitions the entire set of names by class, allowing a single database to store mappings for multiple protocol suites.

The syntax of a name does not determine what type of object it names or the class of protocol suite. In particular, the number of labels in a name does not determine whether the name refers to an individual object or a domain.

2.3.2 Mapping Domain Names To Addresses

The domain name scheme includes an efficient, reliable, general-purpose, distributed system for mapping names to addresses. The system is distributed in the sense, meaning that a set of servers operating at multiple sites co-operatively solve the mapping problem. It is efficient in the sense that most names can be mapped locally; only a few require Internet traffic. It is general purpose because it is not restricted to machine names.

Finally, it is reliable in that no single machine failure will prevent the system from operating correctly.

The domain mechanism for mapping names to addresses consists of independent, co-operative systems called name servers. A name server is a server program that supplies name-to-address translation, mapping from domain names to IP number addresses. The client software, called a name resolver, uses one or more name servers when translating a name.

The easiest way to understand how domain servers work is to imagine them arranged in a tree structure that corresponds to the naming hierarchy.

The root of the tree is a server that recognizes the top-level domains and knows which server resolves each domain. Given name to resolve, the root can choose the correct server for that name. At the next level, a set of name servers each provides answers for one top-level domain (e.g., edu). A server at this level knows which servers can resolve each of the sub domains under its domain.

The conceptual tree continues with one server at each level for which a sub domain has been defined.

The servers may be located at arbitrary locations on an Internet. The server uses the Internet for communication.

2.3.3 Domain Name Resolution

Domain name resolution proceeds top-down, starting with the root name server and proceeding to servers lojcated at the leaves of the tree. There are two ways to use the domain server system:

- Contacting name servers one at a time
- Asking the name server system to perform the complete translation

In either case, the client software forms a domain name query that contains the name to be resolved, a declaration of the class name, the type of answer desired, and a code that specifies whether the name server should translate the name completely. It sends the query to a name server for resolution.

When a domain name server receives a query, it checks to see if the name lies in the sub domain for which it is an authority. If so, it translates the name to an address according to its database, and appends an answer to the query before sending it back to the client.

If the name server cannot resolve the name completely, it checks to see what type of interaction the client specified. If the client requested complete translation, the server contacts a domain name server that can resolve the name and returns the answer to the client.

If the client requested non-recursive resolution, the name server cannot supply an answer. It generates a reply that specifies the name server the client should contact next to resolve the name.

A client must know how to contact at least one name server. To ensure that a domain name server can reach others, the domain system requires that each server know the address of at least one root server. In addition, a server may know the address of a server for the domain immediately above it.

Domain name servers use a well-known protocol port for all communication, so clients know how to communicate with a server once they know the IP address of the machine in which the server executes. There is no standard way for hosts to locate a machine in the local environment on which a name server runs; that is left to whoever designs the client software.

In some systems, the address of the machine that supplies domain name service is bound into application programs at compile time, while in others, the address is configured into the operating system at start-up. In others, the administrator places the address of a server in a file on secondary storage.

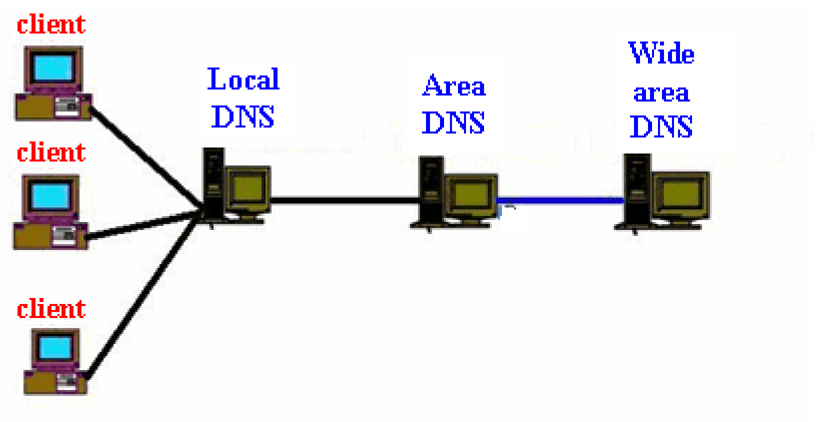


Fig.2.3.2 Domain name resolution

2.3.4 Efficient Translation

It may seem natural to resolve queries by working down the tree of the name servers, but it can lead to inefficiencies for three reasons.

- Most name resolution refers to local names, those found within the same subdivision of the name space as the machine from which the request originates. Tracing a path to contact the local authority would be inefficient.
- If each name resolution always started by contacting the topmost level of the hierarchy, the machine at that point would become overloaded.
- Failure of machines at the topmost levels would prevent name resolution, even if the local authority could resolve the name.

These ideas lead to a two-step name resolution mechanism that preserves the administrative hierarchy but permits efficient translation.

In the two-step name resolution process, resolution begins with the local name server. If the local server cannot resolve a name, the query must then be sent to another server in the domain system.

2.3.5 Name caching

The cost for lookup for non-local names can be extremely high if resolvers send each query to the root server. Even if queries could go directly to the server that has authority for the name, name lookup can present a heavy load to an Internet.

Therefore to improve the overall performance of a name server system, it is necessary to lower the cost of lookup for non-local names.

Internet domain name servers use **name caching** to optimize search costs. Each server maintains a cache of recently used names as well as a record of where the mapping information for that name was obtained. When a client asks to resolve a name, the server first checks to see if it has authority for the name according to the standard procedure. If not, the server checks its cache to see if the name has been resolved recently. Servers report cached information to clients, but mark it as a non authoritative binding, and give the domain name of the server, S, from which they obtained the binding. The local server also sends along additional information that tells the client the binding between S and an IP address. Therefore clients receive answers quickly, but there is a chance that information may be out-of-date/inaccurate.

If efficiency is important, the client will choose to accept the non-authoritative answer and proceed. If accuracy is important, the client will choose to contact the authority and verify that the binding between name and address is still valid.

2.3.6 Time to live

Name to address binding changes, therefore to keep the cache correct, servers time each entry and dispose of entries that exceed a reasonable time. Servers do not apply a single fixed time out to all entries, but allow the authority for an entry to configure its time-out. Whenever an authority responds to a request, it includes a **time to live (TTL)** value in the response that specifies how long it guarantees the binding to remain. Authorities can reduce network overhead by specifying long time-outs for entries that they expect to remain unchanged, while improving correctness by specifying short time-outs for entries that they expect to change frequently.

Caching is important in host as well as in local domain name servers. The host downloads the complete database of names and addresses from a local domain server at start-up, maintains its own cache of recently used names, and uses the server only when names are not found. A host that maintains a copy of the local server database must check with the server periodically to obtain new mappings, and the host must remove entries from its cache after they become invalid.

Keeping a copy of the local server's database has several advantages:

- It makes resolution on local hosts extremely fast because no network activity is involved.
- The local site has protection in case the local name server fails.
- It reduces the computational load on the name server, and makes it possible for a given server to supply names to more machines.

2.3.7 Abbreviation Of Domain Names

Abbreviation provides a method shortening names when the resolver can supply part of the name automatically. As in the telephone number hierarchy, when we make a call within the local area, we do not add the STD code.

For example, within the department of mathematics in Osmania University, the abbreviate name oumath is equivalent to math.osmania.ac.in

Most client software implements abbreviation with a domain suffix list. The net master configures a list of suffixes that can be appended.

An abbreviation is not part of the DNS but exists in many clients software's to make local names easy for the users.

In short the hierarchical naming systems let us use a lot of names without one site of administration.The DNS offers a hierarchical naming scheme.

The DNS uses distributed search in which domain name servers map each name to an IP address. Client begins with the local resolver. If the name is not found the name resolving is done through a tree of servers.

2.4 Dialup networking

In dialup networking, the existing telephone lines are used to get Internet connectivity. The block diagram is as shown

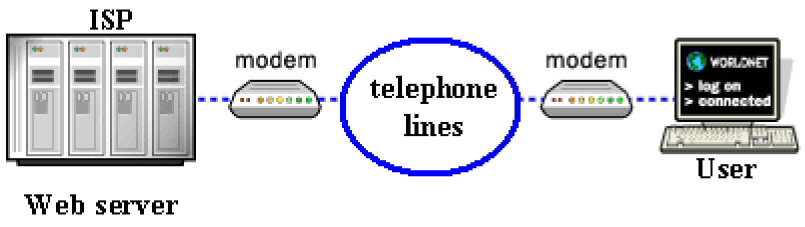


Fig.2.4.1 Accessing Internet through dialup

The technology most often used to access Internet services is dialup networking. Most ISPs offer this kind of telephone-based service. To use dialup access, a computer must have a modem that connects to the phone lines. The modem is known as one type of data communication equipment. Once properly configured, it instructs the modem to call to number provided by the ISP. At the ISP end there exists another modem, which answers the call and agrees to send Internet packets.

2.4.1 Access criteria

Once gets connected, the dialup Internet access appears to provide users the same kind of service as a dedicated / leased circuit (like broad-band). However there are some fundamental differences between dialup access and dedicated access. A dedicated access has more bandwidth and instant connection to the net. In the dialup access there is a tedious procedure of establishing the connection to the web through modem dialing. The user has to wait for modem to dial the required number and wait for ISP’s response every time he/she wishes to logon to the net. Once ISP’s modem and user’s modem gets synchronized then only the user can start browsing through another application software called web browser.

- In dialup connection the user dials the number provided by the ISP

- If the webserver is not engaged lifts the phone. The carrier detect (CD) lamp on the user modem lits indicating that it has established the contact with the remote host
- After asking user id and password the system logged on to the Internet.

The protocol for routing these packets through the modem is called the **Point-to-Point Protocol** (**PPP**). The basic idea is simple – the users' computer's TCP/IP stack forms its TCP/IP data grams normally, and then the datagrams are handed to the modem for transmission. The ISP receives each datagram and routes it appropriately onto the Internet. The same process occurs to get data from the ISP to the users' computer.

2.5 The Internet service provider (ISP)

A company that provides Internet access is known as an Internet service provider (ISP). Like any company, an ISP charges for its services. In general, ISPs levy two types of fees:

- A charge for using the Internet.
- A charge for a physical connection to the Internet.

All customers of an ISP must pay the first type, a charge for the Internet use. User charges are usually billed at flat rate. That is, the ISP charges each customer a fixed rate per month, independent of the number of minutes a customer uses the service, the destinations with which the customer communicates, or the amount of data transferred. In return for the use charges, the ISP agrees to forward packets from customer's computers to destinations on the Internet and from the computers on the Internet back to the customer's computer.

Although user charges are billed at a fixed rate, ISPs do discriminate among classes for users. For example, an ISP charges more to a business that transfers large volumes of data daily than an individual who has a single computer and uses the Internet casually. In addition, the ISP may make the rate depend on the type of physical connection a customer has – a customer whose connection is capable of transferring larger volumes of data is charged more than a customer with a lower-capacity connection.

The second type, a charge for a connection, applies only to customers who have a separate, dedicated connection between their site and the ISP.

2.6 Newer technologies

The following points made to develop newer technologies

- The technology must accommodate large number of connections and must be inexpensive.
- Provide high-speed connectivity economically.
- Till recently only leased digital circuit was used for high-speed connection. The customers used pay some large amount as installation charges and then monthly fee depending on bandwidth.
- For most people dialup connection was least expensive but not instant access

2.6.1 Cable modem

Makes use of already laid cables meant for TV viewing. A cable modem has a small electronic box that connects customer's computer to the cable system. In addition to the cable modem at each customer's site, the cable company needs cable modems at its end. The modems carry the data traffic over the cable TV's coaxial lines without interfering with other TV channels. This is made possible by using FDM (Frequency domain multiplexing) technique (that is how several TV channels coexists on a single coaxial cable).

- They are faster than dialup connection.
- Provides continuous and instant connection.
- Cheaper than leased lines.
- No extra wiring is needed.
- As the number of users grows there is a performance penalty.

2.6.2 Asymmetric digital subscriber line (ASDL)

Asymmetric digital subscriber line was invented to provide high-speed access over the existing telephone lines. ASDL should not be confused with dialup access because the former does not use modem. Instead, it uses only the wiring. These telephone wires carry digital signals without affecting the telephone signals and vice versa.

- Uses existing telephone wires.
- Allows simultaneous use of Internet and phone
- ADSL provides high data rates comparable with that of cable modem.
- Each user has separate pair of wires and thus no sharing of bandwidth.
- Continuous and instant connection.
- Cheaper than leased lines.

2.6.3 Wireless technology

To provide remote areas engineers have developed wireless access technology. They use same technology as cell phone but need not dial a number to access the Internet. The transmitter runs all the time and thus providing continuous and instant access. More over wireless technology offers high data transfer rates like cable modem and ADSL.

Summary:

The Internet is based on client server model. The server is designed to be more reliable because all the PCs in the network depend on its proper functioning. If it fails, the entire network suffers.

The Domain Name Service (DNS) is a network service that translates domain names to IP address which is a string of numbers appended to each other and separated by periods. Most users of the domain technology follow the hierarchical labels used by the official Internet domain system.

Internet domain name servers use **name caching** to optimize search cost.
In dialup networking, the existing telephone lines are used to get Internet connectivity.
Cable modem makes use of already laid cables meant for TV viewing. A cable modem has a small electronic box that connects customer’s computer to the cable system.

Asymmetric digital subscriber line was invented to provide high-speed access over the existing telephone lines. These telephone wires carry digital signals without affecting the telephone signals and vice versa.

Short questions:

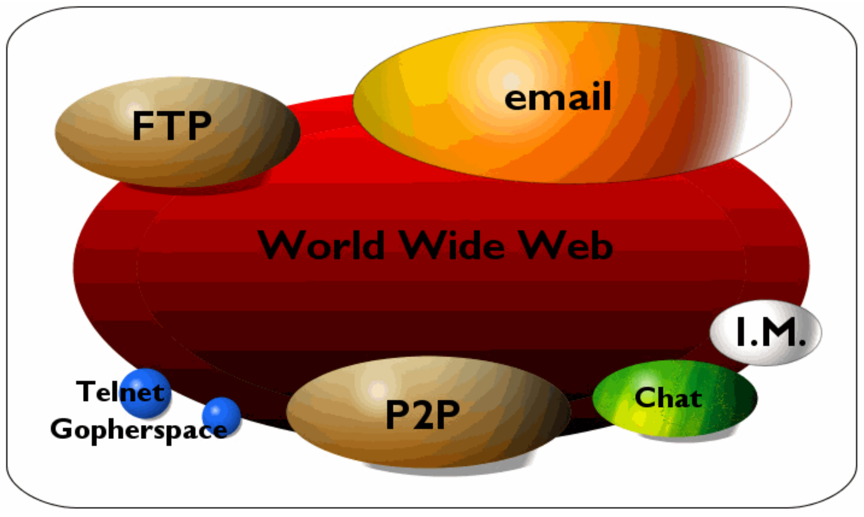
1). State the advantages of client/server model
2). Why servers should robust?
3). What is socket?
4). Contrast official/unofficial domain names
5). List atleast three domain names
6). What is the protocol employed in dialup net.
7). Modem stands for …..
8). The one who provides Internet access is ….
9). The advantage of ADSL is
10). What is time to live?

Long questions:

1). Differentiate a time sharing system with client/server model
2). What are the intricacies involved in a client/server model?
3). How client/server communicate? – Discuss in detail
4). Narrate the domain name system
5). Describe various schemes of domain name schemes
6). What is domain name resolution?
7). How caching improves DNS
8). Draw the block diagram of dialup networking and explain its operation
9). State the functions of ISP
10). Depict various Internet access schemes with their access criteria.

3.1 The Internet

The Internet is not confined to just web browsing. It has several other services, which are equally popular. The following diagram shows the various services offered by the Internet.



3.2 World Wide Web (WWW)

The World Wide Web (abbreviated as the Web or WWW) is a system of Internet servers that supports hypertext to access several Internet protocols on a single interface. Almost every protocol type available on the Internet is accessible on the Web. This includes e-mail, FTP, Telnet, and Usenet News. In addition to these, the World Wide Web has its own protocol: Hyper Text Transfer Protocol, or HTTP.

The World Wide Web provides a single interface for accessing all these protocols. This creates a convenient and user-friendly environment. It is no longer necessary to be conversant in these protocols within separate, command-level environments. The Web gathers together these protocols into a single system. Because of this feature, and because of the Web's ability to work with multimedia and advanced programming languages, the Web is the fastest-growing component of the Internet.

The operation of the Web relies primarily on hypertext as its means of information retrieval. HyperText is a document containing words that connect to other documents. These words are called links and are selectable by the user. A single hypertext document can contain links to many documents. In the context of the Web, words or graphics may serve as links to other documents, images, video, and sound. Links may or may not follow a logical path, as each connection is programmed by the creator of the source document. Overall, the Web contains a complex virtual web of connections among a vast number of documents, graphics, videos, and sounds.

Producing hypertext for the Web is accomplished by creating documents with a language called HyperText Markup Language, or HTML. With HTML, tags are placed within the text to accomplish document formatting, visual features such as font size, italics and bold, and the creation of hypertext links. Graphics and multimedia may also be incorporated into an HTML document. HTML is an evolving language, with new tags being added as each upgrade of the language is developed and released. The World Wide Web Consortium (W3C), coordinates the efforts of standardizing HTML. The W3C now calls the language XHTML and considers it to be an application of the XML language standard.

The World Wide Web consists of files, called pages or home pages, containing links to documents and resources throughout the Internet.

The Web provides a vast array of experiences including multimedia presentations, real-time collaboration, interactive pages, radio and television broadcasts, and the automatic "push" of information to a client computer. Programming languages such as Java, JavaScript, Visual Basic, Cold Fusion and XML are extending the capabilities of the Web. A growing amount of information on the Web is served dynamically from content stored in databases. The Web is therefore not a fixed entity, but one that is in a constant state of development and flux.

3.3 Web browsing

With your web browser as your ship, you can navigate through an ocean of information on the web. A browsing service permits users to view information from remote computers without knowing the names of the individual file names

In order to use the Internet, you need to access to some computer that has access to the Internet.
This computer might be:

- Your personal computer at home, connected to the Internet through an Internet Service Provider (ISP).
- A computer or workstation at your school or place of employment that is connected to the Internet
- A computer at a library or Internet cafe, which is available for public use
- A handheld device that accesses the Internet through wireless signals
- A computer or your home or office that uses Internet connections such as a cable modem, DSL, or some other means
- A terminal at your school or place of work that is connected to a main computer via Web browsers

3.3.1 Browser software access the web

The world wide web uses client-server interaction. The browser program acts as a client that uses the Internet to contact a remote server for a copy of the requested page on the web.

Browser software can display audio, video, text and graphics. When a document appears on the screen, some of the items contain a link to another document. It permits the user to navigate through document's hypertext media with the help of mouse.

Browsers interact with user in two ways. The browser's primary form of interaction occurs when the browser obtains an item that the user has requested; the browser displays the contents of the web site requested. The displayed document may contain integrated menus and embedded text called hypertext. The other form of interaction occurs when the displayed documents contain several other hyper links. The user has choice going further deep into the content at his will.

An identifier used to specify a particular page of WWW information is called **Uniform Resource Locator (URL)**. When a browser displays a page it also displays the URL of the page in the address bar of the browser. An URL is analogous to a telephone number - a short string that identifies uniquely among millions of web pages. Given a valid URL, a browser can go directly to the page without passing through other documents.

3.3.2 Start up a Web browser

Once you have access to an Internet-connected computer, you can access the Web if that computer has Web browser software installed. Two popular Web browsers are Netscape Navigator and Microsoft Internet Explorer. Small handheld devices may have their own versions of these Web browsers or have a special Web browser that operates on them.

Find out what Web browser the computer has and start it up. If you have a Web browser, you can access much of the Internet's content on Web sites.

3.3.3 Using the Web browser

You use the Web browser typically through manipulating it with your mouse and cursor, and entering information from your keyboard.

To open a **Web address (URL)**, use your browser's menu and chose File->Open Location. You might also be able to click on the text in the "Address" box at the top of your browser, alter or enter a URL there, and press the Return key.

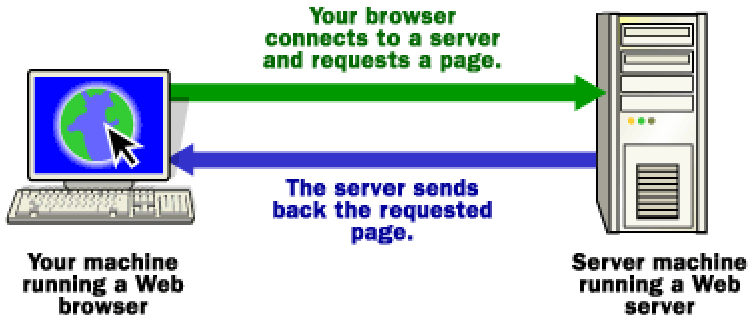

Your browser connects to a server and requests a page.
The server sends back the requested page.
Your machine running a Web browser
Server machine running a Web server

Fig. 3.3.1 A browsing service using client server interaction.

To find out more ways to access a Web site, you can look in your browser's documentation about how you can "open a URL" or "point your browser to a URL" or "bring up a Web site". Use your browser's Help or online documentation to learn more about your browser, such as how to bookmark information or create folders of "favorites."

At the most basic level possible, the following diagram shows the steps that brought that page to your screen:

- Browser formed a connection to a Web server, requested a page and received it.
- When you type a URL into a browser, say “http://www.google.com/web-server.htm” the following steps occur:
- The browser breaks the URL into three parts:
- The protocol ("http")
- The server name ("www.google.com")
- The file name ("web-server.htm")
- The browser communicates with a name server to translate the server name, "www.google.com" into an IP address, which it uses to connect to that server machine.
- The browser then forms a connection to the Web server at that IP address on port 80 (well known port for http).
- Following the HTTP protocol, the browser sends a GET request to the server, asking for the file "http:// www.google.com"
- The server sends the HTML text for the Web page to the browser. The browser reads the HTML tags and formats the page onto the screen.
- Note here the browser running client software and web server running server software.

3.3.4 Bookmarks

To move quickly to a user’s favorite URL, browsers have special mechanism called hot list or bookmark. This mechanism allows maintaining one’s own favorite sites and upon clicking bookmarks button browser displays the list in the form of a pull down menu. The user can pick any one from the list.

The advantage is convenience and fast. Every time an user need not type the entire URL address in the address bar.

The bookmarks are permanent. Once user saves an URL in the bookmark list they are stored permanently even after the system gets shutdown. By running add/remove bookmarks utility option of the browser the user can maintain his favorite list.

3.3.5 Cookies

Cookies provide capabilities that make the Web much easier to navigate. The designers of almost every major site use them because they provide a better user experience and make it much easier to gather accurate information about the site's visitors. They started receiving tremendous media attention back in 2000 because of Internet privacy concerns, and the debate still rages.

A cookie is a piece of text that a Web server can store on a user's hard disk. Cookies allow a Web site to store information on a user's machine and later retrieve it. The pieces of information are stored as **name-value pairs**.

Using cookies, sites can determine:

- How many visitors arrive
- How many are new vs. repeat visitors
- How often a visitor has visited
- Sites can **store user preferences**
- E-commerce sites can implement things like **shopping carts**

In e-commerce cookies allows selling your information to others who might want to sell similar products to you. That is the fuel that makes telemarketing and junk mail possible. An undesirable side effect!

3.3.6 Browser in detail

Whichever Web browsers you have, spend some time getting to know it. Learn about some customizations that can save you time and make it easier for you to surf the Web.

Microsoft **Internet Explorer** is the most used Web browser.

- **Customize your Web browser**

There are so many things, which a browser can do (see the fig.3.3.2). In the presentation option you can over ride many specifications, which a web designer has chosen for transmission of his web page. For example choose a font scheme to override Web "designers" who fix the font size to be too small:

Choose the menu Tools->Internet Options. Choose the Accessibility box (lower right-hand corner). Click the boxes for "Ignore font styles specified on Web pages"

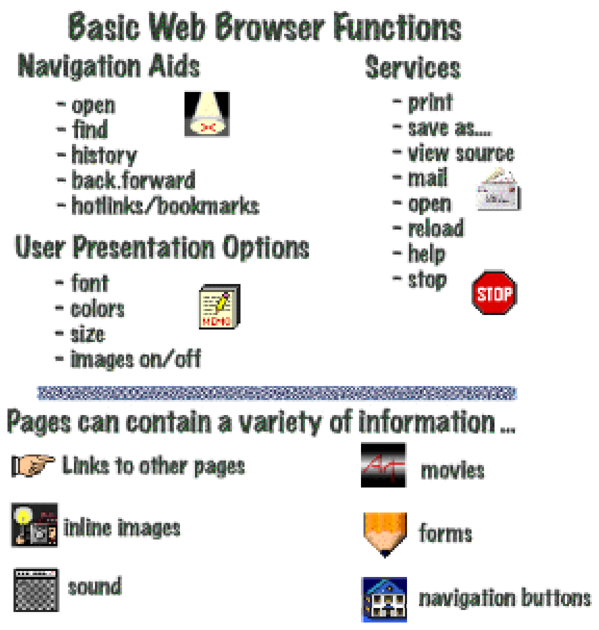


Fig.3.3.2 Browser functions

Set your own font style in menu Tools->Internet Options, Fonts box. Choose your own font sizes in menu View->Text Size.

Set your own start page; or use a blank page (setting a blank page to be your start page--this saves a lot of time when your Web browser launches as it does not have to load a Web page from the Internet.). The Web browser manufacturer who usually sets it to a very graphically busy page like msn.com sets the default Web page when you first install your Web browser. Every time you launch a Web browser, this page has to load.

If you want to set your start page to be a specific Web page, bring that page up in the Web browser. This page could be an HTML file stored on your hard drive.

Refuse cookies except from sites you approve. A cookie is a file that is downloaded to your hard drive as a result of visiting a Web site. Every Web page or image you see in your Web browser has to be downloaded to your disk in a temporary area on your hard drive, otherwise you wouldn't be able to see it at all. But cookies are special little files that can be used to track sites that you visit. Many people like to clean them out or control when they are used. Use cookie management built into your browser. Internet Explorer has some good features for controlling cookies. Here is how to set them:

- Choose the Internet Explorer menu option Tools->Internet Options
- Choose **Privacy tab**
- Choose **Advanced** box
- Check **"Override automatic cookie handling"**
- Check Block in First-party Cookies
- Check Block in Third-party Cookies
- Check Always allow session cookies
- Check OK

In Privacy tab, select Edit to handle cookies for individual Web sites Enter the domain names of trusted Web sites that save information for you from session to session or require cookies in order to operate properly. For example: osmania.ac.in, yahoo.com, etc. Check OK

- **Customizing browser's toolbar:**

menu View->Toolbars->Customize... explore the options

- **Use key sequences to save time**

For each of these, make sure the Web browser of interest is the active window (indicated by highlighted title bar; click on a window to make active).

The following are few key sequences worth remembering

Internet Explorer Key Sequences

Ctrl/N	Launch another Web browser
Ctrl/F	Search for a text string on a Web page (the F stands for "Find")
Ctrl/W	Close a Web browser
Ctrl/H	Bring up your history in a side panel; shows pages you have visited
Ctrl/I	Bring up your favorites in a side panel; shows pages you have "bookmarked" to quickly visit again
Ctrl/D	Save a page URL to your favorites
Esc	Stop the Music! (Some Web "designers" cause a music file to automatically start when you visit a Web page; also stops loading graphics.)
Tab	Move to next field in a Web form.
Tab	Move to next link on Web page
Alt/D	Move to the address box
Shift/click on hypertext link	This will cause the link to open in a new Web browser

Internet Explorer Tricks

Get to a something.com site quickly; Works for sites of the form http://www.something.com	Type something in address (URL) box Press **control/Enter**
Get to a something.else site quickly; Works for sites of the form http://something.else	Type the something.else in address (URL) box Press Enter
Save a picture from a Web page to your computer's hard drive	Place your cursor over picture Right click mouse Select "Save Picture As ..." Or choose "Set as Background" to make picture your wallpaper
Overcome dark text on dark or textured background; some Web "designers" do this, making the page hard (or impossible) to read!	Choose menu Edit->Select All (or Ctrl/A) Resulting page is in reverse video--perhaps enabling you to read the page If this fails: Choose the menu Tools->Internet Options. Choose the Colors box (lower left-hand corner) Unclick the box for "Use windows colors" and then select your own colors

- **Stop the popup windows**

You may discover that when visiting or leaving some Web sites, a new window, usually containing an advertisement, will appear on your computer. These are called popup windows and sometimes they can be irritating and make using the Web efficiently almost impossible.

3.3.7 The elements of web navigation

Indexes and search engines of various kinds have greatly facilitated access to the wealth of information on the Internet. There are three general types of sites that can help you find information on subjects that you're interested in:

Directory Sites. These sites put web sites into a structure of predefined categories after a review by a human being. You should start with a directory site if you are looking for a category of information, like the Environment, Gardening, or Photography, or for a well-known site likely to be in their directory. They can be a good place to search first followed by a wider investigation with a general search engine if necessary.

Search Engines: Search engines automatically scan millions of web sites across the Internet, and then provide you with search access to the resulting database. These databases are larger than those of directory sites, but don't use any human quality control. You should use a search engine when you are looking for detailed information, when you want to search the largest number of web pages, and when you want to use advanced search features.

Specialized Search Sites: These sites provide specialized search functionality such as meta-searches (searching several engines at once), multimedia searches, legal information searches, and other capabilities.

There are four directions you can surf, and five basic techniques.

Directions: From any page you can surf in one of four directions:

- **Back**. Go back to the previous page -- press the Back button, right-click on the window and select "Back", or press <alt><left arrow>.
- **Forward**. Go forward to a new page after going back -- press the Forward button, right-click on the window and select "Forward", or press <alt><right arrow>.
- **Link**. Click a link and jump to a new page.
- **Jump**. Select a new page from an external source such as your *bookmarks*.

Techniques: The five basic surfing techniques are described as follows:

Surfing: You don't have to wait for a page to load to either click a link, press the back button, or select a new link from your bookmarks. You can take action whenever you are ready. Jumping ahead of the browser is recommended if the link you want is already loaded but the rest of the page is lagging behind. When you click on a link as soon as it is available, you speed up and enjoy the feeling of surfing from wave to wave.

Chains: After you click on several links and proceed through several pages, you create a chain of web sites accessible with the down-arrow beside your browser's Back button. You can click on the browser Back button to return to a previous page and read it again, and then repeatedly click Forward to return to the last page without the trouble of finding the links you used last time. If you click a new link from any page, you start a new chain from that point on.

Reloading: You can stop and then reload a page at any time if it is having problems loading or to ensure you have the latest copy of a page that updates regularly. Click the Refresh button/or right-click on the window and select "Refresh", press <ctrl>-r, or press F5.

Stopping: You can stop the load of any page at any time by clicking Stop on the toolbar or pressing <Esc>. The browser will display as much of the page as it loaded, and all of the displayed links will be operational.

Restarting: If a page seems to be taking a long time to load, don't hesitate to stop the connection and then select the link again. As long as the messages in the bottom border show that some parts of the page are loading then you should let it continue, but if nothing happens for more than a minute then something is likely stalled, and you should stop and reload the page again. HTTP connections often get dropped on busy web sites, and requesting the page again will often load it quickly on a new connection.

3.3.8 Searching

When should one use a search engine?

When you have a narrow or obscure topic or idea for research

- When you are looking for a specific site
- When you want to search the full text of millions of pages

- When you want to retrieve a large number of documents on your topic
- When you want to search for particular types of documents, file types, source locations, languages, date last modified, etc.
- When you want to take advantage of newer retrieval technologies such as concept clustering, ranking by popularity, link ranking, and so on

- **Google search engine is useful when...**

  - You are looking for a specific fact/person/event/narrow topic
  - Your topic is made up of multiple ideas
  - You like Google's specialized features such as spell checking, phone book and flight lookups, stock prices, etc.
  - You want to take advantage of Google's advanced search interface that lets you fill out a form to do a search targeted to your needs

Google is a general search engine that is everyone's favorite these days. It ranks results by the number of links from pages also ranked high by the service. This unique ranking system can be quite effective.

Google Features: Returns results ranked by the number of links from a high number of pages ranked high by the service; the number of links to them also determines high-ranking pages. In determining relevancy ranking, the engine also looks at various textual clues including linking text
Suggests an alternative searches when search terms are misspelled.
Search results include sites from the *Open Directory Project*, offering an interesting mix of sites from the wider Web and those chosen by editors for inclusion into the directory the *Google Web Directory*.

Requires no syntax: By default it has AND association. i.e you need not put AND between two words
OR searching is supported if "OR" is typed in CAPS, e.g., university OR college; works only with multiple single words
For more refined searches, use quotations for exact phrases ("El Nino")
A minus sign (-) for which doesn’t have that phrase
Results include the text from the source document that matches your query
I'm feeling lucky option returns the top-ranked source for a query.
Searches the deep Web for such information as:

Files in Portable Document Format, Microsoft Word, Excel, and PowerPoint, Rich Text Format and PostScript
Images, from the Advanced Web Search interface or from *Google Image Search*
Maps from Yahoo! or MapBlast (enter an address)
Phone book entry (enter first and last name, and city or zip)
Stock prices (enter a comapny's ticker symbol)

- **Limitations:**

New Web pages will not appear in your results, as it takes time for the creators of other Web pages to link to new resources. People who maintain Web pages can manipulate Google results. Hackers and others sometimes attempt to associate words with a link to a specific Web site to make a political or other point. For example, the search terms "miserable failure" point to the official George Bush site. This site may become the number one hit on Google, even though the words are not relevant to it.

Few search examples..

Concept search
Query: I'd like to learn more about Richard Nixon's resignation.

Type: Nixon resignation [Google defaults to Boolean AND logic]
Examine the results for relevancy
Note the related categories from the Google Web Directory listed at the top of the results screen

Phrase Search
Query: I'd like to see information on the movie Gone with the Wind.

Type "Gone with the Wind" and press Enter/or click search button
[search criteria in quotes, capitalization is not necessary]
See the results

Field Search
Field searching is a way to narrow the search to specific parts of the document or record. Google offers a variety of ways to use field searching to better focus your results. First, let's try a simple search that is *not* a field search.
Query: I'd like to see information on slavery.

This is isn't the wisest search to do in a large, full-text database like Google because it brings back too many results.

Let's look into ways to focus our results by using field searching. We will try these searches using Google's basic search box. Keep in mind that most of Google's field search options are also available on their *Advanced Search* form that is even easier to use.

intitle:slavery will look for slavery in the title field embedded within the HTML document.
inurl:slavery   will look for slavery in the URL of the file,

Query: I'd like information about the Mars rover missions from the NASA site.

Search: +"Mars rover"   +site:nasa.gov

This is a nicely-focused search. It uses the plus (+) sign to be sure that all of our search terms appear on the retrieved documents. In addition, the phrase **Mars Rover** is enclosed in quotation marks, and we have narrowed our search to retrieve documents only from the NASA site.

3.4 News groups

A newsgroup is a repository, for messages posted from many users at different locations. The term is somewhat confusing, because it is usually a discussion group. Newsgroups are technically distinct from, but functionally similar to, discussion forums on the World Wide Web. Newsreader software is used to read newsgroups.

3.4.1 News group Hierarchies

Newsgroups are often arranged into hierarchies, theoretically making it simpler to find related groups. The term top-level hierarchy refers to the hierarchy defined by the prefix prior to the first dot.

The most commonly known hierachies are the usenet hierarchies. So for instance newsgroup rec.arts.sf.starwars.games would be in the rec.* top-level usenet hierarchy, where the asterisk (*) is defined as a wildcard character. There were seven original major hierarchies of usenet newsgroups, known as the "Big 7":

- *comp.**—Discussion of computer-related topics
- *news.**—Discussion of Usenet itself
- *sci.**—Discussion of scientific subjects
- *rec.**—Discussion of recreational activities (e.g. games and hobbies)
- *soc.**—Socialising and discussion of social issues.
- *talk.**— Discussion of contentious issues such as religion and politics.
- *misc.**—Miscellaneous discussion—anything which doesn't fit in the other hierarchies.

These were all created in 1986–1987, prior to which all of these newsgroups were in the net.* hierarchy. At that time there was a great controversy over what newsgroups should be allowed. Among those that the usenet (who effectively ran the Big 7 at the time) did not allow hierarchies concerning recipes, drugs, and sex.

This resulted in the creation of an *alt.** (short for "alternative") usenet hierarchy where these groups would be allowed. Over time the laxness of rules on newsgroup creation in alt.* compared to Big 7 meant that many new topics that could, given time, gain enough popularity to get a Big 7 newsgroup had newsgroups instead created in alt.*. This resulted in a rapid growth of alt.* which continues to this day.

3.4.2 Types of newsgroups

Typically, a newsgroup is focused on a particular topic such as "shellfish". Some newsgroups allow the posting of messages on a wide variety of themes, regarding anything a member chooses to discuss as on-topic, while others keep more strictly to their particular subject, frowning on off-topic postings. The news admin (the administrator of a news server) decides how long articles are kept before being expired (deleted from the server). Usually they will be kept for one or two weeks, but some admins keep articles in local or technical newsgroups around longer than articles in other newsgroups.

Newsgroups generally come in either of two types, binary or text. There is no technical difference between the two, but the naming differentiation allows users and servers with limited facilities the ability to minimize network bandwidth usage. Generally, Usenet conventions and rules are enacted with the primary intention of minimizing the overall amount of network traffic and resource usage.

Newsgroups are much like the public message boards on old bulletin board systems. For those readers not familiar with this concept, envision an electronic version of the corkboard in the entrance of your local grocery store.

There are currently well over 100,000 Usenet newsgroups, but only 20,000 or so of those are active. Newsgroups vary in popularity, with some newsgroups only getting a few posts a month while others get several hundred (and in a few cases several thousand) messages a day. Weblogs have replaced some of the uses of newsgroups [especially because, for a while, they were less prone to spamming(un-wanted messages)].

3.4.3 How newsgroups work

Various organizations and institutions host newsgroup servers. Most Internet Service Providers (ISPs) host their own News Server, or rent access to one, for their subscribers. There are also a number of companies who sell access to premium news servers.

Every host of a news server maintains agreements with other news servers to regularly synchronize. In this way news servers form a network. When a user posts to one news server, the message is stored locally. That server then shares the message with the servers that are connected to it if both carry the newsgroup, and from those servers to servers that they are connected to, and so on.

3.4.4 Binary newsgroups:

While Newsgroups were not created with the intention of distributing binary files, they have proven to be quite effective for this. Due to the way they work, a file uploaded once will be spread and can then be downloaded by an unlimited number of users. More useful is the fact that every user is drawing on the bandwidth of their own news server. This fact means that opposed to a P2P technology, the user's download speed is under their own control, as opposed to under the willingness of other people to share files. In fact this is another benefit of Newsgroups: it is usually not expected that users share. If every user makes uploads then the servers would be flooded, thus it is acceptable and often encouraged for users to just leech.

Files can be attached to a post, but there is a very small limit in the size of the file, which can be attached. As such people have come up with methods to encode a file into text, which is posted as a post rather than attached to a post. There is a limit to how large a single post may be as well, so methods were developed to chain multiple posts together. The newsreader then intelligently joins the posts and decodes it into a binary file. A number of websites exist for the purpose of keeping an index of the files posted to binary Newsgroups.

3.5 Mailing lists:

A mailing list is a collection of names and addresses used by an individual or an organization to send material to multiple recipients. The term is often extended to include the people subscribed to such a list, so the group of subscribers are referred to as "the mailing list", or simply "the list".

At least two quite different types of mailing lists can be defined: the first one is closer to the literal sense, where a "mailing list" of people is used as a recipient for newsletters, periodicals or advertising. Traditionally this was done through the postal system, but with the rise of e-mail, the electronic mailing list became popular.

When similar or identical material is sent out to all subscribers on a mailing list, it is often referred to as a mailshot.

Mailing lists are often rented or sold. If rented the renter agrees to use the mailing list for only the agreed upon times. The mailing list owner typically enforces this by "salting" the mailing list with fake addresses and creates new salts for each time the list is

3.6 Chat rooms:

A chat room is an online forum where people can chat online (talk by broadcasting messages to people on the same forum in real time). Sometimes these venues are moderated either by limiting who is allowed to speak (not common) or by having moderation volunteers patrolling the venue watching for disruptive or otherwise undesirable behavior.

Chat systems include Internet Relay Chat (or IRC, where rooms are called "channels"), There are several proprietary systems on the Microsoft Windows and Java platforms.

Some chat rooms go beyond text messages incorporating 2D and 3D graphics referred as visual chat or virtual chat. These environments are capable of incorporating elements such as games and educational material most often developed by individual site owners, who in general are simply more advanced users of the systems.

Some chatroom sites incorporate audio and video communications. People may chat in audio and watch each other there. Lesser known is the UNIX based talker. Chatrooms are often confused (especially by the popular media) with discussion groups, which are similar but do not take place in real time and are usually run over the World Wide Web.

Recently much chat room and instant messaging technology has begun to merge as the dominance of the big three instant messaging providers (AOL, Yahoo and MSN) have tied chat rooms directly into their instant messaging interfaces. This centralization trend is likely to continue to dominate the chat world as these providers begin to merge their services and cooperate in their IM and chat protocols.

3.7 E-Mail

Electronic mail was originally designed to allow a pair of individuals to communicative via computer. The first electronic mail software provided only a basic facility: it allowed a person using one computer to type a message and send it across the Internet to a person using another computer.

Current Electronic mail systems provide services that permit complex communication and interaction. For example Electronic mail can be used to:

- Send a single message to many recipients
- Send a message that includes text, audio, video or graphics.
- Send a message to a user on a network outside the Internet.
- Send a message to which a computer program responds.

Researchers working on early computer networks realized that networks can provide a form of communication among individuals that combines the speed of telephone communication with permanence of postal mail. A computer can transfer small notes or large documents across a network almost instantaneously. The Designers called the new form of communication *electronic mail* often abbreviated as *email.* The concept of Email has become extremely popular in the Internet as well as on most other computer networks.

To receive electronic mail, a user must have a mailbox, a storage area, usually on disk, that holds incoming email messages until the user has time to read them. In addition, the computer on which a mailbox resides must also run email software. When a message arrives, email software automatically stores it in the user’s mailbox. An email mailbox is private in the same way that postal mailboxes are private: anyone can send a message to a mailbox, only the owner can examine mailbox contents or remove messages.

Like a post office mailbox each email mailbox has a mailbox address. To send email to another user, one must know the recipients mail box address. Thus

- Each individual who participates in electronic mail exchange has a mailbox identified by a unique address xyz@yahoo.com
- Any User can send mail across the Internet to another user’s mailbox if they know the mailbox address.
- Only the owner can examine the contents of a mail box and extract messages.

To send electronic mail across the Internet, an individual runs an email application program on their local computer. The local application operates similar to a word processor-- It allows a user to compose and edit a message and to specify a recipient by giving a mailbox address.

Once the user finishes entering the message and adds attachments, email software sends it across the Internet to the recipient's mailbox.

When an incoming email message arrives, system software can be configured to inform the recipient. Some computers print a text message or highlight a small graphic on the users display (e.g., a small picture of letters in a postal mail box). Other computers sound a tone or play a recorded message. Still other computers wait for the user to finish viewing the current application before making an announcement. Most systems allows a user to suppress notification altogether, in which case the user must periodically check to see if email has arrived.

Once email has arrived, a user can extract messages from his or her mailbox using an application program. The application allows a user to view each message, and optionally, to send a reply. Usually, when an email application begins, it tells the user about the messages waiting in the mailbox. The initial summary contains one line for each email message that has arrived; the line gives the sender's name, the time that message arrived, and the length of the message. After examining the summary, a user can select and view messages on the list. Each time a user selects a message from the summary, the email system displays the message contents. After viewing a message, a user must choose an action. The user can send a reply to whoever sent the message, leave the message in the mail box so it can be viewed again, save a copy of the message in the file, or discard the message.

- A computer connected to the Internet needs application software before users can send or receive electronic mail.
- Email software allows a user to compose and send message or to read messages that have arrived.
- A user can send a reply to any message.

Usually, the sender only needs to supply information for the TO and SUBJECT lines in a message header because email software fills in the date and the senders mailbox address automatically. In a reply, the mail interface program automatically constructs the entire header. It uses the contents of the FROM field in the original message as the contents of the TO field in the reply. It also copies the subject field from the original message to a reply. Having software fill in the header lines is convenient, and also makes it difficult to forge email.

In practice, most email systems supply additional header lines that help identify the sending computer, give the full name of the person who sent the message, provide a unique message identifier that can be used for auditing or accounting, and identify the type of message (e.g., text or graphics). Thus, email messages can arrive with dozens of lines in the header. A lengthy header can be annoying to a recipient who must skip past it to find the body of a message. Software used to read email can make it easier for the recipient by skipping most header lines. To summarize:

Although most email messages contain many lines of header, software generates most of the header automatically. User-Friendly software hides unnecessary header lines when displaying an email message.

3.7.1 E-Mail Operation:

A computer communication always involves interaction between two programs called a client and a server. E-mail systems follow the client-server approach: Two programs co operate to transfer an email message from the sender's computer to the recipient's mail box (transfer requires two programs because an application running on one computer cannot store data directly in a mailbox on another computer's disk). When a user sends an email message, a program on the sender's computer become a client. It contacts an email server program on the recipient's computer and transfers a copy of the message. The server stores the message in the recipient's mailbox.

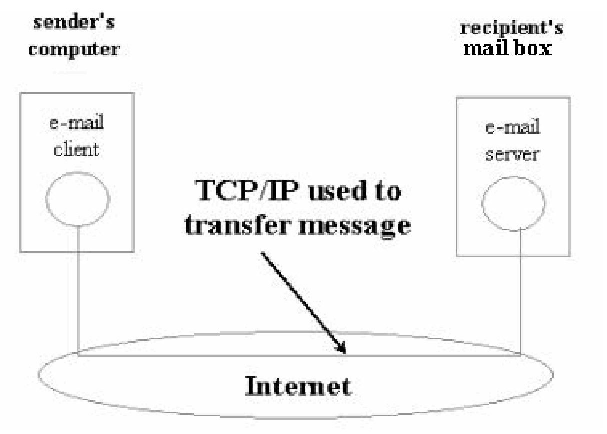


Fig.3.7.1 Email operation

Client software starts automatically as a user finishes composing an email message. The client uses the recipients email address to determine which remote computer to contact. The client uses TCP to send a copy of the email message across the Internet to the server. When the server receives a message, it stores the message in the recipient’s mailbox and informs the recipient that email has arrived.

The interaction between a client and a server is complex because at any time computers or the Internet connecting them can fail (e.g., someone can accidentally turn off one of the computers). To ensure that email will be delivered reliably, the client keeps a copy of the message during the transfer. After the server informs the client that the message has been received and stored on the disk, the client erases its copy.

A computer cannot receive e-mail unless it has an e-mail server program running. On large computers, the system administrator arranges to start the server when the system first begins, and leaves the server running at all times. The server waits for an email message to arrive, stores the message in the appropriate mailbox on disk, and then waits for the next message.

A user who has a personal computer that is frequently powered down or disconnected from the Internet cannot receive email while the computer is inactive. Therefore, most personal computers do not receive email directly. Instead, a user arranges to have a mailbox on a large computer with a server that always remains ready to accept an email message and store it in the users mailbox. For example, a user can choose to place their mailbox on their company’s main computer, even if they use a personal computer for most work. To read email from a personal computer, a user must contact the main computer system and obtain a copy of their mailbox.

3.8 Internet Fax

Fax (facsimile) is a simple form of digital transmission for transmission of images over voice grade telephone system. Each fax machine consists of four main components: a printer, scanner, dial-up modem and an embedded (dedicated) computer system, which coordinates the former three components.

While sending the machine uses modem to dial a number and wait for another modem to answer. Most fax machines are designed to answer after three ring tones. Once modem answers they get synchronized and the sending machine can use its scanner to scan the page line by line,

digitizes the image into electrical signals and transmits across the telephone lines.
At the receiving end incoming digitized values are sent to the printer to create a copy of the transmitted page.

Now with the advent of the Internet computers can be used to send a fax. The two-fax machines can be modified to communicate across the Internet or a file document can be sent from a computer to a fax machine. Since Internet handles digital information more efficiently the future fax machines are going to be based on the Internet technology.

3.9 File Transfer Protocol (FTP)

Although services like email, Internet fax can be utilized for sending files over the net they are not designed for handling large volumes of data. For sending large volumes of data reliably over the net **File Transfer Protocol (FTP)** is preferred instead.

FTP works in interactive environment. Just type ftp at the DOS command prompt to enter into ftp interactive session. FTP responds to each command the user enters. For example, when a session begins, the user enters a command to identify a remote computer. FTP then establishes a connection to the remote computer. In the same way, to terminate a session user tells FTP to relinquish its connection.

3.9.1 FTP commands

There are around 58 separate commands but the average user need to know only three following basic commands

- **open <name of the ftp computer>:** for connecting to a remote computer.
- **get <filename>:** for retrieve a file from the computer.
- **bye:** Terminate the connection and leave the ftp session.

ftp can be used not only to retrieving files but also for uploading/sending file by using **send** command. Once a connection has established just type the command **send** along with the file name to be sent. A copy of the file will be transferred to the remote computer. Of course, the FTP on the remote site must be configured to allow file storage. Many Internet sites that run ftp allow storage.

3.9.2 FTP File Types

FTP understands only two basic file formats. It classifies each file as either a text file or a binary file. Text file supports ASCII encoding. FTP has commands to convert a non ASCII text file to ASCII text file.

FTP uses classification of binary files for all non text files. For example the following type of files should be specified as a binary files.

- A computer program
- Audio data
- A graphic or video image.
- A spread sheet
- A word processor document
- Compressed files

The compressed file refers to a file, which has been processed to reduce its size by running file compress utility. By using file uncompressing utilities like unzip the original file can be reconstructed.

Choosing between binary and ASCII transfer can be sometimes difficult. When you are unsure about the type of the file choose binary option for transferring the file. If a user requests FTP to perform a transfer using incorrect type the resulting transferred copy may be damaged.

3.9.3 FTP login

The user must login into the ftp site as an authentic user before performing any ftp based transactions. Usually the user will be provided with login name and password. This way the site is protected from malicious users and keeps the data secure.

To make files available to the general public, a system administrator can configure FTP to honor **anonymous** FTP. It works like standard FTP, except that it allows anyone to access public files. To use anonymous FTP, a user enters the login as anonymous and the password as guest. Few sites may prompt for the user’s email address just in case of any errors like log failures so that those error messages can be emailed.

Most users invoke FTP through a web browser so that the ftp transactions can be made in a graphic user interface (GUI) environment

3.9.4 FTP operation

FTP operation is also based on client server model.

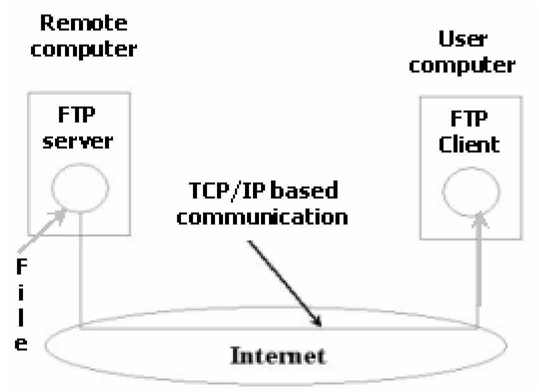


Fig.3.9.1 FTP operation

The user invokes a local FTP program or enters a URL that specifies FTP. The local FTP program or the user's browser becomes an FTP client that uses TCP to contact an FTP server program on the remote computer. Each time the user requests a file transfer, The client and server program interacts to send a copy of the data across the Internet.

The FTP server locates the file that the user requested, and uses TCP to send a copy of the entire contents of the file across the Internet to the client As the client program receives data, it writes the data into a file on the user's local disk. After the file transfer completes, the client and server programs terminate the TCP connection used for the transfer.

Here is an example typical FTP session:

```
$ ftp plaza.aarnet.edu.au
Connected to plaza.aarnet.edu.au.
220 plaza.aarnet.EDU.AU FTP server (Version wu-2.4(2) Fri Apr 15
14:04:20 EST 1994) ready.
Name (plaza.aarnet.edu.au:jphb): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230-
230-This is the AARNet Archive Server, Melbourne, Australia.
230-
230-
230-The disk that failed back in September is still not back on-line.
```

```
230-As a consequence of this, we are only shadowing files modified in
230-the last 100 days on many of the more popular archives. We
apologise
230-for this inconvenience.
230-
230-Local time is Tue Jun  4 17:46:00 1996
230-
230-Please read the file /info/welcome-ftpuser
230-  it was last modified on Fri Apr 22 14:47:05 1994 - 774 days ago
230 Guest login ok, access restrictions apply.
ftp> pwd
257 "/" is current directory.
226 Transfer complete.
214 bytes received in 0.018 seconds (11 Kbytes/s)
ftp> cd rfc
250 CWD command successful.
ftp> get rfc1048.txt.gz
200 PORT command successful.
150 Opening ASCII mode data connection for rfc1048.txt.gz (5141
bytes).
226 Transfer complete.
local: rfc1048.txt.gz remote: rfc1048.txt.gz
5161 bytes received in 1.6 seconds (3.2 Kbytes/s)
ftp> quit
221 Goodbye.
….
```

3.10 TELNET

The Telnet protocol is often thought of as simply providing a facility for remote logins to the computer via the Internet. This was its original purpose although it can be used for many other purposes.

It is best understood in the context of a user with a simple terminal using the local telnet program (known as the client program) to run a login session on a remote computer where his communications needs are handled by a telnet server program on the remote computer. It should be emphasized that the telnet server can pass on the data it has received from the client to many other types of process including a remote login server.

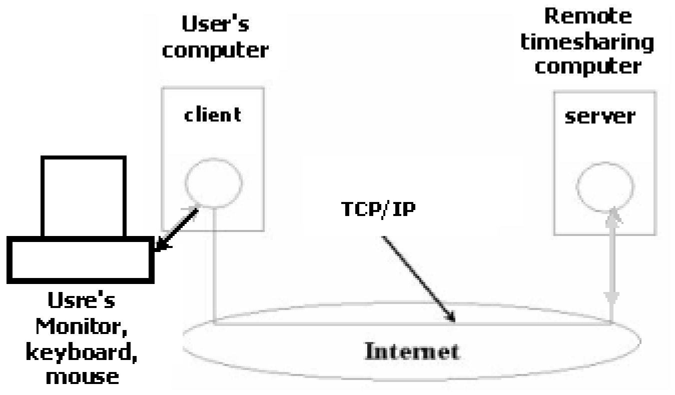


Fig 3.10.1 Telnet session

Once connection has been established between the client and server, the software allows the user to interact directly with the remote computer's operating system. For all user's inputs the server sends output and displays on user's screen.

After a user logs out of the remote computer, the server on the remote computer terminates the Internet connection informs the client that the session has expired and control of the keyboard, mouse and display returns to the local computer

The remote access by telnet has three significant reasons. It makes computation remote from the user. Instead of sending a data file or a message from one computer to another, remote access allows a program to accept input, process it and send back the result to the remote user. Secondly, once user logs in to the remote computer the user can execute any kind of program residing in the remote server. Finally users working in heterogeneous platforms telnetting may become a common interface for different machines.

Here's an example of a telnet session to osmania

```
$ telnet
telnet> toggle options
Will show option processing.
telnet> open osmania
Trying 202.54.70.200
Connected to linux osmania
Escape character is '^]'.
```

Summary

The Internet is not confined to just web browsing. It has several other services, which are equally popular.
With your web browser as your ship, you can navigate through an ocean of information on the web. Whichever Web browsers you have, spend some time getting to know it. Learn about some customizations that can save you time and make it easier for you to surf the Web.

When you have a narrow or obscure topic or idea for research or for personal enlightment use search engines. The popular search engine is google.com
A mailing list is a collection of names and addresses used by an individual or an organization to send material to multiple recipients.
A newsgroup is a repository, for messages posted from many users at different locations.
A chat room is an online forum where people can chat online
Electronic mail was originally designed to allow a pair of individuals to communicative via computer.

For sending large volumes of data reliably over the net **File Transfer Protocol (FTP)** is preferred instead.
The Telnet protocol is often thought of as simply providing a facility for remote logins to the computer via the Internet.

Short questions:

1). List the various services of the internet
2). What is the protocol of WWW?
3). How many ways one can have Internet access?
4). What is a bookmark?
5). What is URL?
6). HTTP stands for ……
7). What are search engines?
8). When FTP is used?
9). What is the purpose of telnet?
10). State the usage og mailing list

Long questions:

!). Describe the various services offered by Internet and discuss about each of them in detail.

2). State the sequence of events that take place when an user enters an URL in his browser.
3). List the various parameters of the browser which you can customize.
4). Elaborate the operation of email
5). Describe the FTP operation
6). Discuss in detail about telnet
7). Describe a real FTP session (you can use fictitious names)
8). Furnish Internet explorer’s short cut keys
9). List at least three FTP commands and explain about them.
10). Show the various elements of web navigation.

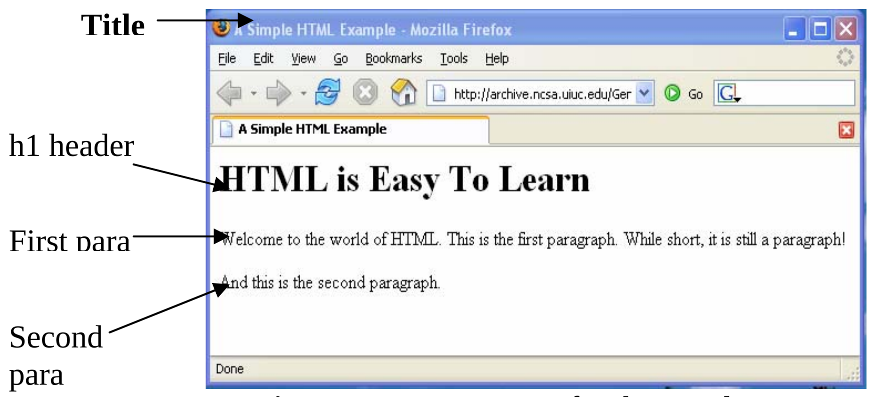
4. HTML

- HTML stands for **Hyper Text Markup Language**
- An HTML file is a text file containing small markup tags
- The markup tags tell the Web browser how to display the page
- An HTML file must have an **htm** or **html** file extension
- An HTML file can be created using a simple text editor
- HTML documents are text files made up of HTML elements.
- HTML elements are defined using HTML tags.
- HTML tags are surrounded by the two characters < and >
  The surrounding characters are called angle brackets
- HTML tags normally come in pairs like <b> and </b>
  The first tag in a pair is the start tag, the second tag is the end tag
  The text between the start and end tags is the element content
- HTML tags are not case sensitive, <b> means the same as <B>.
  However the World Wide Web Consortium (W3C) recommends lowercase tags in their HTML 4 recommendation, and XHTML (the next generation HTML) demands lowercase tags.

4.1 A Simple HTML example

Each HTML document consists of head and body tags. The head contains the title, and the body contains the actual text that is made up of paragraphs, lists, and other elements. Browsers expect specific information because they are programmed according to HTML specifications. The following is a simple HTML program.

```
<html>
<head>
<TITLE>A Simple HTML Example</TITLE>
</head>
<body>
<h1>HTML is Easy To Learn</h1>
<P>Welcome to the world of HTML.
This is the first paragraph. While short it is
still a paragraph!</P>
<P>And this is the second paragraph.</P>
</body>
</html>
```


Title
h1 header
First para
Second para
A Simple HTML Example - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://archive.ncsa.uiuc.edu/Ger
Go
A Simple HTML Example
Done

HTML is Easy To Learn

Welcome to the world of HTML. This is the first paragraph. While short, it is still a paragraph!

And this is the second paragraph.

Fig.4.1 Browser output for the simple program

4.2 Formatting

4.2.1 Text Formatting

Headings: <h$_k$> …</h$_k$> ('k' is a number from 1 to 6) Use heading tags for headings only. Don't use them for making text bold. There are separate commands for that purpose.

```
<h1>  ----text-----   </h1>        largest
<h2>  ---- text ----   </h2>          ↑
<h3>  ---- text ----   </h3>          |
<h4>  ----- text ---   </h4>          |
<h5>  ----- text ---   </h5>          |
<h6>  ------ text --   </h6>       smallest
```

<p> Paragraph: As earlier mentioned browser creates a paragraph. A blank line is added before and after the each paragraph.
You can center a paragraph in the browser output by including the **ALIGN=***alignment* attribute in your source html file.

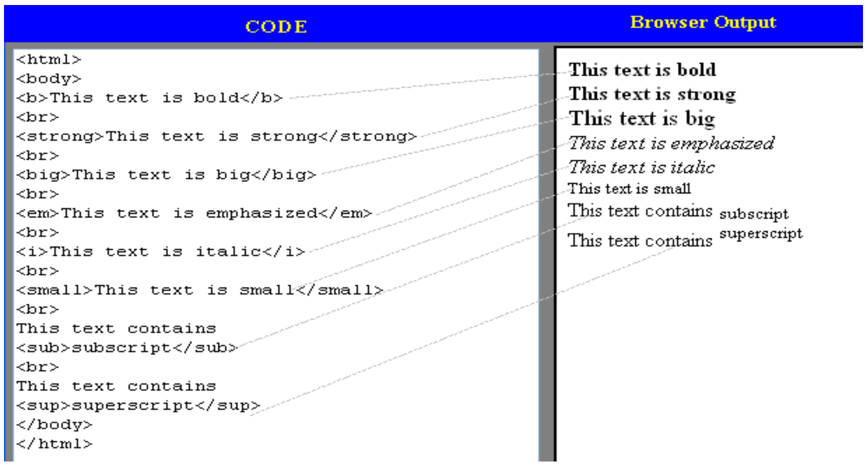
```
<P ALIGN=CENTER> This is a centered paragraph.</P>
```

<br>Line breaks: The browser simply ignores white spaces, new lines present in the body of the text of a paragraph. However by placing <br> command a new line is forced from that point in that paragraph (but not new paragraph!)

<!.. ..>Comments: For documentation purpose insert your comments in <!..>

<! ………..comments………..>

<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

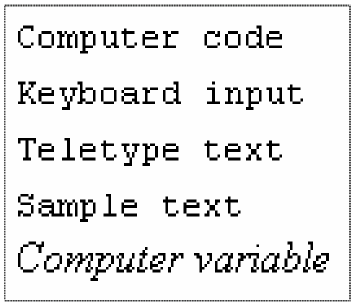


```
<html>
<body>
<b>This text is bold</b>
<br>
<strong>This text is strong</strong>
<br>
<big>This text is big</big>
<br>
<em>This text is emphasized</em>
<br>
<i>This text is italic</i>
<br>
<small>This text is small</small>
<br>
This text contains
<sub>subscript</sub>
<br>
This text contains
<sup>superscript</sup>
</body>
</html>
```

```
This text is bold
This text is strong
This text is big
This text is emphasized
This text is italic
This text is small
This text contains subscript
This text contains superscript
```

Fig.4.1.1 Computer output tags

<code>	Defines computer code text
<kbd>	Defines keyboard text
<samp>	Defines sample computer code
<tt>	Defines teletype text
<var>	Defines a variable
<pre>	Defines preformatted text



```
Computer code
Keyboard input
Teletype text
Sample text
Computer variable
```

Fig.4.1.2 Browser output for different computer tags

4.2.2 Citations, Quotations, and Definition Tags

<abbr> Defines an abbreviation.**<abbr>**
title="Hyper Text Markup Language">HTML</abbr>
<acronym> Defines an acronym. The title attribute is used to show the spelled-out version when holding the mouse pointer over the acronym or abbreviation
<address> Defines an address element
<bdo> Defines the text direction. bi-directional override (bdo)
<bdo dir="rtl">Hello World!</bdo> <! ..Right to left..>
Browser output = !dlroW olleH

<blockquote>	Defines a long quotation
<q>	Defines a short quotation
<cite>	Defines a citation
<dfn>	Defines a definition term

4.2.3 Character Entities

Few characters like the '< 'character, have a special meaning in HTML, and therefore cannot be used in the normal text. We have to use character entities in order to display these characters. They can also be called by **&#asci** number in decimal.

&nbsp non-breaking space &#160. Normally HTML will truncate spaces in your text. To add spaces to your text, use the &nbsp character entity.

&lt < less than; &#60;
&gt > greater than; &#62;
&amp & ampersand; &#38;
&quot " quotation mark; &#34;
&apos ' apostrophe
Hyper links

4.3 Hyperlinks

Hyperlinks are created using Anchor tags.
Anchor tag **a** with href attribute
<a href="http://www.apboi.ac.in/">Visit Board of Intermediate**</a>**

In the browser it would look like as a hyperlink to http://www.apboi.ac.in for that text – Visit Board of Intermediate -. Whenever user places his mouse pointer and clicks on that text, the web page with specified URL will be opened. If any sub folders are there in the specified URL, include them in the href definition with slashes. In such cases the browser makes more than one request. The first one for the main URL and the next one for the sub folder within the URL*.

Visit Board of Intermediate

4.4 URLs

A URL includes the type of resource being accessed (e.g., Web, gopher, FTP), the address of the server, and the location of the file. The syntax is:

scheme://*host.domain* [:*port*]/*path*/ *filename*

Where *scheme* is one of the

file	a file on your local system
ftp	a file on an anonymous FTP server
http	a file on a World Wide Web server
gopher	a file on a Gopher server
WAIS	a file on a WAIS server
news	a Usenet newsgroup
telnet	a connection to a Telnet-based service

The *port* number can generally be omitted.

4.4.1 Target attribute

The following code will open the URL in a new browser window

<a href="http://www.apboi.ac.in/" target =“_blank”>Visit Board of intermediate</a>

4.4.2 Name attribute

Using named anchors we can create links that can jump directly into a specific section on a page within the document

<a name=”go to second chapter>”4.2 Formatting</a>

4.5 Background Graphics

If you want to include a background, make sure your text can be read easily when displayed on top of the image.

Background images can be a texture (linen finished paper, for example) or an image of an object (a logo possibly). You create the background image as you do any image.

However you only have to create a small piece of the image. Using a feature called tiling, a browser takes the image and repeats it across and down to fill your browser window. In sum you generate one image, and the browser replicates it enough times to fill your window. This action is automatic when you use the background tag shown below.
The tag to include a background image is included in the **<BODY>** statement as an attribute:

<BODY BACKGROUND="*filename*.gif">
The *filename.gif* should be in current directory.

If the image source is on web specify its location along with its URL. Most of the browsers support gif and jpeg image files.

4.5.1 Background Color

You change the color of text, links, visited links, and active links (links that are currently being clicked on) using further attributes of the <BODY> tag. For example:

<BODY BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#9690CC">
This creates a window with a black background (BGCOLOR), white text (TEXT), and silvery hyperlinks (LINK).

The color numbers are in hexadecimal notation. In HTML each color is given an unique number. The six-digit number and letter combinations represent colors by giving their RGB (red, green, blue) value. The six digits are actually three two-digit numbers in sequence, representing the amount of red, green, or blue as a hexadecimal value in the range 00-FF. For example, 000000 is black (no color at all), FF0000 is bright red, 0000FF is bright blue, and FFFFFF is white (fully saturated with all three colors).

4.6 External Images, Sounds, and Animations

You may want to have an image open as a separate document when a user activates a link on either a word or a smaller, inline version of the image included in your document. This is called an external image, and it is useful if you do not wish to slow down the loading of the main document with large inline images.

To include a reference to an external image, enter:
   <A HREF="*MyImage.gif*">link anchor</A>

You can also use a smaller image as a link to a larger image. Enter:
   <A HREF="*LargerImage.gif*"><IMG SRC="*SmallImage.gif*"></A>

The reader sees the SmallImage.gif image on the browser and clicks on it to open the LargerImage.gif file.

Use the same syntax for links to external animations and sounds. The only difference is the file extension of the linked file. For example,

```
<A HREF="Titanic.mov">link anchor</A>
<!..specifies a link to a movie. ..>
```

Some common file types and their extensions are:

plain text	.txt
HTML document	.html
GIF image	.gif
TIFF image	.tiff
X Bitmap image	.xbm
JPEG image	.jpg or .jpeg
PostScript file	.ps
AIFF sound file	.aiff

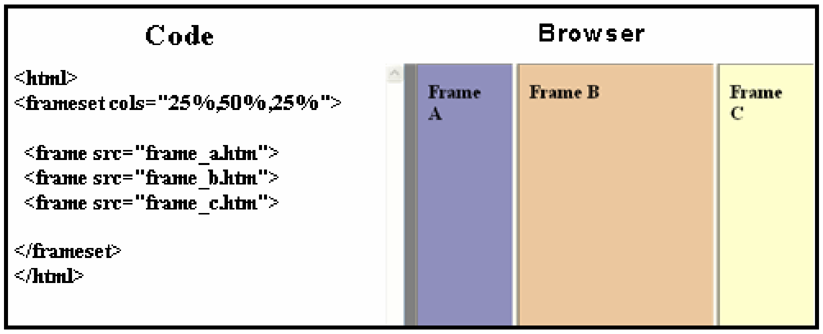
AU sound file	.au
WAV sound file	.wav
QuickTime movie	.mov
MPEG movie	.mpeg or .mpg

4.7 Frames

Using frames multiple html documents can be displayed in same browser window. Each individual html document is called frame and each frame is independent to each other.

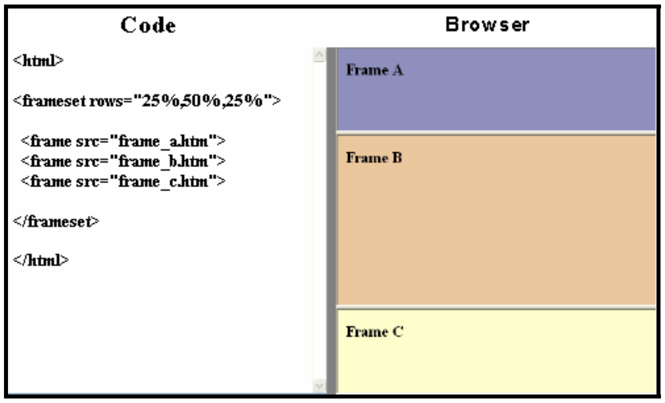
The <frame> tag defines what HTML document to put into each frame
The <frameset> tag defines how to divide the window into frames
Each frameset defines a set of rows or columns
The values of the rows/columns indicate the amount of screen area each row/column will occupy
The attribute src can be an URL/ or any valid html source file
The 100% frameset represents the total browser window area
<noframes> defines no frame section for the browsers which do not handle frames.

4.7.1 Vertical frame set



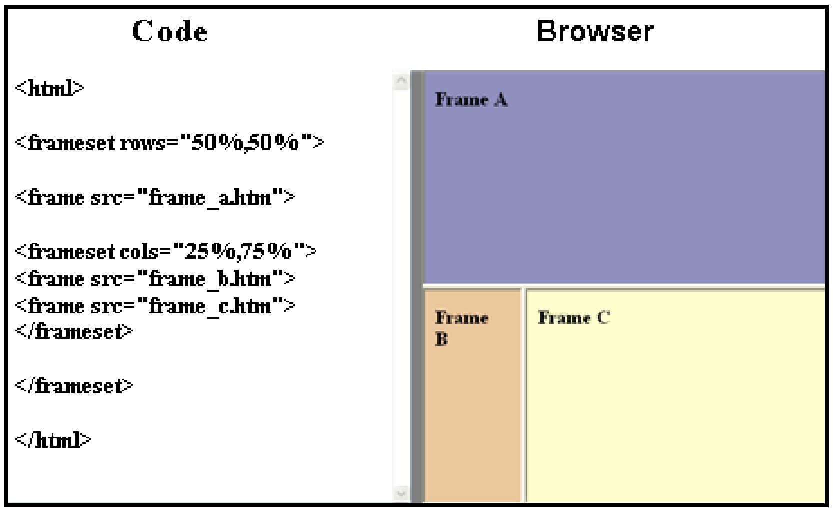
```
<html>
<frameset cols="25%,50%,25%">

  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">

</frameset>
</html>
```

4.7.2 Horizontal frame set



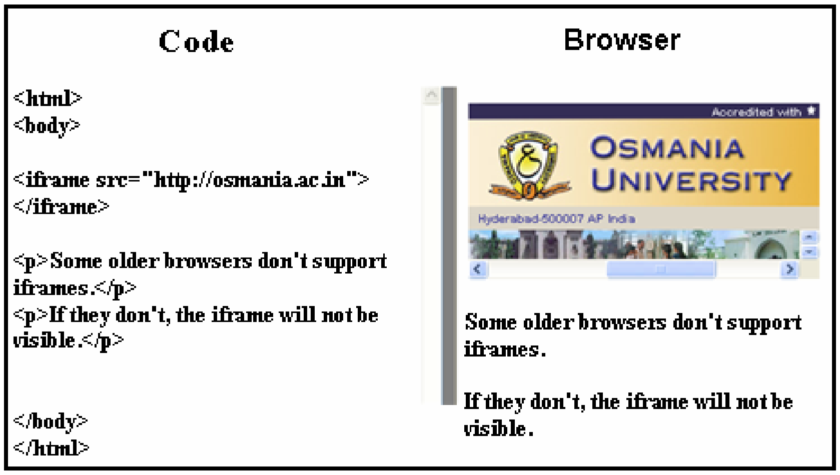
```
<html>

<frameset rows="25%,50%,25%">

  <frame src="frame_a.htm">
  <frame src="frame_b.htm">
  <frame src="frame_c.htm">

</frameset>

</html>
```

4.7.3 Mixed frame set



```
<html>

<frameset rows="50%,50%">

<frame src="frame_a.htm">

<frameset cols="25%,75%">
<frame src="frame_b.htm">
<frame src="frame_c.htm">
</frameset>

</frameset>

</html>
```

4.7.4 Inline frames



```
<html>
<body>

<iframe src="http://osmania.ac.in">
</iframe>

<p>Some older browsers don't support
iframes.</p>
<p>If they don't, the iframe will not be
visible.</p>


</body>
</html>
```

```
Some older browsers don't support
iframes.

If they don't, the iframe will not be
visible.
```

4.8 Tables

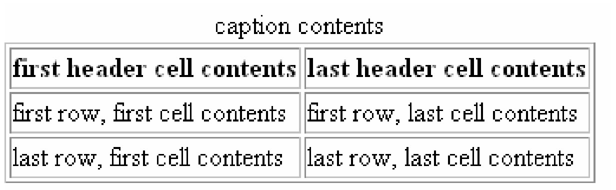
Tables are very useful for presentation of tabular information as well as a boon to creative HTML authors who use the table tags to present their regular Web pages.

A table has heads where you explain what the columns/rows include, rows for information, cells for each item. In the following table, the first column contains the header information, each row explains an HTML table tag, and each cell contains a paired tag or an explanation of the tag's function.

General Table Format:

```
<TABLE>
<!-- start of table definition -->
<CAPTION> caption contents </CAPTION>
<!-- caption definition -->

TR>
<!-- start of header row definition -->
 <TH> first header cell contents </TH>
  <TH> last header cell contents </TH>
</TR>
<!-- end of header row definition -->
```

```
<TR>
<!-- start of first row definition -->
   <TD> first row, first cell contents </TD>
   <TD> first row, last cell contents </TD>
</TR>
<!-- end of first row definition -->

<TR>
<!-- start of last row definition -->
   <TD> last row, first cell contents </TD>
   <TD> last row, last cell contents </TD>
</TR>
<!-- end of last row definition -->
</TABLE><!-- end of table definition -->
```



caption contents

first header cell contents	**last header cell contents**
first row, first cell contents	first row, last cell contents
last row, first cell contents	last row, last cell contents

Fig.4.8.1 Browser output of Table with Caption

Table Elements	
Element	Description
<TABLE> ... </TABLE>	Defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border.
<CAPTION> ... </CAPTION>	Defines the caption for the title of the table. The default position of the title is centered at the top of the table. The attribute ALIGN=BOTTOM can be used to position the caption below the table. NOTE: Any kind of markup tag can be used in the caption.
<TR> ... </TR>	Specifies a table row within a table. You may define default attributes for the entire row: ALIGN (LEFT, CENTER, RIGHT) and/or VALIGN (TOP, MIDDLE, BOTTOM). See Table Attributes at the end of this table for more information.
<TH> ... </TH>	defines a table header cell. By default the text in this cell is bold and centered. Table header cells may contain other attributes to determine the characteristics of the cell and/or its contents. See Table Attributes at the end of this table for more information.
<TD> ... </TD>	Defines a table data cell. By default the text in this cell is aligned left and centered vertically. Table data cells may contain other attributes to determine the characteristics of the cell and/or its contents. See Table Attributes at the end of this table for more information.

Table Attributes	
NOTE: Attributes defined within <TH> ... </TH> or <TD> ... </TD> cells override the default alignment set in a <TR> ... </TR>.	
Attribute	Description
ALIGN (LEFT, CENTER, RIGHT)	Horizontal alignment of a cell.
VALIGN (TOP, MIDDLE, BOTTOM)	Vertical alignment of a cell.
COLSPAN=*n*	The number (*n*) of columns a cell spans.
ROWSPAN=*n*	The number (*n*) of rows a cell spans.
NOWRAP	Turn off word wrapping within a cell.

The above code can be used as template for constructing any type of table adding new rows or cells as necessary.

The **<TABLE>** and **</TABLE>** tags must surround the entire table definition.
The first item inside the table is the **CAPTION**, which is optional.
Then you can have any number of rows defined by the **<TR>** and **</TR>** tags.
Within a row you can have any number of cells defined by the **<TD>**...**</TD>** or **<TH>**...**</TH>** tags. Each row of a table is, essentially, formatted independently of the rows above and below it.
There are some additional elements like <thead>,<tbody> and <tfoot> which are seldom used, because of bad browser support.

4.8.1 Tables for Non tabular Information

Some HTML authors use tables to present non-tabular information. For example, because links can be included in table cells, some authors use a table with no borders to create "one" image from separate images. Browsers that can display tables properly show the various images seamlessly, making the created image seem like an *image map* (one image with hyper linked quadrants).

Using table borders with images can create an impressive display as well. Experiment and see what you like.

Example: Try this

```
<html>
<body>

<table border="1">
<caption><b>Table with different attributes</b></caption>
<tr>
  <td>
   <p>This is a paragraph</p>
   <p>This is another paragraph</p>
  </td>
  <td bgcolor="cyan">This cell contains a table:
   <table border="9">
   <tr>
     <td>A</td>
     <td>B</td>
   </tr>
   <tr>
     <td>C</td>
     <td>D</td>
   </tr>
   </table>
  </td>
</tr>
<tr>
  <td bgcolor="red">
This cell contains a list
   <ul>
    <li>apples</li>
    <li>bananas</li>
    <li>pineapples</li>
   </ul>
  </td>
  <td bgcolor="yellow">
HELLO</td>
</tr>
</table></body></html>
```

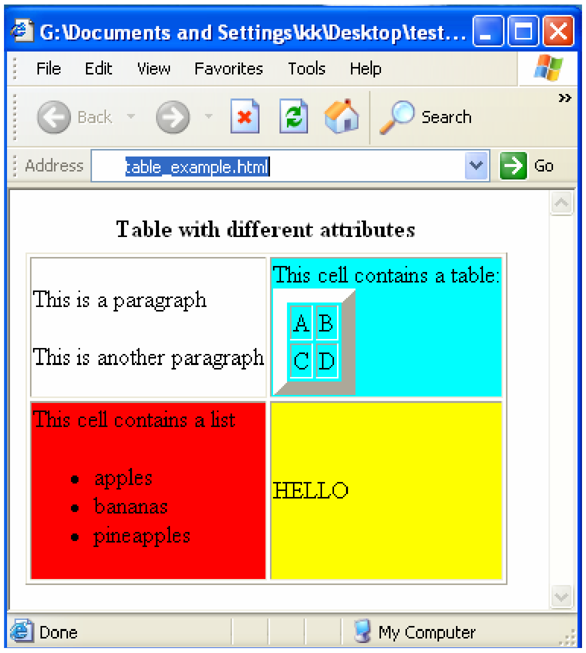

G:\Documents and Settings\kk\Desktop\test...
File Edit View Favorites Tools Help
Back
Search
Address table_example.html
Go
Done
My Computer

Table with different attributes

This is a paragraph This is another paragraph	This cell contains a table: A B C D
This cell contains a list • apples • bananas • pineapples	HELLO

Fig.4.8.2 Output for the program

4.9 Lists

Lists display the content in a compact tight format. Increases the comprehension of the presentation content. There are three types of lists.

Unordered lists
Ordered lists
Definition lists

4.9.1 Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles). An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

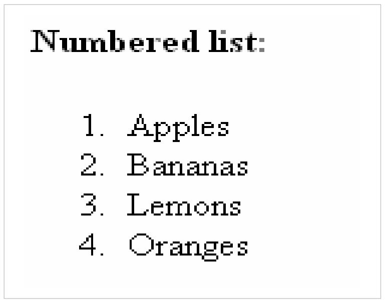
4.9.2 Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers. An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

4.9.3 Definition Lists

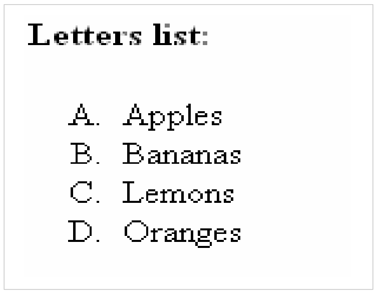
A definition list is not a list of items. This is a list of terms and explanation of the terms. A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag. Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc. They get properly indented. Try the following program: ypical outputs are shown in boxes

```
<html>
<body>
<h4>Numbered list:</h4>
<ol>
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```



```
Numbered list:

   1. Apples
   2. Bananas
   3. Lemons
   4. Oranges
```

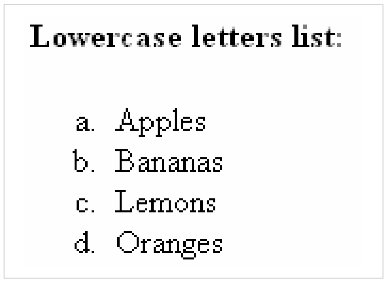
```
<h4>Letters list:</h4>
<ol type="A">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```



Letters list:

A. Apples
B. Bananas
C. Lemons
D. Oranges

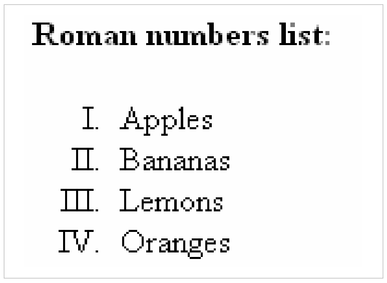
```
<h4>Lowercase letters list:</h4>
<ol type="a">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```



Lowercase letters list:

a. Apples
b. Bananas
c. Lemons
d. Oranges

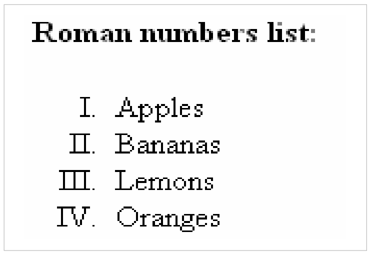
```
<h4>Roman numbers list:</h4>
<ol type="I">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```



Roman numbers list:

I. Apples
II. Bananas
III. Lemons
IV. Oranges

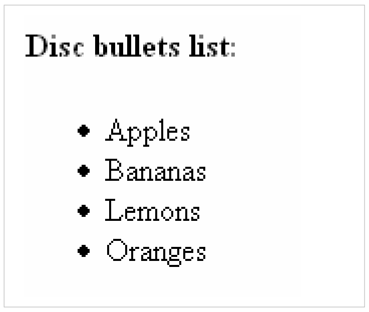
```
<h4>Lowercase Roman numbers list:</h4>
<ol type="i">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```



Roman numbers list:

I. Apples
II. Bananas
III. Lemons
IV. Oranges

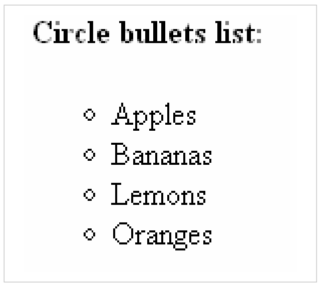
```
<h4>Disc bullets list:</h4>
<ul type="disc">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
```



Disc bullets list:

- Apples
- Bananas
- Lemons
- Oranges

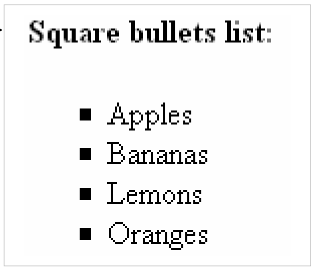
```
<h4>Circle bullets list:</h4>
<ul type="circle">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
```



Circle bullets list:

○ Apples
○ Bananas
○ Lemons
○ Oranges

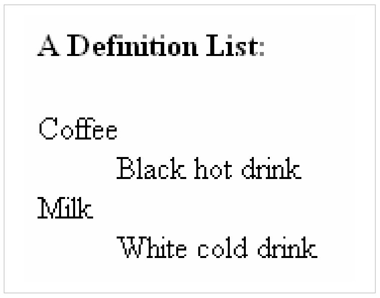
```
<h4>Square bullets list:</h4>
<ul type="square">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
```



Square bullets list:

■ Apples
■ Bananas
■ Lemons
■ Oranges

```
<h4>A Definition List:</h4>
<dl>
  <dt>Coffee</dt>
  <dd>Black hot drink</dd>
  <dt>Milk</dt>
  <dd>White cold drink</dd>
</dl>
```



A Definition List:

Coffee
    Black hot drink
Milk
    White cold drink

```
</body></html><!.. This program types of list elements…>
```

4.10 Forms

Forms can be used to gather information from users. To be able to use forms, the server on which your site is located must provide CGI capabilities. CGI is a system of files that process data received from forms. The popular CGI scripts are perl, php …(see CGI Chapter)

4.10.1 Form Attributes

There are two form attributes viz., **method** & **action**

There are two methods used by browsers to send information to a server. They are **Post** and **Get**. These are specified in code as:

- <form method="**POST**">
- <form method="**GET**">

Action tells the browser what CGI program to use and where it is located.
A form tag with both attributes would be coded as

<form **method**="POST" **action**="cgi-bin/demo_html40.pl">

The code tells the browser to use the post method, and to use the CGI program, demo_html40.pl, located in the cgi-bin (directory or folder) located on the server.

4.10.2 Input

To query the web or submit information to the web, Forms provide a very user-friendly interface. The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

The input tag and its attributes:

- Text, name:
  <input type="text" name="firstname">
- Radio, name, value:
  <input type="radio" name="sex" value="male"> Male
- Checkbox, name:
  <input type="checkbox" name="bike">I have a bike

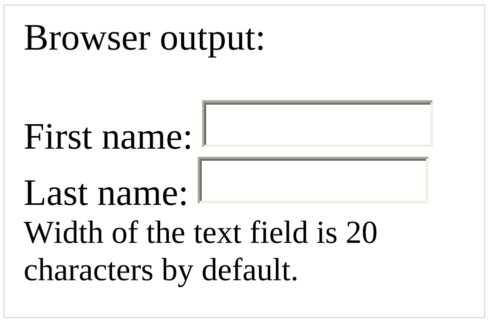
- Type, value:
  `<input type="submit" value="Submit">`
- Drop-down box   Drop-down box is a selectable list ;
  `<select name>`
  `<option  value>`
- Textarea
  `<textarea>defines a multi line text control</textarea>`

Examples:

- **Text, name Fields**

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name:
<input type="text" name="firstname">
<br>
Last name:
<input type="text" name="lastname">
</form>
```



Browser output:

First name:

Last name:

Width of the text field is 20 characters by default.

- **Radio Buttons**

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex"
 value="male"> Male
<br>
```

```
<input type="radio" name="sex"
 value="female"> Female
</form>
```

Browser output:

◉ Male

○ Female

Only one option is allowed

- **Checkboxes**

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
<input type="checkbox" name="bike">
I have a bike
<br>
<input type="checkbox" name="car">
I have a car
</form>
```

Browser output:

☐ I have a bike

☑ I have a car

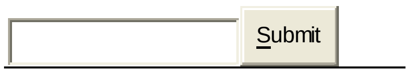
The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to the web server. The file defined in the action attribute usually does something with the received input. If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

```
<form name="input"
action="html_form_action.asp"
```

```
method="get">
Username:
<input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

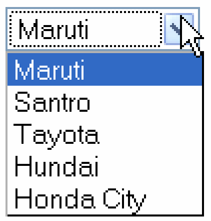
Browser output:
Username:



- **Drop-down box**

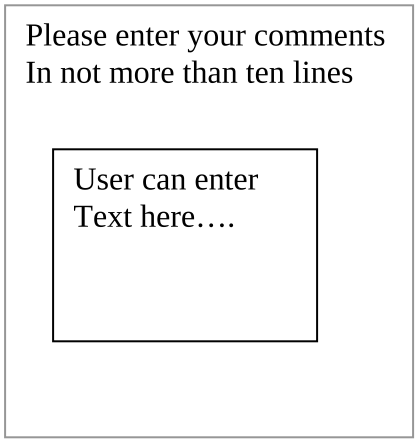
```
<html>
<body>

<form>
<select name="cars">
<option value="Maruti">Maruti
<option value="Santro">Santro
<option value="Toyota">Tayota
<option value="Hundai">Hundai
<option value="Honda City">Honda City

</select>
</form></body></html>
```

Browser output:



- **Textarea** (a multi line text control)

```
<html>
<body>
 <p>
 Please enter your comments
In not more than ten lines
</p>
<textarea rows="10" cols="30">
</textarea>
 </body>
 </html>
```



4.11 Image maps

Imagemaps (client-side, server-side) have been one of the major innovations in web interactivity. They are a very popular navigation aid. These clickable graphics can be used to help create a site identity and give visual impact. Unfortunately they also can add considerably to download time.

Often imagemaps are used so that sites and their designers can show off some 'cool' graphics - Navigating such sites is a pain - having waited a considerable time for the image to download, the first hurdle to overcome is often figuring out what they mean. Imagemaps allow users to click on a particular spot in an image to retrieve another HTML document or to run a specified script Client-Side Imagemaps

With client-side image maps, the MAP that relates parts of the image to different URLs is stored in the current file. This saves a round trip to the server, and should present documents to you faster. Since the MAP information is stored in the document you are viewing, the destination URLs can be displayed in the status area as you pass the mouse over the image map.

<MAP NAME="*name*">
<AREA [SHAPE="*shape*"] COORDS="*x,y,...*" [HREF="*reference*"] [NOHREF]>
</MAP>

- The **name**

specifies the name of the map so that it can be referenced by an IMG element.

- The **shape**

gives the shape of this area. Currently the only shape defined is "RECT".

- The **COORDS** attribute

gives the coordinates of the shape, using image pixels as the units. For a rectangle, the coordinates are given as "*left,top,right,bottom*".

- The **NOHREF** attribute

indicates that clicks in this region should perform no action. An HREF attribute specifies where a click in that area should lead.

4.11.1 The USEMAP attribute in an IMG element

Indicates that it is a client-side image map. It can be combined with the ISMAP attribute to indicate that the image can be processed as either a client-side or server-side image map. The argument specifies which map to use with the image, *imagename*. The proposed spec on image maps allows for the option to include ALT text in the IMG element.
<IMG SRC="*imagename*" [USEMAP="# *name*"] [ALT="*text*"]>

Serverside image maps:

```
INPUT <A      HREF  = "banner.map">
<IMG  SRC    = "banner.gif"
          ALT     = "The Web Developer's Virtual Library"
          ISMAP></A>
```

If the **ISMAP** attribute is present in the **IMG** tag, and the image tag is within an anchor, the image will become a "clickable image". The pixel coordinates of the cursor will be appended to the URL (after a "?") specified in the anchor if the user clicks within the ismap image. You'll need to specify a 'map' file, similar to the following (server-dependent):

```
default http://xyz.com/
rect    http://Xyz.com/Location/Maps/              0, 0    459,29
rect    http://Xyz.com/Gallery/                    0,30     67,59
rect    http://Xyz.com/Authoring/Graphics/        60,30   139,59
rect    http://Xyz.com/Vlib/Internet/Jobs.html   140,30   179,59
rect    http://Xyz.com/Reference/Library/        180,30   239,59
rect    http://Xyz.com/URLy.html                 240,30   285,59
rect    http://Xyz.com/Location/Search/          286,30   341,59
rect    http://Xyz.com/avcs/Stats/Top/           342,30   397,59
rect    http://Xyz.com/avcs/HTML/Tutorial/       397,30   479,59
```

4.12 The Common Gateway Interface (CGI)

CGI is not a language. It's a simple protocol that can be used to communicate between Web forms and your program. A CGI script can be written in any language that can read STDIN(e.g. keyboard), write to STDOUT(e.g. Monitor), and read environment variables, i.e. virtually any programming language, including C, Perl, or even shell scripting can be written for CGI.

4.12.1 Server-side Processing

The Common Gateway Interface, or CGI, permits interactivity between a client and a host operating system through the World Wide Web via the Hyper Text Transfer Protocol (HTTP). It's a standard for external gateway programs to interface with information servers, such as HTTP or Web servers. A plain HTML document that the Web server delivers is static, which means it doesn't change. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information - perhaps a weather reading, or the latest results from a database query. CGI allows someone visiting your Web site to run a program on your machine that performs a specified task.

Gateways are programs, which handle information requests and return the appropriate document or generate a document on the fly. Your server can serve information, which is not in a form readable by the client (e.g. an SQL database), and act as a mediator between the two to produce something, which clients can use.

Gateways can be used for a variety of purposes, the most common being the handling of FORM requests for HTTP. An HTTP server is often used as a gateway to a legacy information system; for example, an existing body of documents or an existing database application. The Common Gateway Interface is a convention between HTTP server implementers about how to integrate such gateway scripts and programs.

Gateway programs, or scripts, are executable programs, which can run by themselves. They have been made external programs in order to allow them to run under various (possibly very different) information servers interchangeably.

Gateways conforming to this specification can be written in any language, which produces an executable file. Some of the more popular languages to use include: C or C++, Perl, PHP and many others. It doesn't matter what language the program is written in, as long as you have the permission and resources to run it on your machine and the program is written correctly.

Here is a simple example demonstrating the Common Gateway Interface. This example uses the Perl language because of its portability and relative ease of use. When we explain operating system commands we will generally speak UNIX. Remember UNIX is Case sensitive!

Some servers allow your CGI programs to be anywhere in your web directories, so long as the file name ends in ".cgi". Others require you to put them only in the "/cgi-bin" directory. Check with your system administrator. Now, create a file called Hello.cgi: Using chmod command make this file executable.

```
#!/usr/bin/perl

$t      = "Hello World!";
print   <<EOT;
Content-type: text/html

<Title> $t </Title>
<H1>  $t </H1>
EOT
```

The first line must contain the path to your Perl interpreter.
Scalar variables names in Perl start with the "$" character. "$t" contains some text to be used later.
The print statement prints everything following until the "EOT". Text printed to "standard output" goes to the server and thence to the browser.

The first line printed is an HTTP header to tell the browser that an HTML file is coming. (HTTP header lines must always be separated by a blank line from actual data).

The data sent is a valid HTML document. The Perl interpreter replaces "$t" with "Hello World!".

Create a link to it like this:

<a href="Hello.cgi">Hello.cgi</a>

Click on Hello.cgi file, and if all's well, the script should respond with "Hello World!"

4.12.2 Structure of a CGI Script

Here's the typical sequence of steps for a CGI script:

- Read the user's form input.
- Do what you want with the data.
- Write the HTML response to STDOUT.

When the user submits the form, your script receives the form data as a set of name-value pairs. The names are what you defined in the INPUT tags (or SELECT or TEXTAREA tags), and the values are whatever the user typed in or selected. (Users can also submit files with forms).

This set of name-value pairs is given to you as one long string, which you need to parse. It's not very complicated, and there are plenty of existing routines to do it for you.

the long string is in one of these two formats:
"name1=value1&name2=value2&name3=value3"
"name1=value1;name2=value2;name3=value3"

So just split on the ampersands or semicolons, then on the equal signs. Then, do two more things to each name and value:
Convert all "+" characters to spaces, and Convert all "%xx" sequences to the single character whose ascii value is "xx", in hex. For example, convert "%3d" to "=".

This is needed because the original long string is URL-encoded, to allow for equal signs, ampersands, and so forth in the user's input. So where do you get the long string? That depends on the HTTP method the form was submitted with: **Get /Post**

For GET submissions, it's in the environment variable QUERY_STRING.
For POST submissions, read it from STDIN. The exact number of bytes to read is in the environment variable CONTENT_LENGTH.
(POST is more general-purpose, but GET is fine for small forms.)

Sending the Response Back to the User

First, write the line Content-type: text/html
plus another blank line, to STDOUT. After that, write your HTML response page to STDOUT, and it will be sent to the user when your script is done. That's all there is to it.

Yes, you're generating HTML code on the fly. It's not hard; it's actually pretty straightforward. HTML was designed to be simple enough to generate this way. Hello world in another way

```
#include <stdio.h>
void main() {

   printf("Content-type: text/html\n\n") ;
   /** Print the HTML response page to STDOUT. **/
   printf("<html>\n") ;
   printf("<head><title>CGI Output</title></head>\n") ;
   printf("<body>\n") ;
   printf("<h1>Hello, world.</h1>\n") ;
   printf("</body>\n") ;
   printf("</html>\n") ;

   exit(0) ;
}
```

Perl example:

```
# Print the HTML response page to STDOUT
print <<EOF ;
<html>
<head><title>CGI Results</title></head>
<body>
<h1>Hello, world.</h1>
</body>
</html>
EOF
exit ;
```

4.12.3 CGI for mail

One of the most common uses of a CGI script is to send mail form data to an email address. Upon completion the user will receive a feedback indicating that mail has been successfully delivered!

4.12.4 Security

A CGI script is a program that anyone in the world can run *on your machine*. So accordingly be watchful about security lapses in the program code. Never trust the user data input. Don't put user data in a shell command without verifying the data carefully,

It's easy for a hacker to send *any* form variables to your script, with any values (even non-printable characters). Your security shouldn't rely on fields having certain values, or even existing or not existing.

4.13 Cascading Style Sheets (CSS)

CSS is a standard for formatting Web pages that goes well beyond the limitations of HTML. Promulgated by the World Wide Web Consortium (W3C), the Internet's standards body, CSS extends HTML with more than 70 style properties that can be applied to HTML tags. With CSS, Web developers have at their disposal a wealth of additional formatting options for color, spacing, positioning, borders, margins, cursors and much, much more.

4.13.1 Adding CSS To HTML Documents

HTML gives you the developer a certain level of control over the formatting of a document. You can set headings <h1>, <h2>...etc), make text bold <b> or italic <i>, define lists (<ul>, <ol>), and so forth. However, this level of control is fairly limited. For example, developers have no control over the absolute positioning of items on the page, and are limited in their ability to control size, spacing, and color of page elements. Seeking to surpass these limitations, developers have resorted to workarounds such as converting text to graphics, creating complicated table layouts, using images for white space control, and using proprietary HTML extension and add-ons.

CSS shatters the HTML barrier by putting at the developer's disposal a set of additional properties specifically geared towards page formatting and layout. These properties are applied to the document without modifying the underlying HTML. Browsers that are not CSS-compliant will still see the page in its unaltered HTML state, while browsers that support CSS will see the page in all its CSS-enhanced presentation.

There are two steps to adding CSS styles to an HTML document:

Declaring the styles and applying the styles to HTML elements: For example, "I want some blue, bold, italic text" would be a simple declaration
OR I want my entire document subheading to be blue, bold, and italic would be an application of the style.

Unfortunately, we have to do a bit more than simply utter a few statements in front of the monitor -- at least until voice recognition and HTML editing technologies get more sophisticated. We need to understand the syntax of declaring CSS styles and the different ways in which we can apply these styles to our HTML documents.

You can add CSS properties to your documents in four ways:

Using inline styles
Using an embedded style sheet
Using a linked style sheet
Using an imported style sheet

4.13.1 Using Inline Styles

To use an inline style, you add a <STYLE> attribute to a specific instance of an HTML element. The following syntax:
 <b style="color: #FF0000">Color Me RED</b>.
Which would render like this: **Color Me RED** In red color

An inline style may be applied to any HTML element and modifies only the specific instance (occurrence in the document) of the element to which you apply it. In the example above, only that instance of <b> would be RED-- the rest of the <b> tags in the document would be unaffected.

Using lots of inline styles to format a document allows for great precision, but very tedious to code. If you have a lot of styles, the inline style method can also result in a fair amount of unnecessary coding. Inline styles are also somewhat difficult to maintain, because the CSS properties are scattered around many pages in your web site.

4.13.2 Using Embedded Style Sheets

To use an embedded style sheet, you define a style block (delimited by the <style type="text/css"> and </style> tags), which should be placed in the HEAD section of the document. This block consists of a set of style rules, where each rule defines a style for an HTML element or group of elements. A style rule has two parts:

- A selector that identifies a HTML element or group of elements.
- A declaration of the style properties to be applied to that selector

The generic syntax for a style rule is as follows:
selector {property: value; property: value;} When using embedded style sheets it is best to surround the selections and properties with the comment tags. This keeps older browsers from reading them and displaying your style sheet coding.

4.13.3 Using Linked Style Sheets

You can keep your style sheets in a separate file and link to ir from a document or set of documents, using a <link> tag in the HEAD section of the linking document, as follows:

<LINK REL="stylesheet" TYPE="text/css" HREF="stylesheets.css">.

The linked style sheet (stylesheet.css) consists of a set of style rules, exactly like an embedded style sheet, except that the style rules are not enclosed in <style type="text/css"></style> and comment <!-- --> tags. Linking to an external sheet allows the developer to apply a set of rules across a group of HTML documents, thus extending the benefits of embedded style sheets to a set of pages.

4.13.4 Using Imported Style Sheets

An external style sheet may also be imported into a document by using the @import property in a style sheet: @import: url(stylesheets.css). The @import tag should appear at the *beginning* of a style block or on a linked sheet, before any declarations. Rules in imported style sheets are applied *before* other rules defined for the containing style sheet, putting them at the bottom of the "pecking order' of the imported sheet.

4.13.5 CLASS as Selector

If you expect to have formatting variations for different instances of a single element, or you would like to have different elements share the same format, using a class name as a selector is a good approach. This is often referred to as "sub-classing" an element. CLASS is an HTML attribute that has no display characteristics and that you can apply to any element, like this: <b class="clsRed">Classy, red, and bold</b>.

The *style* rule for *clsRed* could be declared as follows:

```
<style type="text/css">
<!--
.clsRed {color:red;}
-->
</style>
```

Note that the selector begins with a period (.), which is the required syntax for class names as selectors. If we add the above rule to the style sheet, every element to which we assign a class name of *clsRed* will have red text. If an element doesn't have this class name, even it it is of the same type as another element that does, it will not have this style applied result –
Classy, red and bold displayed in red color

After knowing how to apply cascading style sheet elements, lets only use the embedded style sheets in this unit. That means you will use CSS for each page of your site. Later on you may want to take all those embedded tags and turn them into one complete LINKED style sheet for better control of your site.

4.13.6 Cascading Style Sheet Tags

The list below is some of the more commonly used CSS tags. (For a complete listing visit ttp://msdn.microsoft.com/workshop/author/css/css-ie4.asp)

font-family (enter name of font, if more than one, separate by commas)
font-style (italic or normal)
font-size (small, smaller, large, larger, pints, em)
font-weight (points, em, percentage of default)
color (#hex code ex: #FEDCBA98)
letter-spacing (places white space between letters)
text-decoration (used to take the underline out of links, overline, line-through, or none)
text-align (used to justify, right, left or center text)
text-indent (moves the first letter of each paragraph in x amount of pixels)
line-height (places space between the lines of text)
margin-right (moves the text and graphics to the right
margin-left (moves the text and graphics to the left)
…………See the power!

Here is how a paragraph using all of the CSS tags that are listed above would look. This embedded style coding would be placed in the HEAD section of the page that you are embedding this style.

```
<style type="text/css">
<!--
p.neat {font-family: verdana, arial, helvetica; font-style: normal;
 font-weight: normal; color: #000000; font-weight: bold;
letter-spacing: 5px; text-decoration: line-through;
 text-align: justify; text-indent: 25px; line-height: 25px;
margin-right: 65px; margin-left: 65px}
-->
</style>
```

Thus Web browser merges the contents of your Web documents (in an HTML file) with the specification (in a CSS file) of how the HTML elements should look. The result in the browser window is your content, but altered in appearance by the style sheet.

4.13.7 Style Sheet Use for Information Producers

Because any HTML document can link to a style sheet, a style sheet gives you a single place to specify the appearance of an entire set of Web pages. Thus, you can use style sheets to efficiently create a consistent look and feel for your Web pages and change this look and feel quickly and easily.

For example, many HTML writers like to indent the first line of each paragraph by five spaces. To do this, they would place the &nbsp; entity five times at the start of each paragraph. This worked fine for pages generated by HTML editors, or even on a small scale for hand-prepared HTML pages.

But what if the web designer decided that the paragraphs should not have the first line of each paragraph indented? It would be a laborious process to have to go through all the pages of the web to remove the &nbsp; entities. Instead, you could put this line in your style sheet to indent every paragraph in your document by 10 pixels:

```
P {text-indent: 10px;}
```

4.13.8 Style Sheet Use for Consumers

The people who use your Web pages will need to have the latest versions of the popular Internet Explorer (3.0 or above) or Netscape Navigator (4.0 or above) in order to see the style effects. If your users don't have these style-enabled browsers, they won't see all the effects you create.

Therefore, in using style sheets, like many other kinds of new HTML elements that have come before, you will need to be sensitive to users who do not have the most current Web browser software. With a little common sense and creativity, however, you'll find that you can still meet the needs of users who cannot use the style sheets.

In fact, by using style sheets, you can avoid much of the kind of syntax gymnastics that HTML writers have had to put into their pages to make things "look right." The result is that your HTML pages can be leaner and cleaner. You can separate style from substance. Instead of diddling with your Web pages to make them look a certain way, you can focus more on what your Web pages *mean*.

With style sheets, you have the ability to fiddle with the appearance of every HTML element in your document. For aesthetic reasons, resist this urge.

Name your classes after the meaning they serve, not their appearance. Think carefully about how you name style sheets. You can reference several style sheets in the same document, so you can create a set of modular style sheets that can be used by different collections of documents.

Over a period of time you will Create Your Own Style!

Study the following example and see how CSS adds to the content of html

It is a LINK element in the head of the document like this
<LINK rel="stylesheet" href="mystyle.css" type="text/css"> merges stylesheet with html.

The CSS file in this example alters the appearance of every H1 element in HTML files that reference it. The H1 elements appear in green. The font is changed to "rockwell," but if that is not available on the user's browser software, some other font of the "serif" family will be used.

The CSS file in this example creates two *classes* of the P(paragraph) element: the warning class and the danger class. Note that the actual class name is "warning," not "P.warning." The "P.warning" syntax is used in the CSS file to identify the element (P) and the class (warning). In your HTML document, all you need to do is use a P element with the class attribute set to "warning," like this: <P class="warning">This is a warning</P>

A CSS file "mystyle.css" contains this	An HTML file "hellostyle.html" contains this	Displayed in a Web browser, hellostyle.html looks something like this:

H1 { font-family: rockwell, serif; color: green; }
 	
P.danger { color: red; font-weight: bold; text-decoration: underline; }
 	
P.warning { color: purple; font-weight: bold; }
	<HTML> <HEAD>
	<TITLE> Style Sheet Demonstration </TITLE>
	<LINK rel="stylesheet" href="mystyle.css" type="text/css">
	</HEAD>
	<BODY>
	<H1> Our Exciting Story </H1>
	<P> We started to dock at the station before we realized what was happening. </P>
	<P class="warning"> Alien approaching! </P>
	<P> The capsule on the weird craft started emitting a beam of some sort. </P>
	<P class="danger"> Destruction imminent! </P>
	</BODY> </HTML>
		Our Exciting Story
		
		We started to dock at the station before we realized what was happening.
		Alien approaching!
		The capsule on the weird craft started emitting a beam of some sort.
		Destruction imminent!

I). A html example using back ground color

```
<html>
<head>
<title>My first program</title></head>
<body bgcolor="yellow">
<h2>Look: Colored Background!</h2>
The content of the body element is displayed in your browser.
</body>
</html>
```

II). HTML code headings and horizontal rule

```
<html>
<head>
<title>Headings & Horizontal rule</title></head>
<body>
<hr>
<h1>This is heading 1</h1>
<hr size=19 >
<h2>This is heading 2</h2>
<hr size=19 NOSHADE >
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
Don't use heading tags for the
text to display in bold or something like that
</body>
</html>
```

III). Image file as background. If the image is smaller than browser screen then the image Tiles.

```
<html>
<head>
<title>Page with background image</title></head>

<body background="background.jpg">
<!-- This gif file should be in the current directory of your system-->
<h1> A background image!</h1>

<p>gif and jpg files can be used as HTML backgrounds.</p>
```

```
<p>If the image is smaller than the page, the image will repeat (tile)itself
</p>
</body>
</html>
```

IV). Various text manipulation tags

```
<html>
<head>
<title>Text Formatting</title></head>

<body>
<b>This text is bold</b>
<br>
<strong>
This text is strong
</strong>
<br>
<big>
This text is big
</big>
<br>
<em>
This text is emphasized
</em>
<br>
<i>
This text is italic
</i>
<br>
<small>
This text is small
</small>
<br>
This text contains
<sub>
subscript
</sub>
<br>
This text contains
<sup>
```

```
superscript
</sup>
<hr>
<pre>
This is
preformatted text.
It preserves      both spaces
and line breaks.
</pre>
<hr>
<p>The pre tag is good for displaying computer code:</p>
<pre>
for i = 1 to 10
     print i
next i
</pre>
<hr>
</body>
</html>
```

V). Program with hyper links

```
<html>
<body>

<p>
<a href="http://www.boiap.ac.in">
Visit BOI</a> is a link to BOARD OF INTERMEDIATE on the web
</p>

<p>
<a href="http://www.osmania.ac.in">
This text</a> is a link to Osmania University on the World Wide Web.
</p>
<hr>
<p>
You can also use an image as a link:
<a href="toppage.htm">
<img border="0" src="buttonnext.gif" width="65" height="38"></a>
<!-- The gif image should be available on your system -->
</p>
</body></html>
```

VI). Frameset tags

Before writing code for mixed frameset, create three different html files frame_A, frame_B and frame_C with code depicted in the following table and place them all in the same directory

frame_A.html	**frame_b.html**	**frame_c.html**
<html> <body bgcolor="yellow"> <h2>Frame_A</h2> </body> </html>	<html> <body bgcolor="red"> <h2>Frame_B</h2> </body> </html>	<html> <body bgcolor="green"> <h2>Frame_C</h2> </body> </html>

Code for mixed frameset

```
<html>
<head>
<title>Mixed frameset Example</title></head>

<frameset rows="50%,50%">
        <frame src="frame_a.htm">
                <frameset cols="25%,75%">
<frame src="frame_b.htm">
<frame src="frame_c.htm">
</frameset>
</frameset>
</html>
```

VII). Inline images

```
<html>
<body>

<iframe src="http://yahoo.com"></iframe>

<p>Few browsers may not support iframes.</p>
<p> iframe will not be visible for such browser.</p>


</body></html>
```

VIII). Table example

```
<html>
<body>

<p>
Each table starts with a table tag.
Each table row starts with a tr tag.
Each table data starts with a td tag.
</p>

<h4>One column:</h4>
<table border="1">
<tr>
  <td>1000</td>
</tr>
</table>

<h4>One row and three columns:</h4>
<table border="1">
<tr>
  <td>ABC</td>
  <td>DEF</td>
  <td>GHI</td>
</tr>
</table>

<h4>Two rows and three columns:</h4>
<table border="1">
<tr>
  <td>XXXX</td>
  <td>YYYY</td>
  <td>ZZZZ</td>
</tr>
<tr>
  <td>4000</td>
  <td>5000</td>
  <td>6000</td>
</tr>
</table>
</body></html>
```

IX). Table with different tags

```
<html>
<body>

<table border="1">
<tr>
  <td>
   <p>This is a paragraph</p>
   <p><i>Another paragraph!</i></p>
  </td>
  <td>A table inside a table:
   <table border="1">
   <tr>
     <td>AAA</td>
     <td>BBB</td>
   </tr>
   <tr>
     <td>CCC</td>
     <td>DDD</td>
   </tr>
   </table>
  </td>
</tr>
<tr>
  <td>Differentiate
   <ul>
    <li>Frames</li>
    <li>Tables</li>
    <li>Iframes</li>
   </ul>
  </td>
  <td><b>HELLO WORLD</b></td>
</tr>
</table>

</body>
</html>
```

X). Simple forms

```
<html>
<body>

<form>
Username:
<input type="text" name="user">
<br>
Password:
<input type="password" name="password">
</form>
<p>
See the difference between text
and password forms
</p>
</body>
</html>
```

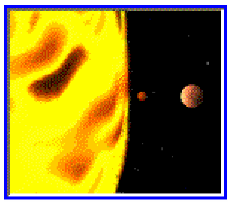
XI). Forms with mailto option

```
<html>
<body>
<form action="MAILTO:xyz@yahoo.com" method="post"
enctype="text/plain">

<h3>This form sends an e-mail to W3Schools.</h3>
Name:<br>
<input type="text" name="name"
value="yourname" size="20">
<br>
Mail:<br>
<input type="text" name="mail"
value="yourmail" size="20">
<br>
Comment:<br>
<input type="text" name="comment"
value="yourcomment" size="40">
<br><br>
<input type="submit" value="Send">
<input type="reset" value="Reset">
```

```
</form>
</body>
</html>
```

XII). Image maps with clickable area

```
<html>
<body>

<p>
Click on one of the planets to watch it closer:
</p>

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

<map id="planetmap" name="planetmap">

<area shape="rect"
coords="0,0,82,126"
alt="Sun"
href="sun.htm">

<area shape="circle"
coords="90,58,3"
alt="Mercury"
href="mercur.htm">

<area shape="circle"
coords="124,58,8"
alt="Venus"
href="venus.htm">

</map>

</body>
</html>
```


planet.gif

Summary:

HTML (Hyper text markup language) is the language used to create and develop web pages. The HTML language is interpreted by a browser. The HTML documents are also called web pages. HTML is a set of special codes that can be embedded in text to add formatting and linking information in web pages.

HTML commands broadly categorized into basic tags, text formatting, hyper links, frames, tables, lists, forms, images, graphics, scripts, and cascade style sheets.

CGI is not a language. It's a simple protocol that can be used to communicate between Web forms and your program. A CGI script can be written in any language including C, Perl, or even shell scripting can be written for CGI.

Cascaded style sheets (CSS) shatters the HTML barrier by putting at the developer's disposal a set of additional properties specifically geared towards page formatting and layout.

Short questions:

1). HTML stands for ……
2). CSS is acronym for ....
3). Which tag is used for setting title of the page?
4). How comments are introduces in HTML page?
5). To create bullets which command is used?
6). HTML color format can be represented by …..
7). Which tag is useful for search engines?
8). By using …… command you can insert blank places in the text.
9). State the popular language for CGI script
10). State one practical application of imagemaps.

Long questions:

1). Describe the basic structure of HTML document.
2). Explain the elements used in text formatting.
3). How to create html hyper links? – site examples
4). What are frames? How they improve the web appearance
5). How tables are created?-site examples
6). Describe images in HTML documents

7). List the various sound and video formats used in HTML
8). Explain various form controls
9). What are CSS? How they improve presentation of web page?
10). Describe the structure of CGI script

5.1 Java Server Pages (JSP)

Java Server Pages (JSP) technology provides an easy way to create dynamic web pages and simplify the task of building web applications that work with a wide variety of web servers, application servers, browsers and development tools. JSP was developed by Sun Microsystems to allow server side development. JSP files are HTML files with special Tags containing Java source code that provide the dynamic content. These were known as CGI server side applications. ASP was developed by Microsoft to allow HTML developers to easily provide dynamic content supported as standard by Microsoft's free Web Server, Internet Information Server (IIS). JSP is the equivalent from Sun Microsystems, a comparison of ASP.

JSP and ASP are fairly similar in the functionality that they provide. JSP may have slightly higher learning curve. Both allow embedded code in an HTML page, session variables and database access and manipulation. Whereas ASP is mostly found on Microsoft platforms i.e.Windows NT, JSP can operate on any platform that conforms to the J2EE specification. JSP allow component reuse by using Java beans and EJBs. ASP provides the use of COM / ActiveX controls.

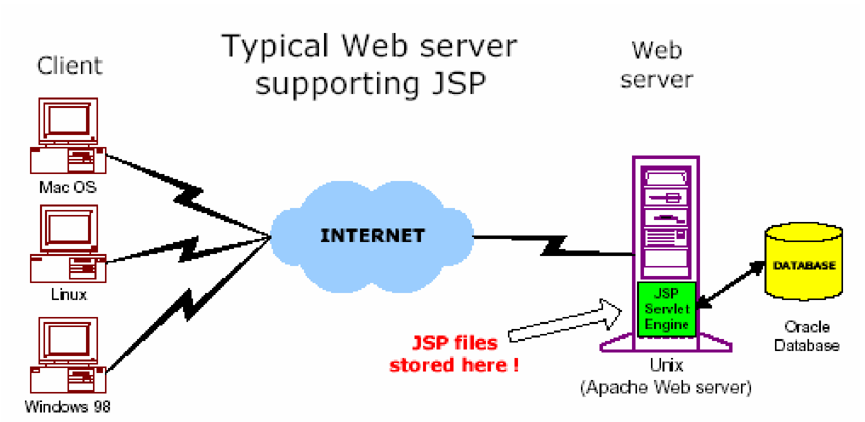


Fig.5.1.1 A web server that supports JSP files. Notice that the web server also is connected to a database.

JSP source code runs on the web server in the JSP Servlet Engine. The JSP Servlet engine dynamically generates the required HTML code and sends the HTML output to the client's web browser.

JSP is easy to learn and allows developers to quickly produce web sites and applications in an open and standard way. JSP is based on Java, an object oriented language. JSP offers a robust platform for web development.

Main reasons to use JSP:

- Multi platform
- Component reuse by using Javabeans and EJB.
- Advantages of Java.

You can take one JSP file and move it to another platform, web server or JSP Servlet engine.

HTML and graphics displayed on the web browser are classed as the presentation layer. The Java code (JSP) on the server is classed as the implementation. By having a separation of presentation and implementation, web designer's work only on the presentation and Java developers concentrate on implementing the application.

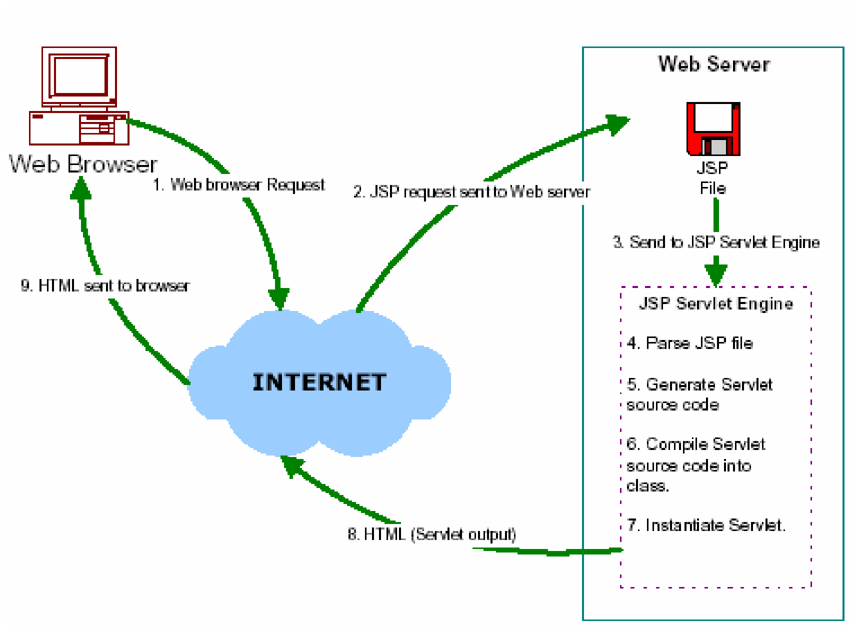


Fig.5.1.2 JSP parsing by web server

JSPs are built on top of SUN's servlet technology. JSPs are essentially an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java servlet source file. It then compiles the source file into a class file, this is done the first time and this why the JSP is probably slower the first time it is accessed. Any time after this the special compiled servlet is executed and is therefore returns faster.

1. The user goes to a web site made using JSP.(ending with .jsp). The web browser makes the request via the Internet.
2. The JSP request gets sent to the Web server.
3. The Web server recognises that the file required is special (.jsp), therefore passes the JSP file to the JSP Servlet Engine.
4. If the JSP file has been called the first time, the JSP file is parsed, otherwise go to step 7.
5. The next step is to generate a special Servlet from the JSP file. All the HTML required is converted to println statements.
6. The Servlet source code is compiled into a class.
7. The Servlet is instantiated, calling the *init* and *service* methods.
8. HTML from the Servlet output is sent via the Internet.
9. HTML results are displayed on the user's web browser.

5.1.1 JSP Environment

- JDK 1.3 minimum required
- JSP environment preferably with J2EE reference implementation Tomcat option
- Set TOMCAT_MOME = C:\tomcat
- To start server goto \tomcat\bin and execute startup
- Create your first program under the directory \tomcat\webapps\
- (all your programs should reside here!)
- Call it via the browser: type http://localhost:8080/firstjsp.jsp

5.1.2 Using JSP tags

There are four main tags:

1. Declaration tag
2. Expression tag
3. Directive Tag
4. Scriptlet tag
5. Action tag

- **Declaration tag (<%! %>)**

This tag allows the developer to declare variables or methods.

Before the declaration you must have <%! And at the end of the declaration, the developer must have %> Code placed in this tag must end in a semicolon (;).

Declarations do not generate output so are used with JSP expressions or scriptlets. For Example:

```
<%! private int counter = 0 ;
private String get Account ( int accountNo) ;
%>
```

- **Expression tag (<%= %>)**

This tag allows the developer to embed any Java expression and is short for out.println().
A semicolon (;) does not appear at the end of the code inside the tag.

```
<%= new java.util.Date() %>//(To show the current date and time)
```

- **Directive tag (<%@ directive … %>)**

A JSP directive gives special information about the page to the JSP Engine. There are three main types of directives:

- Page – processing information for this page.
- Include – files to be included.
- Tag library – tag library to be used in this page.

Directives do not produce any visible output when the page is requested but change the way the JSP Engine processes the page. For example, you can make session data unavailable to a page by setting a page directive (session) to false.

Page directive: This directive has 11 optional attributes that provide the JSP Engine with special processing information. The following table lists the 11 different attributes with a brief description:

Language	language the file uses. <%@ page language = “java” %>
extends	Superclass used by the JSP engine for the translated Servlet. <%@ page extends =“com.taglib…” %>
import	Import all the classes in a java package into the current JSP page. This allows the JSP page to use other java classes. <%@ page import = “java.util.*” %>
session	Does the page make use of sessions. By default all JSP pages have session data available. There are performance benefits to switching session to false. Default is set to true.
buffer	Controls the use of buffered output for a JSP page. Default is 8kb. <%@ page buffer = “none” %>
autoFlush	Flush output buffer when full. <%@ page autoFlush = “true” %>
isThreadSafe	Can the generated Servlet deal with multiple requests? If true a new thread is started so requests are handled simultaneously.
info	Developer uses info attribute to add information/document for a page. Typically used to add author, version, copyright and date info. <%@ page info =“it&webdesign, copyright 2005. “ %>
errorPage	Different page to deal with errors. Must be URL to error page. <%@ page errorPage = “/error/error.jsp” %>
IsErrorPage	This flag is set to true to make a JSP page a special Error Page. This page has access to the implicit object exception.
contentType	Set the mime type and character set of the JSP.

Include directive: Allows a JSP developer to include contents of a file inside another. Typically include files are used for navigation, tables, headers and footers that are common to multiple pages.

<%@ include file = “include/privacy.html %>This includes the html from privacy.html found in the include directory into the current jsp page.

<%@ include file = “navigation.jsp %> To include a naviagation menu (jsp file) found in the current directory.

Tag Lib directive: A tag lib is a collection of custom tags that can be used by the page.

<%@ taglib uri = “tag library URI” prefix = “tag Prefix” %>

Custom tags were introduced in JSP 1.1 and allow JSP developers to hide complex server side code from web designers.

- **Scriptlet tag (<% … %>)**

Between <% and %> tags, any valid Java code is called a Scriptlet. This code can access any variable or bean declared.
To print a variable:

```
<% String username = “board_of_Intermediate” ;
out.println ( username ) ;
%>
```

- Action tag

There are three main roles of action tags :

enable the use of server side Javabeans
transfer control between pages
browser independent support for applets.

5.1.3 Javabeans

A Javabean is a special type of class that has a number of methods. The JSP page can call these methods so can leave most of the code in these javabeans. For example, if you wanted to make a feedback form that automatically sent out an email. By having a JSP page with a form, when the visitor presses the submit button this sends the details to a Javabean that sends out the email. This way there would be no code in the JSP page dealing with sending emails (JavaMail API) and your Javabean could be used in another page (promoting reuse).

To use a Javabean in a JSP page use the following syntax:

```
<jsp : usebean id = “ …. “ scope = “application” class = “com…” />
```

The following is a list of Javabean scopes:

page – valid until page completes.
request – bean instance lasts for the client request
session – bean lasts for the client session.
application – bean instance created and lasts until application ends.

Code examples:
helloworld.jsp

```
<html>
<head>
<title>My first JSP page
</title>
</head>
<body>
<%@ page language=”java” %>
<% System.out.println(“Hello World”); %>
</body>
</html>
**
```

The following example will declare two variables; one string used to stored the name of a website and an integer called counter that displays the number of times the page has been accessed. There is also a private method declared to increment the counter. The website name and counter value are displayed.

```
<HTML>
<HEAD>
<TITLE> JSP Example 2</TITLE>
</HEAD>
<BODY> JSP Example 2
<BR>
<%!
String sitename = “boiap.ac.in”;
int counter = 0;
private void increment Counter()
{
counter ++;
}
%>
Website of the day is
<%= sitename %>
<BR>
page accessed
<%= counter %>
</BODY></HTML>
```

5.1.4 Implicit Objects

So far we know that the developer can create Javabeans and interact with Java objects. There are several objects that are automatically available in JSP called implicit objects.

The implicit objects are

Variable	Of type
Request	Javax.servlet.http.httpservletrequest
Response	Javax.servlet.http. httpservletresponse
Out	Javax.servlet.jsp.JspWriter
Session	Javax.servlet.http.httpsession
PageContent	Javax.servlet.jsp.pagecontext
Application	Javax.servlet.http.ServletContext
Config	Javax.servlet.http.ServletConfig
Page	Java.lang.Object

Page object
Represents the JSP page and is used to call any methods defined by the servlet class.
Config object
Stores the Servlet configuration data.
Request object
Access to information associated with a request. This object is normally used in
looking up parameter values and cookies.

This following code snippet is storing the parameter “dev” in the string devStr. The result is displayed underneath.

```
<% String devStr = request.getParameter(“dev”); %>
Development language = <%= devStr %>
```

5.2 JavaScript

JavaScript is a compact, object-based scripting language for developing client and server Internet applications.

JavaScript statements can be embedded directly in an HTML page. These statements can recognize and respond to user events such as mouse clicks, form input, and page navigation. For example, you can write a JavaScript function to verify that users enter valid information into a form. Without any network transmission, an HTML page with embedded JavaScript can interpret the entered text and alert the user with a message dialog if the input is invalid. Or you can use JavaScript to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in) in response to the user opening or exiting a page.

Using JavaScript, even less-experienced developers will be able to direct responses from a variety of events, objects, and actions. It provides anyone who can compose HTML with the ability to change images and play different sounds in response to specified events, such as a users' mouse click or screen exit and entry.

Java script is multi functional in the sense that it can be used at client side scripting as well as server side scripting.

JavaScript code is typically embedded into an HTML document using the SCRIPT tag. It can be placed inside comment fields to ensure that old browsers that do not recognize JavaScript do not display your JavaScript code. The markup to begin a comment field is <!-- while you close a comment field using //-->.

```
<script language="JavaScript">
<!--  document.write("Hello World!");
//-->
</script>
```

Attribute of the SCRIPT tag, SRC, can be used to include an external file containing JavaScript code rather than code embedded into the HTML:

```
<script language="JavaScript" src="funfunctions.js"></script>
```

5.2.1 Elements of JavaScript

Variables: Name of the memory location on which code is going to act upon. A changeable value during the execution of program. Example: *sum* may possess a value of 100. It is a loosely typed language i.e. one type of data can be automatically converted into other types implicitly according to the need.
Operators: Arithmetic and logical operators. +,-, || ,&& ..etc
Expressions: Any combination of variables, operators, and statements which evaluate to some result.
Statements: Conditional, iterative statements. Ex: if(expression) {……}; while(expression){…} … etc
Objects: An object has some set of values and methods to accessing it. Can be broadly categorized into browser objects, built-in objects and user defined objects.
Functions and Methods: A method is simply a function which is contained in an object. For instance, a function which closes the current window, named *close()*, is part of the window object; thus, *window.close()* is known as a method.

- **Variables:**

A JavaScript identifier, or name, must start with a letter or underscore ("_"); subsequent characters can also be digits (0-9). Because JavaScript is case sensitive, letters include the characters "A" through "Z" (uppercase) and the characters "a" through "z" (lowercase). As a good programming practice variable names are chosen to be meaningful regarding the value they hold. For example, a good variable name for containing the total price of goods orders would be *total_cost*.

A variable may be scoped as either *global* or *local*. A global variable may be accessed from any JavaScript on the page. A local variable may only be accessed from within the function in which it was assigned.

Commonly, you create a new global variable by simply assigning it a value:

```
newValue=5;
```

However, if you are coding within a function and you want to create a local variable which only scopes within that function you must declare the new variable using the var statement:

```
function newFun()
{ var loop=1; //   local variable
  total=0; // Global variable
  ...additional statements...
}
```

In the example above, the variable *loop* will be local to *newFunction()*, while *total* will be global to the entire page.

- Data types:

Numbers Can be integers or Real numbers. Ex: 5 (integer), 5.15
Booleans True or False. not usable as 1 and 0. In a comparison, any expression that evaluates to 0 is taken to be false, and any statement that evaluates to a number other than 0 is taken to be true.
Strings "Hello World!" Strings are enclosed in quotes. Use single quotes to type strings that contain double quotation marks inside their strings and vice versa.
Objects myObj = new Object();
Null Means void, nothing - not even zero!
Undefined A value that is undefined.

- Operators:

Take one or more variables to return a new value. The operator, which performs on a single variable, is called unary operator and operator, which performs on two variables, are called binary operator.

Arithmetic operators: +,-,*,/,% are binary operators (they perform on two operands) for addition, subtraction, multiplication, division, and modulus respectively. Modulus operand returns the remainder after division. Ex: 11%3 = 2; 12 % 3 = 0.

++ , -- are Unary increment and decrement operators. This operator only takes one operand. The operand's value is increased / decreased by 1. The value returned depends on whether the ++ or -- operator is placed before or after the operand; e.g. *++x* will return the value of *x* following the increment (pre increment) whereas *x++* will return the value of *x* prior to the increment (post increment).

There is one operator, which can be used as unary as well as a binary operator. The operator - when used as Unary negation: returns the negation of operand.

Logical Operators: && , || are AND , OR binary operators respectively. Returns the Boolean output of the two operands
! "NOT" an unary operator returns true if the negation of the operand is true (e.g. the operand is false).

Comparison Operators:

= = "Equal to" returns true if operands are equal.
!= "Not equal to" returns true if operands are not equal.
> "Greater than" returns true if left operand is greater than right operand.
>= "Greater than or equal to" returns true if left operand is greater than or equal to right operand.
< "Less than" returns true if left operand is less than right operand.
<= "Less than or equal to" returns true if left operand is less than or equal to right operand

Assignment Operators:

= Assignment operator. Ex: z = x+y
+= Short hand for addition and assignment i.e a +=b is same as a=a+b; Short hand operands are permissible with all arithmetic and logical binary operators.

Statements:

Statements are organized as either conditionals, loops, object manipulations, and comments.

Conditional statements:

```
if (condition)   { statements1; }
                                  else    { statements2; }

switch (expression){
                  case (first match):
                                   statement;
                                  break;
```

```
                    case (second match) :
                    statement;
                    break;
                     …….
                     …….
                    default (if no match): statement;
          }
```

Loop statements:

for (initial-statement; test; increment) { statements; }
do { statements;} while (condition)
while (condition) { statements; }

break Aborts execution of the loop, drops out of loop to the next statement following the loop.

continue Aborts *this single* iteration of the loop, returns execution to the loop control, meaning the condition specified by the loop statement. Loop may execute again if condition is still true.

Object manipulation:

for...in ***for...in*** statement is used to cycle through each property of an object or each element of an array.

with The ***with*** statement serves as a sort of shorthand, allowing to execute a series of statement who all assume a specified object as the reference.

Ex: with (object) {statements; }

Comments:

```
                    // for single comment
                    /* block of comments */
```

- **Functions:**

Increases modularity: There are several advantages if you adopt modular approach towards the development of software. If the software that you are going to develop is visualized as several small groups of statements called modules (rather than a single large block of code statements) then the individual modules can be assigned to different programmers. This causes great reduction in development time. Frequently used tasks can be made as subroutines. Whenever found needed, call them. In JavaScript a function definition does this. This way it also increases the code comprehension and easy to localize the problems that may arise during its usage.

Reusability: Avoids duplicated effort. Programmer need not start from scratch. Make use of existing components that are time tested and build over them.
Reliable software: In software engineering point of view reliable software comprised of highly cohesive modules.

Function definition:

```
function funcName(arg1,arg2,….argn){
  statements;
  }
```

[arg1,arg2, ….., argn] are parameter list, they are going to be input to the function and the function processes them. There may be functions that may not have any parameter list. It is represented by empty brackets () after the function name indicating that it does not have any parameters to take.

It's best to define the functions for a page in the HEAD portion of a document. Since the HEAD is loaded first, this guarantees that functions are loaded before the user has a chance to do anything that might call a function. Alternately, some programmers place all of their functions into a separate file, and include them in a page using the SRC attribute of the SCRIPT tag. Either way, the key is to load the function definitions before any code is executed.

```
function blinkbold(message) {
document.write("<blink><strong>"+message+"</strong></blink>");
}
```
It is invoked by simply specifying it along its parameters.
```
Blinkbold(“Blink Boldly!”);
```

Every function in JavaScript can return at the most one value.

```
function to_power_of(x,y) {
    total=1;
    for (j=0; j<y; j++)
   { total = total*x; }
  return total; //result of x raised to y power
}
```
It is invoked by
```
result = to_power_of(4,3);
```

One more interesting point! In JavaScript most of the times functions are called by event handlers

5.2.2 JavaScript Objects:

In object-oriented design, objects are comprised of properties (variables with values) and methods (functions) relevant to those properties. They are tightly bonded together and are treated under single class. In real life also we can think this world comprised of objects. Every object has its own properties and its own well-defined functions (methods).

5.2.3 Document object model (DOM)

In JavaScript you may create your own objects for storing data. More commonly, though, you will use the many "built-in" objects, which allow working with, manipulating, and accessing the Web page and Web browser. This set of pre-existing objects is known as the "Document Object Model". DOMs are not just for JavaScript. Any other programming language can access them. Netscape and Microsoft have developed their own individual DOM's, which are not entirely compatible. Thus JavaScript is actually made up of JavaScript, the language, and the DOM, or object model, which JavaScript can access.

JavaScript Objects:

There are three types of JavaScript Objects

- Browser Objects
- Utility Objects
- User defined Objects
- Browser Objects:

These are defined in DOM. Whenever html tags <script> …. </script> are encountered, browser makes use of these pre defined objects.

#Browser objects:

Document: The document use to access the current contents of html content
Form: Holds the current contents form information of the html document.

Frame: Refers to frame(s) in the browser window
History: Information about the previous sites, which the browser has visited.
Location: Current web page’s URL and its port
Navigator: Information about the script, which the browser is currently executing
Window: Refers to current browser window

Each of these objects has their own properties, which defines the current state of the object. These objects can be accessed by using . (dot) operator. For ex: objectname.propertyname

document.bgcolr // Back ground color of the current web page
location.hostname // Name of the ISP provider

Each of these objects also have their associated methods. They are also accessed by using . (dot) operator.

docoment.writeln // Writes text to the current web page with a carriage return
history.go // Navigates the browser to a location stored in the history file.
window.alert // Displays an alert box
window.open // Opens a new browser window

#Utility Objects:

These objects are used in processing of the data. The following are few frequently used objects

string object: Used for creating new strings

var str   new string(“a new string”) *// with string object and new operator*
var str   “a new string” *// a new string is created without new operator*

The above two statements are identical because JavaScript recognizes any string as a string object

string object properties: length
Ex: *str.length // returns the length of the string that str has*

string object methods:

big()	Surrounds the string with html <big>…</big> tag
blink()	Surrounds the string with html <blink>…</blink> tag
bold()	Surrounds the string with html <bold>…</bold> tag
charat(n)	Returns the character at the specified ‘n’ index
italics()	Surrounds the string with html <I>…</I> tag
tolowercase()	Makes the entire string to lower case
touppercase()	Makes the entire string to upper case

math object:
Using math object various math operations can be performed.
math object properties:

- E Euler’s constant base of the natural logarithms
- LN10 The natural logarithm of 10 (base 2)
- LN2 The natural logarithm of 2 (base 2)
- PI The value of pie i.e. 22/7

math object methods:
abs() Calculates the absolute value
ceil() Returns the lowest integer greater than or equal to the given number
cos() Cosine of the given number
floor() Returns the lowest integer less than or equal to the given number
pow(x,n) Calculates x^n
random() Return a random number between zero and one
sin() Sine of a given number
sqrt() Square root of a given number
tan() Tangent of a given number

date object: Makes use of system clock
date object methods:

getDate()	Returns day of the month
setDate()	Sets date of the month
getHours()	Returns hours of the day
setHours()	Sets hour of the day
getTime()	Returns lapsed time from 1-1-1970 in milli seconds
SetTime()	Sets time in milli seconds lapsed from 1-1-1970

#User defined objects:
JavaScript allows creating user-defined objects
function stud (sname, srollno, sgroup, srank) {
this.sname=sname;
this.srollno=srollno;

this.sgroup=sgroup;
this.srank=srank;}

The above definition creates an object under the name stud. The instance of the object is created in the following way….

Student1 = new stud (“Ram”, 1234, “mpc”,104);
Student2 = new stud (“Krishna”, 4567, “bpc”, 87);

JavaScript Event handling: JavaScript programs are typically event-driven. *Events* are actions that occur on the Web page, usually as a result of something the user does, although not always. For example, a button click is an event, as is giving focus to a form element; resizing the page is an event, as is submitting a form. It is these events, which cause JavaScript programs to spring into action. For example, if you move your mouse over this phrase, messages will pop-up, courtesy of JavaScript.
An event, then, is the action, which triggers an event handler. The event handler specifies which JavaScript code to execute. Often, event handlers are placed within the HTML tag, which creates the object on which the event acts

JavaScript defines events for most of the major objects found in web page including image maps, forms etc.,

We normally include the event-handling attribute for the events in an appropriate html tag and then specify the event handling JavaScript code as the attribute’s value.

< A HREF = “http://www.osmania.ac.in/employee”
onMouseOver = “alert(‘Link to employee info data base’)” >
Move the mouse over this and a pop up menu is displayed </A>

```
        < FORM INPUT TYPE = “BUTTON” VALUE = “Click me!”
        onClick = “Click handling!”>…...</FORM>
```

The following are some of the most used event handlers

onAbort	The loading of an image is aborted as a result of user action
onBlur	An element uses current focus
onChange	Occurs when the data changes

onClick	Occurs when link, image map, or form element is clicked
onDbclick	Occurs when an element is double clicked.
onError	Occurs when there has been JavaScript error.
onFocus	This event will be called when the element gets focused.
onKeyDown	When user presses a key
onKeyPress	When user presses & releases a key.
onKeyUp	When user releases a key.
onLoad	When the element gets loaded.
onMouseDown	When user presses a mouse button.
onMouseMove	When user moves the mouse.
onMouseOut	When mouse moves out of an element.
onMouseOver	When user moves over an element.
onMouseUp	When user releases a button
onReset	When user presses a reset button.
onResize	When element/Page gets resized.
onSelect	When the text is selected.
onSubmit	When user presses submit button.
onUnload	When the user quits a page/document.

I). First JavaScript program

```
<HTML>
<HEAD>
   <TITLE>Simple Javascript</TITLE>
 </HEAD>
    <BODY>
    <H1>First Example of JavaScript</H1>
    <SCRIPT LANGUAGE="JavaScript">

    <!-- hide from old browsers by embedding in a comment

    document.write("Last updated on"  + document.lastModified  )

    // end script hiding -->
    </SCRIPT>
    <div style="display: block; font-family: Verdana, Geneva, Arial;
font-size: 10px">
          This simple program demonstrates using of JavaScript and
object.
          </div>
     </BODY>
</HTML>
```

II). Handling Events in JavaScript, an alert window

```
<HTML>
<HEAD><TITLE>Handling Events Example</TITLE></HEAD>
<BODY>
<H1>Handling Events in JavaScript</H1>
<FORM>
<INPUT TYPE="button" VALUE="Click me"
onClick="alert('You clicked me')" >
</FORM>
<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
This program uses JavaScript’s alert window!
</div>
</BODY>
</HTML>
```

III). Using for loop

```
<HTML>

<HEAD><TITLE>Computing Factorials</TITLE></HEAD>
<BODY>
<H1>Another Example of JavaScript</H1>
<SCRIPT LANGUAGE="JavaScript">
document.write("<H1>Factorial Table</H1>");
for ( i = 1, fact = 1; i < 10; i++, fact = fact * i) {
document.write(i + "! = " + fact);
document.write("<BR>");
}
</SCRIPT>
<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
This program demonstrates the usage of for loop in calculating
Factorial of numbers from 1 to 9
</div>
</BODY>
</HTML>
```

IV). Displaying system date and time

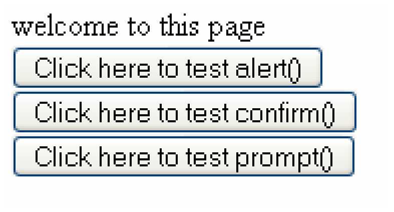
```
<HTML><HEAD><TITLE>Example using new</TITLE>
<SCRIPT LANGUAGE=JavaScript>
function outputDate() {
var d = new Date(); //creates today's date and time
document.write(d.toLocaleString()); } // converts a date to a string
</SCRIPT></HEAD>
<BODY>
<H1>The date and time are</H1>
<SCRIPT LANGUAGE=JavaScript>
outputDate();
</SCRIPT>
<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
This program illustrates use of user defined function
</div>
</BODY>
</HTML>
```

V). Usage of alert, confirm and prompt buttons

```
<HTML><HEAD><TITLE>Example of alert, confirm,
prompt</TITLE>


<SCRIPT LANGUAGE=JavaScript>
function alertUser() { alert("An alert box contains an exclamation
mark");}
function confirmUser() {
var msg = "\n please confirm that you want\n" +
"to test another button?";
if (confirm(msg)) document.write("<h2>You selected OK</h2>");
else document.write("<h2>You selected Cancel</h2>"); }
function promptUser() {
name1=prompt("What is your name?", " ");
document.write("<h2>welcome to this page " + name1 + "</h2>"); }
</SCRIPT></HEAD><BODY>welcome to this page<br>
<FORM>
<INPUT TYPE=button VALUE="Click here to test alert()"
onClick="alertUser()"><BR>
```

```
<INPUT TYPE=button VALUE="Click here to test confirm()"
onClick="confirmUser()"><BR>
<INPUT TYPE=button VALUE="Click here to test prompt()"
onClick="promptUser()"></FORM>

<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
This program uses forms along with Javascript!
</div>
</BODY>
</HTML>
```

Typical Browser output (Buttons only):


welcome to this page
Click here to test alert()
Click here to test confirm()
Click here to test prompt()

VI). Getting browser information

```
<HTML><HEAD><TITLE>Browser Info</TITLE>
</HEAD><BODY>
<SCRIPT language=JavaScript>
document.write("<BR> This browser is "
+ navigator.appName);
document.write("<BR> Version "
+ navigator.appVersion);
if (parseFloat(navigator.appVersion) >= 4)
{ document.write("<BR> You have an up-to-date browser"); }
</SCRIPT>

<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
This program displays the browser name and its version
</div>
</BODY>
</HTML>
```

VII). Experimenting with banners

```
<HEAD><TITLE>Banner</TITLE>

<SCRIPT language=JavaScript>
var banners = new Array();
banners[0] = "banner0";
banners[1] = "banner1";
banners[2] = "banner2";
var which = 0;

function display(id, str) {
        if (document.all) {
                document.all[id].innerHTML=str;}
        else {
                document[id].document.open();
                document[id].document.write(str);
                document[id].document.close();
                }
}
function update() {
        display("banner", banners[which]);
        which++;
        if (which == banners.length) { which = 0;};
}

</SCRIPT></HEAD>
<BODY onload=" if (setInterval) { setInterval('update()', 500);}">
<p>An example of a changing text banner</P>
<P><SPAN ID="banner" STYLE="position:absolute;"><I>
<SCRIPT Language=Javascript>
if (setInterval)
{document.write('the banner is loading');}
else
{document.write('Get a new browser');};
</SCRIPT></I></SPAN>
<BR><P>Here is the remainder of the document</P>

</BODY></HTML>
```

VIII). Image links

```
<HTML><HEAD><TITLE>Image Links</TITLE>
<SCRIPT language=javascript>
   function highlight(imgName) {
     if (document.images) {
             document.images[imgName].src =
onImages[imgName].src;}}

   function unhighlight(imgName) {
     if (document.images) {
             document.images[imgName].src=
offImages[imgName].src;}}

   if (document.images) {
     var onImages = new Array;
          var offImages = new Array;

          onImages["docs"] = new Image();
          onImages["docs"].src = "images/docs_on.gif";
          offImages["docs"] = new Image();
          offImages["docs"].src = "images/docs_off.gif";
          onImages["tech"] = new Image();
          onImages["tech"].src = "images/tech_on.gif";
          offImages["tech"] = new Image();
          offImages["tech"].src = "images/tech_off.gif";
}


</SCRIPT></HEAD>
<BODY>
 <P> <A HREF="subdocs/documents.html"
onMouseOver="highlight('docs')" onMouseOut="unhighlight('docs')">

   <IMG SRC="images/docs_off.gif" height=25 width=25 name=docs
border=0
   alt=documents></A>

   <A HREF="subdocs/tech.html" onMouseOver="highlight('tech')"
// sub directory in current location
onMouseOut="unhighlight('tech')">
```

```
    <IMG SRC="images/tech_off.gif" height=25 width=25 name=tech
border=0
    alt=tech_reports></A>    // besure to have images in images directory
  </P>
</BODY>
</HTML>
```

IX). Simple frame

```
<HTML>
<HEAD>
<TITLE>Simple Frame Ex</TITLE>
</HEAD>
<FRAMESET rows="50%,50%">
            <FRAME src= red.html name=frame1> //red.html must be in
current directory
            <FRAMESET cols="30%,70%" >
<frame src= green.html name=frame2 > // green.html must be in current
directory
<frame src= blue.html name=frame3>// blue.html must be in current
directory
            </FRAMESET>
<NOFRAMES>You need frames to view this document</noframes>
</FRAMESET>
</HTML>
<div style="display: block; font-family: Verdana, Geneva, Arial; font-
size: 10px">
Blank Blank Blank
</div>
```

X). Using document object

```
<HTML>
<HEAD>
<TITLE>Using the Document Object</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    function SetColors()
    {
      document.fgColor = "white";
      document.bgColor = "black";
    }
</SCRIPT>
```

```
</HEAD>


<BODY>
 <SCRIPT LANGUAGE="JavaScript">
 SetColors()
 </SCRIPT>
<PRE>
Explanation: In BODY calls SetColors()
function SetColors()
     {
       document.fgColor = "white";
       document.bgColor = "black";
     }     </PRE>
<hr>
  <font size=+3>This text is displayed in white<br>
  <font color=red>This text is displayed in red</font></font>
</BODY>
</HTML>
```

XI). Using for Input validation

```
<html>
<head>
<title>Using for Input validation</title>

<script language=JavaScript>
    function checkEmpty()
    {
    /* This function checks all of the fields of the form and notifies the
client which, if any, form fields are empty.  It returns a 1 if all the fields
are full, and a 0 otherwise.
    */

            var firstname_filled=false;
            var lastname_filled=false;
            var streetaddress_filled=false;
            var city_filled=false;
            var phonenumber_filled=false;
            var youremail_filled=false;
            var blank="";

        if (document.myform.firstname.value != blank)
```

```
            firstname_filled=true;
         if (document.myform.lastname.value != blank)
            lastname_filled=true;
         if (document.myform.streetaddress.value != blank)
            streetaddress_filled=true;
         if (document.myform.city.value != blank)
            city_filled=true;
         if (document.myform.phonenumber.value != blank)
            phonenumber_filled=true;
         if (document.myform.youremail.value != blank)
            youremail_filled=true;

      if ( (firstname_filled) && (lastname_filled)      &&
(streetaddress_filled) &&
             (city_filled)      && (phonenumber_filled)   &&
(youremail_filled) )
         {
            alert("No missing fields");
            return(true);
         }
         else
         {/* check which fields are missing */

             var alertstring="The following fields are missing:\n";

             if (!firstname_filled)
                   alertstring=alertstring + "first name\n";
             if (!lastname_filled)
                   alertstring=alertstring + "last name\n";
             if (!streetaddress_filled)
                   alertstring=alertstring + "street address\n";
             if (!city_filled)
                   alertstring=alertstring + "city\n";
             if (!phonenumber_filled)
                   alertstring=alertstring + "phone number\n";
             if (!youremail_filled)
                   alertstring=alertstring + "email\n";

             alert(alertstring);
             return(false);
      }
   }
```

```
</script>
</head>
<BODY bgcolor="#ffffee">   <pre>
Explanation: Each field is checked for a missing value, and if empty
a boolean variable is set. A string is created of all empty fields
and output.
</pre><hr>

<H1><FONT color="#000000">Checking For Empty Form
Fields</H1></FONT><HR>

Type in the data you wish to save to a file on the server.

<FORM NAME="myform" METHOD="GET">

First name: <INPUT type="text" name="firstname" size=16 ><br>
Last name: <INPUT type="text" name="lastname" size=16 ><br>
Street address: <INPUT type="text" name="streetaddress" size=41
><br>
City name: <INPUT type="text" name="city" size=21 ><br>
Phone number: <INPUT type='text' name='phonenumber' size=13 ><br>
Email: <INPUT type='text' name='youremail' size =20><br> <hr>

<INPUT type='button' value='Add new entry' onClick="checkEmpty()"
><br>
</FORM>
<a href="../ ">Return to Main Page</a>
</body>
</html>
```

XII). To view selected items list

```
<HTML><HEAD>  <TITLE>Using Multiple Selection Lists</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    function displaySelectionValues(objectName) {
      var ans = "";
      for (var i = 0; i < objectName.length; i++){
                if(objectName.options[i].selected){
                ans += objectName.options[i].text + "<BR>";}
      }

      myWin = window.open("", "Selections", "height=200,width=400")
```

```
   myWin.document.write("You picked these teams:<BR>")
   myWin.document.write(ans)
  }
 </SCRIPT>
</HEAD>
<BODY>
 <FORM NAME="myform" method="GET">
Use Ctrl key to select multiple items<br>
  <SELECT NAME="teams" size=3 multiple>
   <OPTION value="dodgers">Dodgers
   <OPTION value="yankees">Yankees
   <OPTION value="angels">Angels
  </SELECT><BR>
  <INPUT TYPE="button" value="Show Selected Items"
onClick="displaySelectionValues(this.form.teams)">
 </FORM>
 <a href="../">Return to Main Page</a>
</BODY>
</HTML>
```

XIII). Using Date object

```
<HTML>
<HEAD>
<TITLE>Using the Date Object</TITLE>
<SCRIPT LANGUAGE="JavaScript">
  function Clock(hours, minutes, seconds)  {
   this.hours = hours; this.minutes = minutes; this.seconds = seconds;
  }
  function DisplayClock(Clock) {
   var clockdisp=Clock.hours + ":" + Clock.minutes + ":" +
Clock.seconds;
   document.write("<BR>The time is:" + clockdisp);
  }
 </SCRIPT>
</HEAD>
<BODY>
<HR>
 <SCRIPT LANGUAGE="JavaScript">
  var timenow = new Date;
  document.write(timenow);
  thisClock = new
Clock(timenow.getHours(),timenow.getMinutes(),timenow.getSeconds()
);
```

```
   DisplayClock(thisClock);
  </SCRIPT>
  <br>
  <a href="../ ">Return to Main Page</a>

</BODY>
</HTML>
```

Few JSP examples:

These examples will only work when these pages are being served by a servlet engine like Tomcat. They will not work if you are viewing these pages via a "file://..." URL.

Number guess program

```
<%@ page import = "num.NumberGuessBean" %>

<jsp:useBean id="numguess" class="num.NumberGuessBean"
scope="session"/>
<jsp:setProperty name="numguess" property="*"/>

<html>
<head><title>Number Guess</title></head>
<body bgcolor="white">
<font size=4>

<% if (numguess.getSuccess()) { %>
  Congratulations!  You got it.
  And after just <%= numguess.getNumGuesses() %> tries.<p>
  <% numguess.reset(); %>

  Care to <a href="numguess.jsp">try again</a>?
<% } else if (numguess.getNumGuesses() == 0) { %>
  Welcome to the Number Guess game.<p>
  I'm thinking of a number between 1 and 100.<p>
  <form method=get>
  What's your guess? <input type=text name=guess>
  <input type=submit value="Submit">
  </form>
<% } else { %>
```

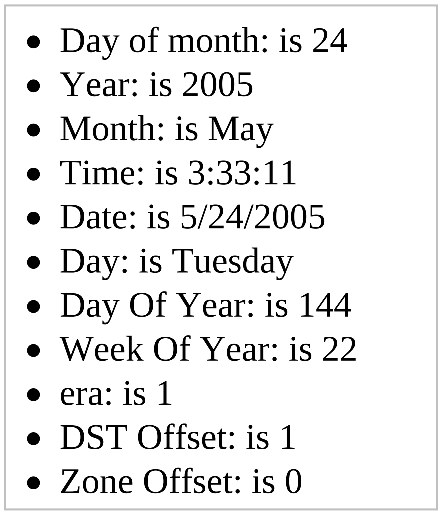
```
  Good guess, but nope.  Try <b><%= numguess.getHint() %></b>.
 You have made <%= numguess.getNumGuesses() %> guesses.<p>

  I'm thinking of a number between 1 and 100.<p>

  <form method=get>
  What's your guess? <input type=text name=guess>
  <input type=submit value="Submit">
  </form>
<% } %>
</font>
</body>
</html>
```

Displaying date

```
<html>
<%@ page session="false"%>

<body bgcolor="white">
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar'
type="dates.JspCalendar" />

<font size=4>
<ul>
<li>     Day of month: is  <jsp:getProperty name="clock"
property="dayOfMonth"/>
<li>     Year: is  <jsp:getProperty name="clock" property="year"/>
<li>     Month: is  <jsp:getProperty name="clock" property="month"/>
<li>     Time: is  <jsp:getProperty name="clock" property="time"/>
<li>     Date: is  <jsp:getProperty name="clock" property="date"/>
<li>     Day: is  <jsp:getProperty name="clock" property="day"/>
<li>     Day Of Year: is  <jsp:getProperty name="clock"
property="dayOfYear"/>
<li>     Week Of Year: is  <jsp:getProperty name="clock"
property="weekOfYear"/>
<li>     era: is  <jsp:getProperty name="clock" property="era"/>
<li>     DST Offset: is  <jsp:getProperty name="clock"
property="DSTOffset"/>
<li>     Zone Offset: is  <jsp:getProperty name="clock"
property="zoneOffset"/>
</ul>
```

```
</font>
</body>
</html>
```

Typical output from the browser



```
• Day of month: is 24
• Year: is 2005
• Month: is May
• Time: is 3:33:11
• Date: is 5/24/2005
• Day: is Tuesday
• Day Of Year: is 144
• Week Of Year: is 22
• era: is 1
• DST Offset: is 1
• Zone Offset: is 0
```

Hello world using tag
hello.jsp

```
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>
<html>
  <head>
    <title>JSP 2.0 Examples - Hello World Using a Tag File</title>
  </head>
  <body>
    <h1>JSP 2.0 Examples - Hello World Using a Tag File</h1>
    <hr>
    <p>This JSP page invokes a custom tag that simply echos "Hello,
World!"
    The custom tag is generated from a tag file in the /WEB-INF/tags
    directory.</p>
    <p>Notice that we did not need to write a TLD for this tag.  We just
    created /WEB-INF/tags/helloWorld.tag, imported it using the taglib
    directive, and used it!</p>
    <br>
    <b><u>Result:</u></b>
    <tags:helloWorld/>
  </body>
</html>
```

helloWorld.tag

```
<%--
  hello world tag!
--%>
Hello, world!
```

Writing plugins:

Tomcat 5 provides a framework for implementing tag plugins. The plugins instruct Jasper, at translation time, to replace tag handler calls with Java scriptlets. The framework allows tag library authors to implement plugins for their tags.
Tomcat 5 is released with plugins for several JSTL tags. Note that these plugins work with JSTL 1.1 as well as JSTL 1.0, though the examples uses JSTL 1.1 and JSP 2.0. These plugins are not complete (for instance, some item types not handled in <c:if>). They do serve as examples to show plugins in action (just examine the generated Java files), and how they can be implemented.

How to write tag plugins

To write a plugin, you'll need to download the source for Tomcat 5. There are two steps:

- Implement the plugin class.
  This class, which implements org.apache.jasper.compiler.tagplugin.TagPlugin instructs Jasper what Java codes to generate in place of the tag handler calls. See Javadoc for org.apache.jasper.compiler.tagplugin.TagPlugin for details.

- Create the plugin descriptor file WEB-INF/tagPlugins.xml
  This file specifies the plugin classes and their corresponding tag handler classes.

choose.jsp

```
<html>
  <head>
    <title>Tag Examples - choose</title>
  </head>
  <body>
    <h1>Tag Plugin Examples - &lt;c:choose></h1>
    <hr>
    </br> <a href="notes.html">Plugin Introductory Notes<font <font
color="#0000FF"></a>
    <br/>
<a href="howto.html">Brief Instructions for Writing Plugins
<font color="#000FF"></a>
    <br/> <br/>
    <hr>

    <font color="#000000"/>
    </br>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <c:forEach var="index" begin="0" end="4">
      # ${index}:
      <c:choose>
            <c:when test="${index == 1}">
          One!</br>
            </c:when>
            <c:when test="${index == 4}">
          Four!</br>
            </c:when>
            <c:when test="${index == 3}">
          Three!</br>
            </c:when>
            <c:otherwise>
          Huh?</br>
            </c:otherwise>
      </c:choose>
    </c:forEach>
  </body>
</html>
```

expression languages

```
<html>
  <head>
    <title>JSP 2.0 Expression Language - Basic Comparisons</title>
  </head>
  <body>
    <h1>JSP 2.0 Expression Language - Basic Comparisons</h1>
```

```
<hr>
This example illustrates basic Expression Language comparisons.
The following comparison operators are supported:
<ul>
  <li>Less-than (&lt; or lt)</li>
  <li>Greater-than (&gt; or gt)</li>
  <li>Less-than-or-equal (&lt;= or le)</li>
  <li>Greater-than-or-equal (&gt;= or ge)</li>
  <li>Equal (== or eq)</li>
  <li>Not Equal (!= or ne)</li>
</ul>
<blockquote>
  <u><b>Numeric</b></u>
  <code>
    <table border="1">
      <thead>
            <td><b>EL Expression</b></td>
            <td><b>Result</b></td>
          </thead>
          <tr>
            <td>\${1 &lt; 2}</td>
            <td>${1 < 2}</td>
          </tr>
          <tr>
            <td>\${1 lt 2}</td>
            <td>${1 lt 2}</td>
          </tr>
          <tr>
            <td>\${1 &gt; (4/2)}</td>
            <td>${1 > (4/2)}</td>
          </tr>
          <tr>
            <td>\${1 &gt; (4/2)}</td>
            <td>${1 > (4/2)}</td>
          </tr>
          <tr>
            <td>\${4.0 &gt;= 3}</td>
            <td>${4.0 >= 3}</td>
          </tr>
          <tr>
            <td>\${4.0 ge 3}</td>
            <td>${4.0 ge 3}</td>
          </tr>
          <tr>
            <td>\${4 &lt;= 3}</td>
            <td>${4 <= 3}</td>
          </tr>
          <tr>
            <td>\${4 le 3}</td>
            <td>${4 le 3}</td>
          </tr>
          <tr>
            <td>\${100.0 == 100}</td>
            <td>${100.0 == 100}</td>
          </tr>
          <tr>
            <td>\${100.0 eq 100}</td>
            <td>${100.0 eq 100}</td>
          </tr>
          <tr>
            <td>\${(10*10) != 100}</td>
            <td>${(10*10) != 100}</td>
          </tr>
          <tr>
            <td>\${(10*10) ne 100}</td>
            <td>${(10*10) ne 100}</td>
          </tr>
        </table>
    </code>
    <br>
    <u><b>Alphabetic</b></u>
    <code>
      <table border="1">
        <thead>
            <td><b>EL Expression</b></td>
            <td><b>Result</b></td>
          </thead>
          <tr>
            <td>\${'a' &lt; 'b'}</td>
            <td>${'a' < 'b'}</td>
          </tr>
          <tr>
            <td>\${'hip' &gt; 'hit'}</td>
            <td>${'hip' > 'hit'}</td>
          </tr>
          <tr>
            <td>\${'4' &gt; 3}</td>
            <td>${'4' > 3}</td>
          </tr>
        </table>
     </code>
   </blockquote>
 </body>
</html>
```

The output for the program is:

This example illustrates basic Expression Language comparisons. The following comparison operators are supported:

- Less-than (< or lt)
- Greater-than (> or gt)
- Less-than-or-equal (<= or le)
- Greater-than-or-equal (>= or ge)
- Equal (== or eq)
- Not Equal (!= or ne)

Numeric

EL Expression	Result
${1 < 2}	true
${1 lt 2}	true
${1 > (4/2)}	false
${1 > (4/2)}	false
${4.0 >= 3}	true
${4.0 ge 3}	true
${4 <= 3}	false
${4 le 3}	false
${100.0 == 100}	true
${100.0 eq 100}	true
${(10*10) != 100}	false
${(10*10) ne 100}	false

Alphabetic

EL Expression	Result
${'a' < 'b'}	true
${'hip' > 'hit'}	false
${'4' > 3}	true

5.3 Web technologies

- **Internet**

The Internet is a global network of networks connecting millions of users worldwide via many computer networks using a simple standard common addressing system and communications protocol called TCP/IP. Clearly, the Internet supports the web, and you should have a working knowledge of topics such as domain names, protocols, security and privacy, etc.

- **Hypertext Markup Language (HTML)**

There are several languages that can be used to create a web site. The underlying foundation, which binds web pages together, is Hypertext Markup Language - the fundamental building stuff of the web.

It is a non-proprietary format, based upon SGML, for describing the structure of hypermedia documents - plain text (ASCII) files with embedded codes for logical markup, using tags like <A> and </A> to structure text into tables, hypertext links interactive forms, headings, paragraphs, lists, and more.

- **Cascading Style Sheets (CSS)**

It is an important step towards separating content and presentation. Style Sheets allow you to control the rendering, e.g. fonts, colors, leading, margins, typefaces, and other aspects of style, of a Web document without compromising its structure. CSS is a simple style sheet mechanism that allows authors and readers to attach style to HTML documents. It also enables some features not offered by HTML, such as removing link underlining.

- **Graphics Tools and Techniques**

Graphics add spice and style to web pages, and can help your visitors visualize what your site is about and how it's structured. You can spice up your pages with tasteful backgrounds, or 3d graphics, using tools such as Adobe Photoshop and Paint Shop Pro.
For many people, *Web Design = Graphics*. Almost all web sites benefit from well-planned images - especially if they are fast-loading, and the site still remains usable if the graphics are ignored, e.g. by text-only browsers etc.

- **Programming and scripting**

The web becomes more than just static pages when you make it interactive with Programming and Scripting techniques. You can combine **HTML**, **Style Sheets** and some **Scripting** for **Dynamic HTML** to make web page more attractive.

- **PHP (Hypertext Preprocessor)**

Self-referentially short for ***PHP**: **H**ypertext **P**reprocessor*, an open source, server-side, HTML embedded scripting language used to create dynamic Web pages.

In an HTML document, PHP script (similar syntax to that of Perl or C) is enclosed within special PHP tags. Because PHP is embedded within tags, the author can jump between HTML and PHP (similar to ASP and Cold Fusion) instead of having to rely on heavy amounts of code to

output HTML. And, because PHP is executed on the server, the client cannot view the PHP code.
PHP can perform any task that any CGI program can do, but its strength lies in its compatibility with many types of databases. Also, PHP can talk across networks using different protocols.

PHP is one of the most popular scripting languages for use on the web. Now in version 4, its phenomenal growth has come mostly at the expense of Perl and CGI. It's open-source, easy-to-learn for Perl programmers, and is rapidly expanding beyond only web use, making it a good language to learn.

- **JavaScript**

JavaScript is NOT JAVA! They are quite different. JavaScript is a lightweight client-only scripting language, suitable for calculators and other relatively non-GUI applications. In particular, it is needed for Dynamic HTML. A client-side technology, essential for effective dynamic pages; problematic due to differences between the major browsers..

- **Dynamic HTML**

Dynamic HTML is typically used to describe the combination of HTML, style sheets and scripts that allows a web page to change after it's loaded into the browser --there doesn't have to be any communication with the web server for an update. You can think of it as 'animated' HTML. For example, a piece of text can change from one size or color to another, or a graphic can move from one location to another, in response to some kind of user action, such as clicking a button.

- **Multimedia**

Graphics spice up a web site, and with GIF animation can be made more eye-catching (or distracting, and certainly larger). But there are many more options, allowing interesting audio and visual effects, e.g Shockwave, Java, etc.

- **Databases**

The power of the WWW comes not simply from static HTML pages - which can be very attractive, and the important first step into the WWW - but especially from the ability to support those pages with powerful software, especially when interfacing to databases. The combination of attractive screen displays, exceptionally easy to use controls and navigational aids, and powerful underlying software, has opened up the potential for people everywhere to tap into the vast global information resources of the Internet.

5. Summary:

Java Server Pages (JSP) technology provides an easy way to create dynamic web pages and simplify the task of building web applications that work with a wide variety of web servers, application servers, browsers and development tools.

JSP and ASP are fairly similar in the functionality that they provide. JSP may have slightly higher learning curve.

JavaScript is a compact, object-based scripting language for developing client and server Internet applications.

Dynamic HTML is typically used to describe the combination of HTML, style sheets and scripts that allows a web page to change after it's loaded into the browser

Short questions:

1). JSP is acronym for …..
2). Where the JSP code executed?
3). On which language JSP is based?
4). On which server JSP is normally executed?
5). State the relation between javabeans and JSP
6). Is JSP is server or / client side application?
7). Explain about JSP directive tag
8). Are Java and JavaScript different or same?
9). Is JavaScript a compiled or/ interpreted language?
10). DOM is acronym for …..

Long questions:

1). Contrast JSP and ASP
2). Justify the reasons for opting JSP
3). Describe the JSP parsing by web server
4). List the steps to implement JSP environment
5). What are JSP tags? How many are there? Furnish their syntax
6). What are JSP’s implicit objectys?
7). What are the elements of JavaScript?
8). Site an example code of JavaScript for input validation
9). How many types of JavaScript objects are there? Discuss in detail
10). What is DHTML? List the technologies involved