GOVERNMENT OF TAMILNADU
DIRECTORATE OF TECHNICAL EDUCATION
CHENNAI – 600 025

STATE PROJECT COORDINATION UNIT

Diploma in Computer Engineering

Course Code: 1052
M – Scheme

e-TEXTBOOK
on
# Web Programminng
for
V Semester Dip. In Comp. Engg.

Convener for Computer Engineering Discipline:

Mr.D. Arulselvan
HOD / Post Diploma in Computer Applications
Thiagarajar Polytechnic College, Salem – 636 005

Team Members :

Mrs. A.Nalini (1st and 2nd units)
HOD / Computer Engg
Nachimuthu Polytechnic College
Pollachi -642003

Mr.B.Krishnakumar (3rd and 4th units)
HOD / Computer Engg
Arasan Ganesan Polytechnic College,
Sivakasi – 626130.

Mr. M.Thirumurugan (5th unit)
Lecturer/Computer Engg.
Arasan Ganesan Polytechnic College,
Sivakasi – 626130.

Validated By
Mr.M.Jeyapal,
Lecturer (Sel.Gr) / Computer Engg.
Govt. Polytechnic College
Arantangi

UNIT 1

INTERNET AND HTML

Introduction to internet:

Definition of Internet

Internet is defined as an Information super Highway, to access information over the web. However, It can be defined in many ways as follows:

- ✓ Internet is a world-wide global system of interconnected computer networks.
- ✓ Internet uses the standard Internet Protocol (TCP/IP).
- ✓ Every computer in internet is identified by a unique IP address.
- ✓ IP Address is a unique set of numbers (such as 110.22.33.114) which identifies a computer location.
- ✓ A special computer DNS (Domain Name Server) is used to give name to the IP Address so that user can locate a computer by a name.
- ✓ For example, a DNS server will resolve a name http://www.nptc.ac.in to a particular IP address to uniquely identify the computer on which this website is hosted.
- ✓ Internet is accessible to every user all over the world.

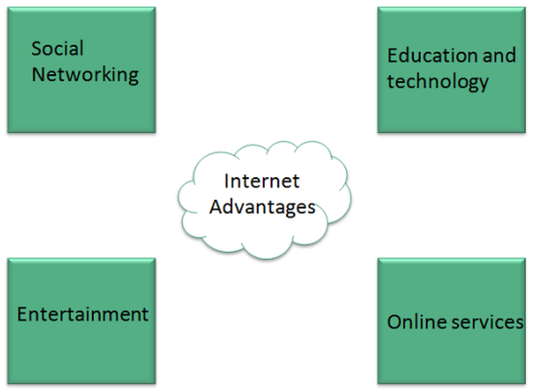
History Of Internet (Evolution)

The concept of Internet was originated in 1969 and has undergone several technological & Infrastructural changes as discussed below:

- ✓ The origin of Internet devised from the concept of Advanced Research Project Agency Network (ARPANET).
- ✓ ARPANET was developed by United States Department of Defense.
- ✓ Basic purpose of ARPANET was to provide communication among the various bodies of government.
- ✓ Initially, there were only four nodes, formally called Hosts.
- ✓ In 1972, the ARPANET spread over the globe with 23 nodes located at different countries and thus became known as Internet.
- ✓ By the time, with invention of new technologies such as TCP/IP protocols, DNS, WWW, browsers, scripting languages etc.,Internet provided a medium to publish and access information over the web.

Advantages

- ✓ Internet covers almost every aspect of life, one can think of. Here, we will discuss some of the advantages of Internet:

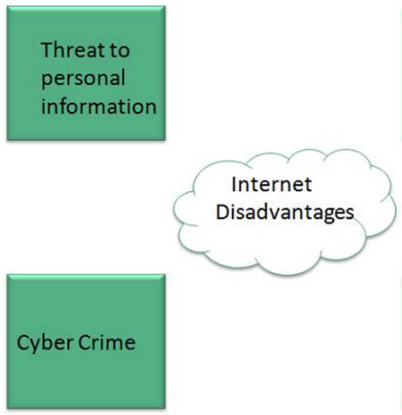


- ✓ Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the wed that uses Internet as a medium for communication. One can find various social networking sites such as:
    - Facebook
    - Twitter
    - Yahoo
    - Google+
    - Flickr
    - Orkut
- ✓ One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.
- ✓ Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet.
    - Online Television
    - Online Games
    - Songs
    - Videos
    - Social Networking Apps
- ✓ Internet allows us to use many services like:
    - Internet Banking
    - Matrimonial Services
    - Online Shopping
    - Online Ticket Booking
    - Online Bill Payment
    - Data Sharing
    - E-mail

- ✓ Internet provides concept of **electronic commerce**, that allows the business deals to be conducted on electronic systems

Disadvantages

However, Internet has proved to be a powerful source of information in almost every field, yet there exists many disadvantages discussed below:

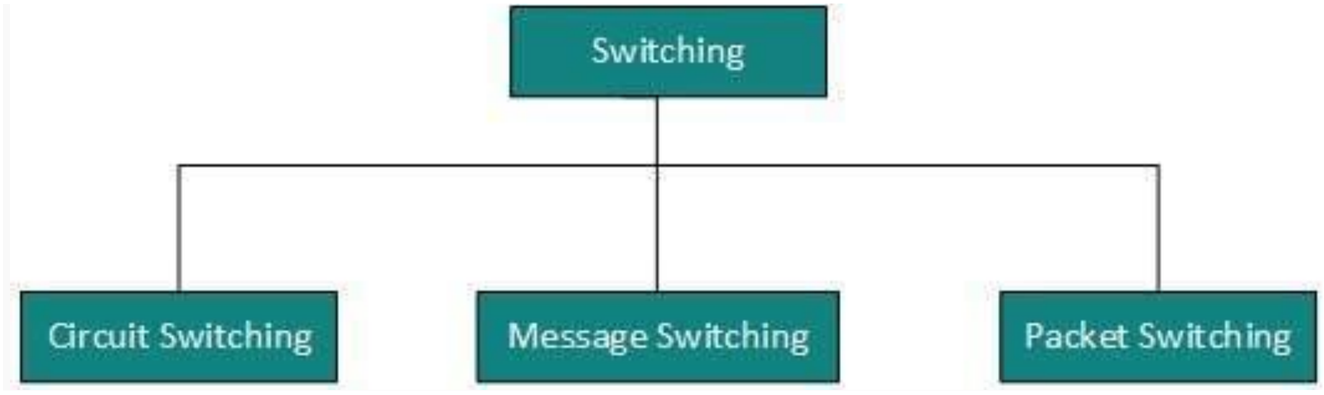


- There are always chances to loose personal information such as name, address, credit card number. Therefore, one should be very careful while sharing such information. One should use credit cards only through authenticated sites.
- Another disadvantage is the **Spamming**. Spamming corresponds to the unwanted e-mails in bulk. These e-mails serve no purpose and lead to obstruction of entire system.
- **Virus** can easily be spread to the computers connected to internet. Such virus attacks may cause your system to crash or your important data may get deleted.
- Also a biggest threat on internet is pornography. There are many pornographic sites that can be found, letting your children to use internet which indirectly affects the children healthy mental life.
- There are various websites that do not provide the authenticated information. This leads to misconception among many people.

Switching

Switching is a mechanism by which data/information sent from source towards destination which are not directly connected. Networks have interconnecting devices, which receives data from directly connected sources, stores data, analyze it and then forwards to the next interconnecting device closest to the destination.

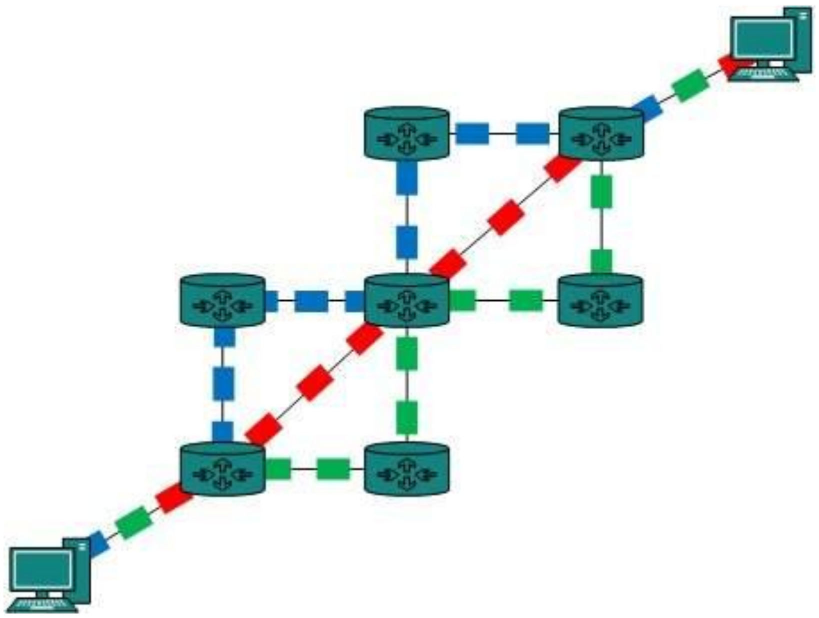
Switching can be categorized as:



Packet Switching

Shortcomings of message switching gave birth to an idea of packet switching. The entire message is broken down into smaller chunks called packets. The switching information is added in the header of each packet and transmitted independently.

It is easier for intermediate networking devices to store small size packets and they do not take much resources either on carrier path or in the internal memory of switches.



Packet switching enhances line efficiency as packets from multiple applications can be multiplexed over the carrier. The internet uses packet switching technique. Packet switching enables the user to differentiate data streams based on priorities. Packets are stored and forwarded according to their priority to provide quality of service.

Different types of Connections

There exist several ways to connect to the internet. Following are these connection types available:

1. Dial-up Connection
2. ISDN
3. DSL
4. ///Cable TV Internet connections ///
5. Satellite Internet connections
6. ///Wireless Internet Connections///

Dial-up Connection

Dial-up connection uses telephone line to connect PC to the internet. It requires a modem to setup dial-up connection. This modem works as an interface between PC and the telephone line.

There is also a communication program that instructs the modem to make a call to specific number provided by an ISP.

Dial-up connection uses either of the following protocols:

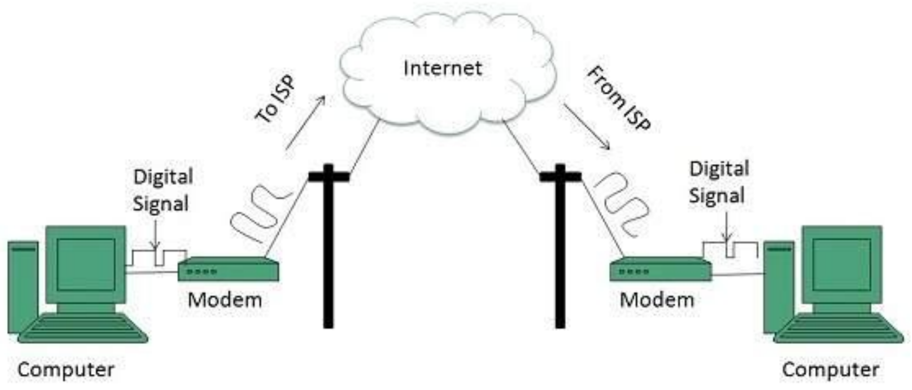
Serial Line Internet Protocol (SLIP)

The **Serial Line Internet Protocol** (also **SLIP**) is an encapsulation of the Internet **Protocol** designed to work over serial ports and modem connections. It is documented in RFC 1055.

Point to Point Protocol (PPP)

In computer networking, **Point-to-Point Protocol**(PPP) is a data link (layer 2) protocol used to establish a direct connection between two nodes. It can provide connection authentication, transmission encryption (using ECP, RFC 1968), and compression.

The following diagram shows the accessing internet using modem:



ISDN

ISDN is acronym of **Integrated Services Digital Network.** It establishes the connection using the phone lines which carry digital signals instead of analog signals.

There are two techniques to deliver ISDN services:

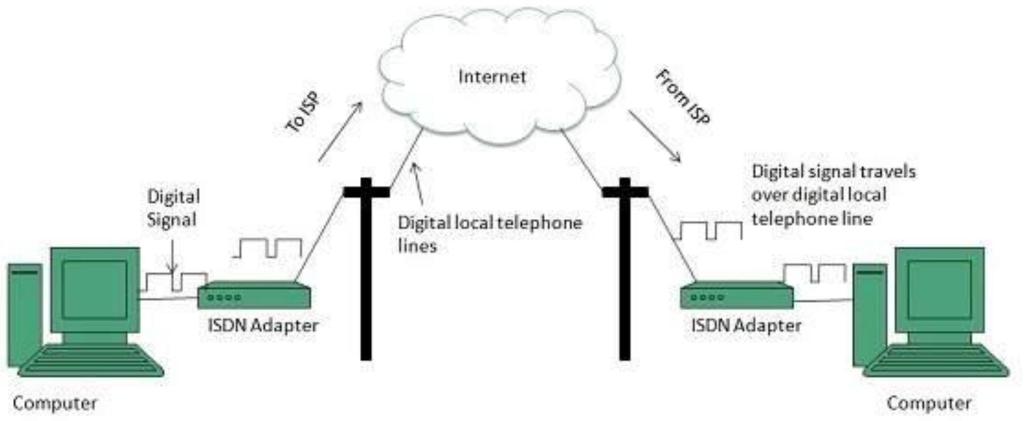
1. Basic Rate Interface (BRI)
2. Primary Rate Interface (PRI)

Key points:

The BRI ISDN consists of three distinct channels on a single ISDN line: t1o 64kbps B (Bearer) channel and one 16kbps D (Delta or Data) channels.

The PRI ISDN consists of 23 B channels and one D channels with both have operating capacity of 64kbps individually making a total transmission rate of 1.54Mbps.

The following diagram shows accessing internet using ISDN connection:


Internet
To ISP
From ISP
Digital Signal
Digital local telephone lines
Digital signal travels over digital local telephone line
ISDN Adapter
ISDN Adapter
Computer
Computer

ISDN Advantages

- ✓ The basic advantage of ISDN is to facilitate the user with multiple digital channels. These channels can operate concurrently through the same one copper wire pair.
- ✓ The digital signals broadcasting transversely the telephone lines.
- ✓ ISDN provides high data rate because of digital scheme which is 56kbps.
- ✓ ISDN network lines are able to switch manifold devices on the single line such as faxes, computers, cash registers credit cards readers, and many other devices. These all devices can work together and directly be connected to a single line.
- ✓ ISDN takes only 2 seconds to launch a connection while other modems take 30 to 60 second for establishment.

ISDN Disadvantages

- ✓ The disadvantage of ISDN lines is that it is very costly than the other typical telephone system.
- ✓ ISDN requires specialized digital devices just like Telephone Company.

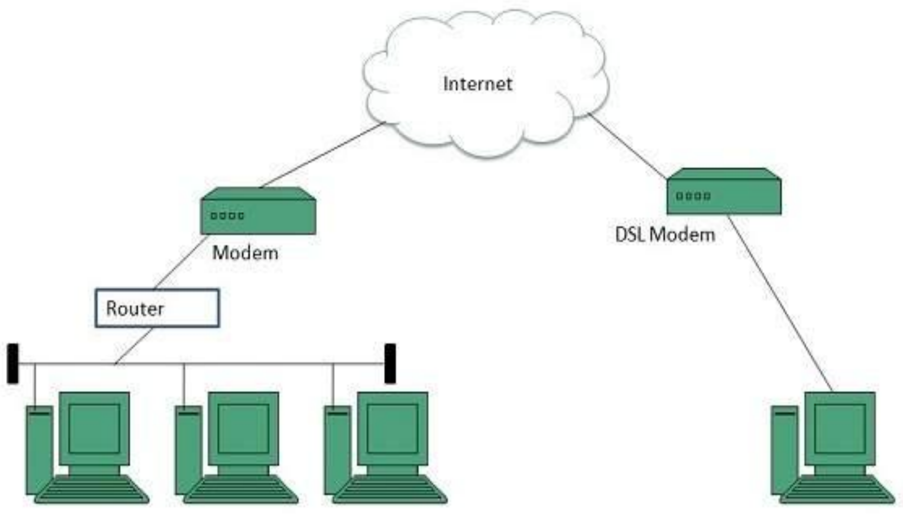
DSL

DSL is acronym of Digital Subscriber Line. It is a form of broadband connection as it provides connection over ordinary telephone lines.

- ✓ Following are the several versions of DSL technique available today:
- ✓ Asymmetric DSL (ADSL)
- ✓ Symmetric DSL (SDSL)
- ✓ High bit-rate DSL (HDSL)
- ✓ Rate adaptive DSL (RDSL)
- ✓ Very high bit-rate DSL (VDSL)
- ✓ ISDN DSL (IDSL)

All of the above mentioned technologies differ in their upload and download speed, bit transfer rate and level of service.

The following diagram shows that how we can connect to internet using DSL technology:



Internet Access Using DSL Modem

ASDL (Asymmetric digital subscriber line)

Asymmetric digital subscriber line (ADSL) is a type of digital subscriber line (DSL) technology, a data communications technology that enables faster data transmission over copper telephone lines than a conventional voice band modem can provide.

Advantages Of ADSL

- ✓ Faster downloads compared to dial-up or ISDN
- ✓ No need for a second phone line – by allowing voice and data transfer at the same time (you can use the phone as normal while connected to the internet).
- ✓ Because ADSL transfers data digitally it doesn't need to convert the data from digital to analogue and back again.
- ✓ ADSL connections are Always on, which makes the usual long wait to connect a thing of the past.

Disadvantages of ADSL

- ✓ ADSL connections are not available to everyone, you need to be within 3 miles of an ADSL enabled exchange.
- ✓ The hardware costs can be quite significant as you will need a special ADSL modem and ADSL filters to use the service, most ISPs allow you to hire these items which can reduce the initial cost.
- ✓ Because ADSL connections are Always on you will need a firewall to protect your PC

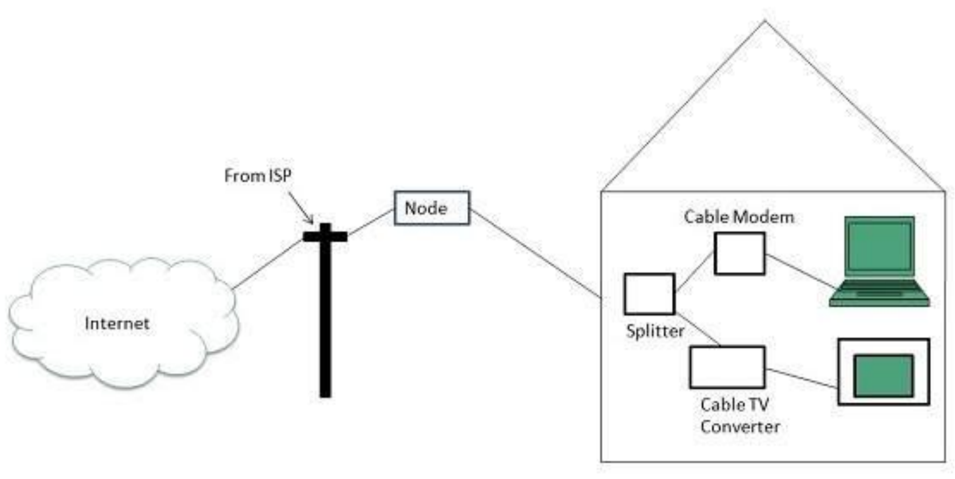
Cable TV Internet Connection

Cable TV Internet connection is provided through Cable TV lines. It uses coaxial cable which is capable of transferring data at much higher speed than common telephone line.

Key Points:

- ✓ A cable modem is used to access this service, provided by the cable operator.
- ✓ The Cable modem comprises of two connections: one for internet service and other for Cable TV signals.
- ✓ Since Cable TV internet connections share a set amount of bandwidth with a group of customers, therefore, data transfer rate also depends on number of customers using the internet at the same time.

The following diagram shows that how internet is accessed using Cable TV connection:



Leased lines.

WANs are often built using leased lines. These leased lines involve a direct point-to-point connection between two sites. Point-to-point WAN service may involve either analog dial-up lines or dedicated leased digital private lines.

Analog lines

A modem is used to connect the computer to the telephone line. Analog lines may be part of a public-switched telephone network and are suitable for batch data transmissions.

Dedicated lines

Digital phone lines that permit uninterrupted, secure transmission at fixed costs.At each end of the leased line, a router connects to the LAN on one side and a hub within the WAN on the other. Leased lines can get pretty expensive in the long run.

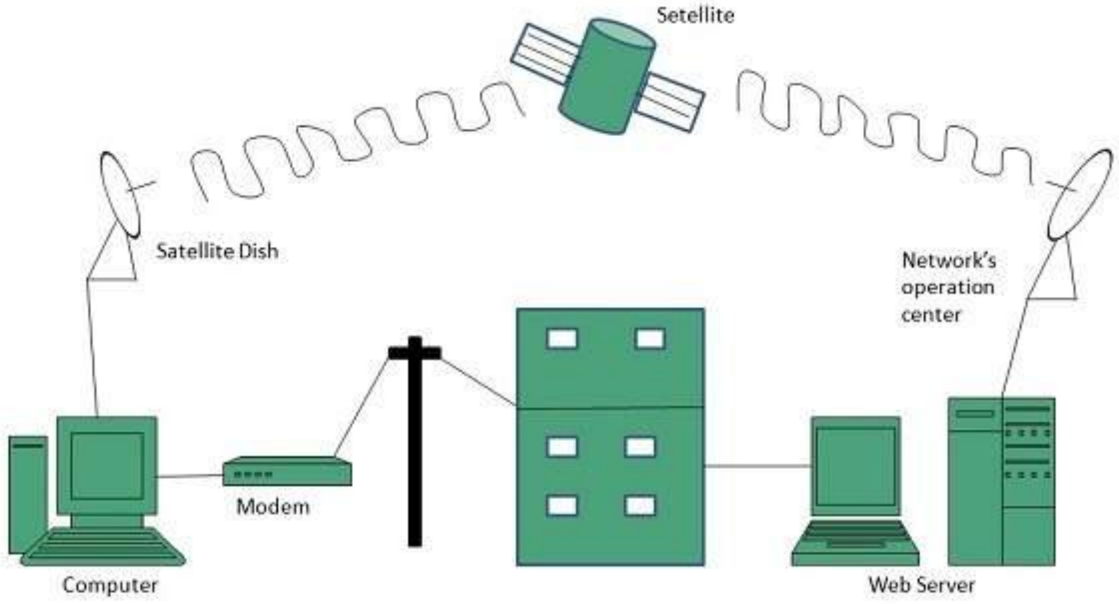
Satellite Internet Connection

Satellite Internet connection offers high speed connection to the internet. There are two types of satellite internet connection: one way connection or two way connection.

In one way connection, we can only download data but if we want to upload, we need a dialup access through ISP over telephone line.

In two way connection, we can download and upload the data by the satellite. It does not require any dialup connection.

The following diagram shows how internet is accessed using satellite internet connection:



Wireless Internet Connection

Wireless Internet Connection makes use of radio frequency bands to connect to the internet and offers a very high speed. The wireless internet connection can be obtained by either WiFi or Bluetooth.

Key Points:

- ✓ Wi Fi wireless technology is based on IEEE 802.11 standards which allow the electronic device to connect to the internet.
- ✓ Bluetooth wireless technology makes use of short-wavelength radio waves and helps to create personal area network (PAN).

Modem

Modem or **Broadband Modem** is a hardware device that connects a computer or router to a broadband network. For example, a Cable Modem and DSL Modem are two examples of these types of Modems.

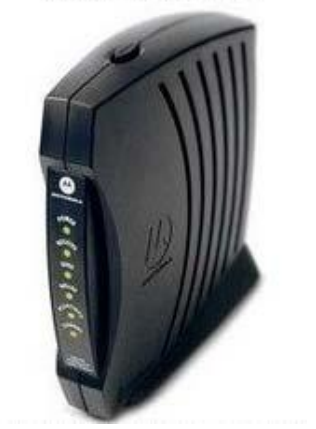
Short for **MODulator/DEModulator**, the first **Modem** known as the Dataphone was first released by AT&T in 1960. It later became more common for home users when Dennis Hayes and Dale Heatherington released the 80-103A Modem in 1977.

Cable Modem

A **cable modem** is a hardware device that allows your computer to communicate with an Internet Service Provider over a landline connection.

It converts an analog signal to a digital signal for the purpose of granting access to broadband Internet. A cable modem works by connecting a coaxial cable to a jack in the wall and then a Cat5 (Ethernet) cord from the modem to a computer or a network router.

Network routers are used to share your Internet connection between multiple computers.



The picture is an example of a traditional stand alone cable modem from Motorola, there are also all in one modems that have a modem and router built into one box.

If your modem only has one coaxial cable connection and one Cat5 connection, your modem is a stand alone modem and needs a router to share the connection.

Cable offers a significant speed increase in Internet performance when compared to a dial-upconnection and is one of the fastest broadband solutions. Comcast and Time Warner are examples of cable Internet providers in the United States.

Internet tools

OpenOffice : Free and works almost identically to Microsoft Office Suite, OpenOffice allows you to write documents, make spreadsheets, create presentations and more. You'll also have the option to save files as Microsoft documents and other popular formats.

Google Search : Google is the gatekeeper to the World Wide Web and the keyholder to all the answers to your questions. Type keywords and find the best answer possible. There's a reason why it's ranked the number one website in the world, you know.

Google Drive : Google Drive is an easy way to drag-and-store all of your documents onto a Google account. You can view documents anytime and anywhere with internet access. If you’re running an online business with multiple employees, use Google Drive to share documents and pass information.

Word Press : A free-to-use blogging platform. Create your own domain and have a website running in minutes. WordPress is arguably the go-to blogging website used by millions of businesses online. It’s free and available on Linux, PC, and MAC.

PowerPoint : Microsoft PowerPoint allows you to create amazing presentations. Here, you can embed videos, insert slides, and create amazing content with style.

Facebook : The world’s most popular social media website has over a billion users. It’s popular social network also carries great educational applications users may find helpful.

SlideShare : The world’s most popular slide sharing tool used by the world’s most popular web publishers. Create slide presentations from anywhere and embed them wherever and whenever you want for everyone to see.

Diigo : A social media site that gives users the luxury to save and annotate favorite content in one spot; and of course, share discoveries with like minds.

EduGlogster: Create and use interactive posters with ease. Insert multiple photos, embed videos, and view them in one interactive template.

Camtasia: Beam presentations to anyone in the world. Choose your mic, comment on images and websites and send them over to your friends, students, or teachers.

VoiceThread : Similar to written threads in forums, VoiceThread creates voice driven group discussions online. Post a lesson or project and recreate group discussions with ease.

Screenr : Instantly create screen casts with a push of a button. Click record and produce web-based commentary screen casts. Post them up and send them to your classmates, friends, or family.

iOS : If you have an iPhone or iPad, you’ll want to find internet tools in application form. With iOS you’ll be able to absorb content on-the-go as you would on a desktop.

Poll_Everywhere: Create polls anytime you want in real-time. Replace expensive hardware with good old fashion web technology. Social media coordinators love this one.

Excel: Create Microsoft Excel spreadsheets on any whim. Input, store and manage spreadsheet data like a boss with Mircrosoft excel.

paper.li: Publish all of the content you find online like a newspaper. Drag and drop your favorite stories and publish it how the New York Times would: Easy.

MentorMob : Build a learning playlist and have the luxury to catch up on tutorials or lessons anytime you want.

quora: This is like having Wikipedia, Yahoo! Answers and Facebook all in one page. Here you'll find an amazing community who offers great answers to all questions and vice versa.

Instapaper:Use Instapaper to save notes and content from various websites to read later. To help you read lessons later, save it on Instapaper

Web server

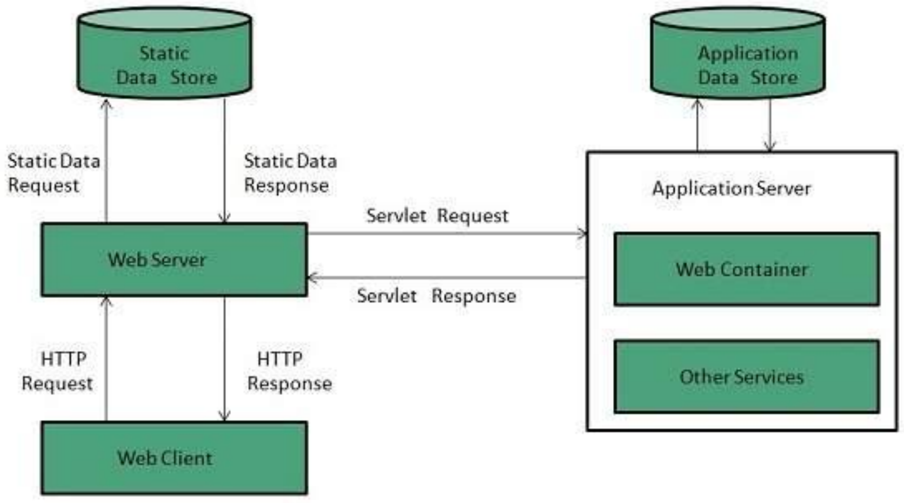
Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web site is collection of web pages while web server is a software that respond to the request for web resources.

Web Server Working

Web server respond to the client request in either of the following two ways:

- ✓ Sending the file to the client associated with the requested URL.
- ✓ Generating response by invoking a script and communicating with database



Key Points

- ✓ When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.

- ✓ If the requested web page is not found, web server will the send an HTTP response:**Error 404 Not found.**
- ✓ If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Examples

Following table describes the most leading web servers available today:

S.N.O	Web Server Description
1	**Apache HTTP Server** This is the most popular web server in the world developed by the Apache Software Foundation. .
2.	**Internet Information Services (IIS)** The Internet Information Server (IIS) is a high performance Web Server from Microsoft. .

Domain name

When **DNS** was not into existence, one had to download a **Host file** containing host names and their corresponding IP address. But with increase in number of hosts of internet, the size of host file also increased. This resulted in increased traffic on downloading this file. To solve this problem the DNS system was introduced.

Domain Name System helps to resolve the host name to an address. It uses a hierarchical naming scheme and distributed database of IP addresses and associated names

Domain Name System Architecture:

The Domain name system comprises of **Domain Names, Domain Name Space, Name Server** that have been described below:

Domain Names

Domain Name is a symbolic string associated with an IP address. There are several domain names available; some of them are generic such as **com, edu, gov, net** etc, while some country level domain names such as **au, in, za, us** etc.

The following table shows the **Generic** Top-Level Domain names:

Domain Name	Meaning
Com	Commercial business

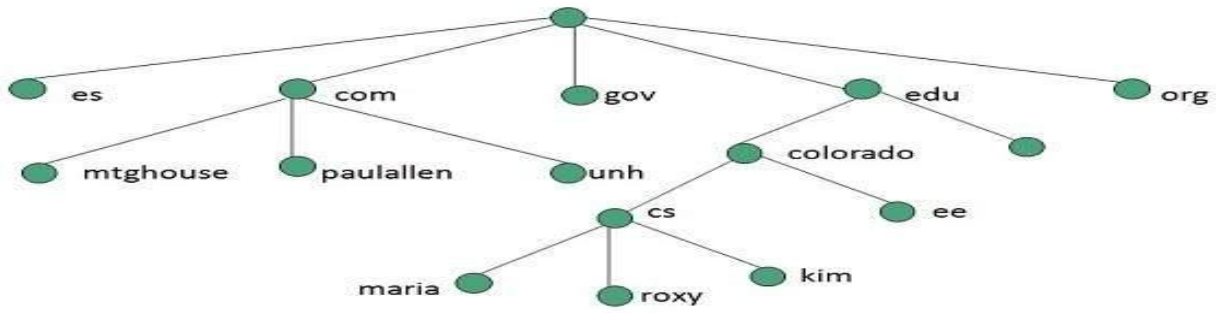
Edu	Education
Gov	U.S. government agency
Int	International entity
Mil	U.S. military
Net	Networking organization
Org	Non profit organization

The following table shows the **Country top-level** domain names:

Domain Name	Meaning
au	Australia
in	India
cl	Chile
fr	France
us	United States
za	South Africa
uk	United Kingdom
jp	Japan
es	Spain
de	Germany
ca	Canada
ee	Estonia
hk	Hong Kong

Domain Name Space

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. The following diagram shows the domain name space hierarchy:



In the above diagram each subtree represents a domain. Each domain can be partitioned into sub domains and these can be further partitioned and so on.

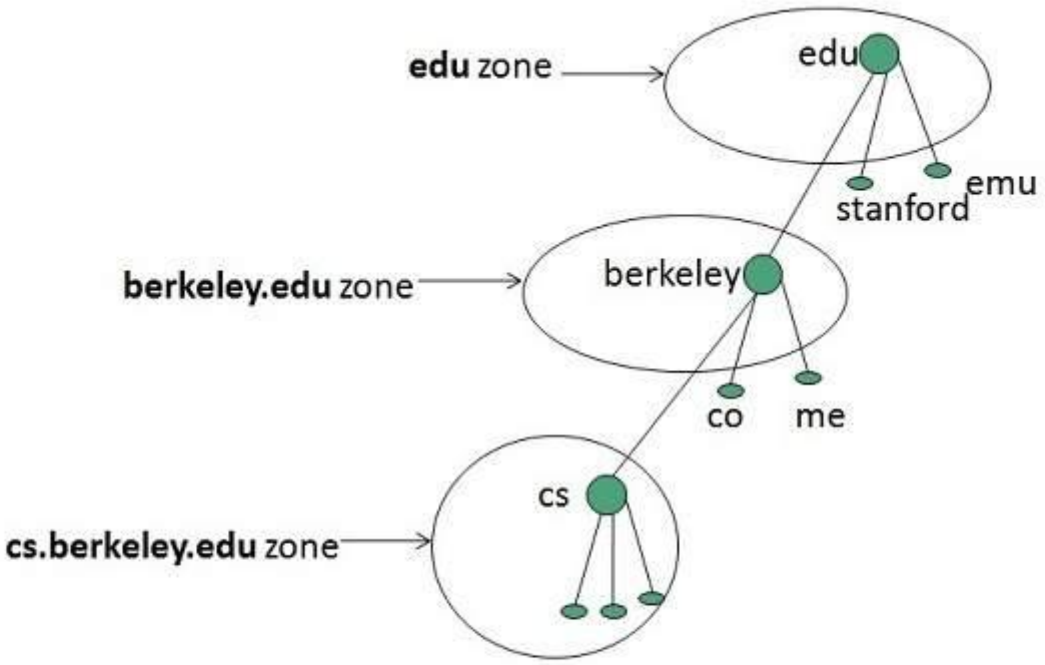
Name Server

Name server contains the DNS database. This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to maintain entire DNS database, therefore, the information is distributed among many DNS servers.

- ✓ Hierarchy of server is same as hierarchy of names.
- ✓ The entire name space is divided into the zones

Zones

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone.



If the domain is not further divided into sub domains then domain and zone refers to the same thing.

The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

1.1.1.1 TYPES OF NAME SERVERS

Following are the three categories of Name Servers that manages the entire Domain Name System:

1. Root Server
2. Primary Server
3. Secondary Server

1.1.1.1.1

1.1.1.1.2 ROOT SERVER

Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server

1.1.1.1.3 PRIMARY SERVERS

Primary Server stores a file about its zone. It has authority to create, maintain, and update the zone file.

1.1.1.1.4 SECONDARY SERVER

Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

DNS Working

DNS translates the domain name into IP address automatically. Following steps will take you through the steps included in domain resolution process:

- ✓ When we type **www.nptc.ac.in** into the browser, it asks the local DNS Server for its IP address.
- ✓ Here the local DNS is at ISP end.
- ✓ When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
- ✓ The root DNS server replies with delegation that **I do not know the IP address of www.nptc.ac.in but know the IP address of DNS Server.**
- ✓ The local DNS server then asks the com DNS Server the same question.
- ✓ The **in** DNS Server replies the same that it does not know the IP address of www.nptc.ac.in but knows the address of www.nptc.ac.in.

- ✓ Then the local DNS asks the www.nptc.ac.in DNS server the same question.
- ✓ Then www.nptc.ac.in DNS server replies with IP address of www.nptc.ac.in.
- ✓ Now, the local DNS sends the IP address of www.nptc.ac.in to the computer that sends the request.

IP Address

IP address is a unique logical address assigned to a machine over the network. An IP address exhibits the following properties:

- ✓ IP address is the unique address assigned to each host present on Internet.
- ✓ IP address is 32 bits (4 bytes) long.
- ✓ IP address consists of two components: **network component** and **host component**.
- ✓ Each of the 4 bytes is represented by a number from 0 to 255, separated with dots. For example 137.170.4.124

IP address is 32-bit number while on the other hand domain names are easy to remember names. For example, when we enter an email address we always enter a symbolic string such as webmaster@tutorialspoint.com.

Uniform Resource Locator (URL)

Uniform Resource Locator (URL) refers to a web address which uniquely identifies a document over the internet.

This document can be a web page, image, audio, video or anything else present on the web.

For example, www.nptc.ac.in/internet_technology/index.htmlis an URL to the index.html which is stored on tutorialspoint web server under internet_technology directory.

URL Types

There are two forms of URL as listed below:

- ✓ Absolute URL
- ✓ Relative URL

1.1.1.2

1.1.1.3 ABSOLUTE URL

Absolute URL is a complete address of a resource on the web. This completed address comprises of protocol used, server name, path name and file name.

For example http:// www.nptc.ac.in / internet_technology /index.htm. where:

- ✓ **http** is the protocol.

- ✓ **tutorialspoint.com** is the server name.
- ✓ **index.htm** is the file name.

The protocol part tells the web browser how to handle the file. Similarly we have some other protocols also that can be used to create URL are:

- FTP
- https
- Gopher
- mailto
- news

1.1.1.4 RELATIVE URL

Relative URL is a partial address of a webpage. Unlike absolute URL, the protocol and server part are omitted from relative URL.

Relative URLs are used for internal links i.e. to create links to file that are part of same website as the WebPages on which you are placing the link.

For example, to link an image on Nptc.ac.in/internet_technology/internet_referemce_models, we can use the relative URL which can take the form like **/internet_technologies/internet-osi_model.jpg.**

Difference between Absolute and Relative URL

Absolute URL	**Relative URL**
Used to link web pages on different websites	Used to link web pages within the same website.
Difficult to manage.	Easy to Manage
Changes when the server name or directory name changes	Remains same even of we change the server name or directory name.
Take time to access	Comparatively faster to access.

Search Engine

Search Engine refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web.

User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user.

Search Engine Components

Generally there are three basic components of a search engine as listed below:

- ✓ Web Crawler
- ✓ Database
- ✓ Search Interfaces

Web crawler

It is also known as **spider** or **bots.** It is a software component that traverses the web to gather information.

Database

All the information on the web is stored in database. It consists of huge web resources.

Search Interfaces

This component is an interface between user and the database. It helps the user to search through the database.

Search Engine Working

Web crawler, database and the search interface are the major component of a search engine that actually makes search engine to work. Search engines make use of Boolean expression AND, OR, NOT to restrict and widen the results of a search. Following are the steps that are performed by the search engine:

- ✓ The search engine looks for the keyword in the index for predefined database instead of going directly to the web to search for the keyword.
- ✓ It then uses software to search for the information in the database. This software component is known as web crawler.
- ✓ Once web crawler finds the pages, the search engine then shows the relevant web pages as a result. These retrieved web pages generally include title of page, size of text portion, first several sentences etc.

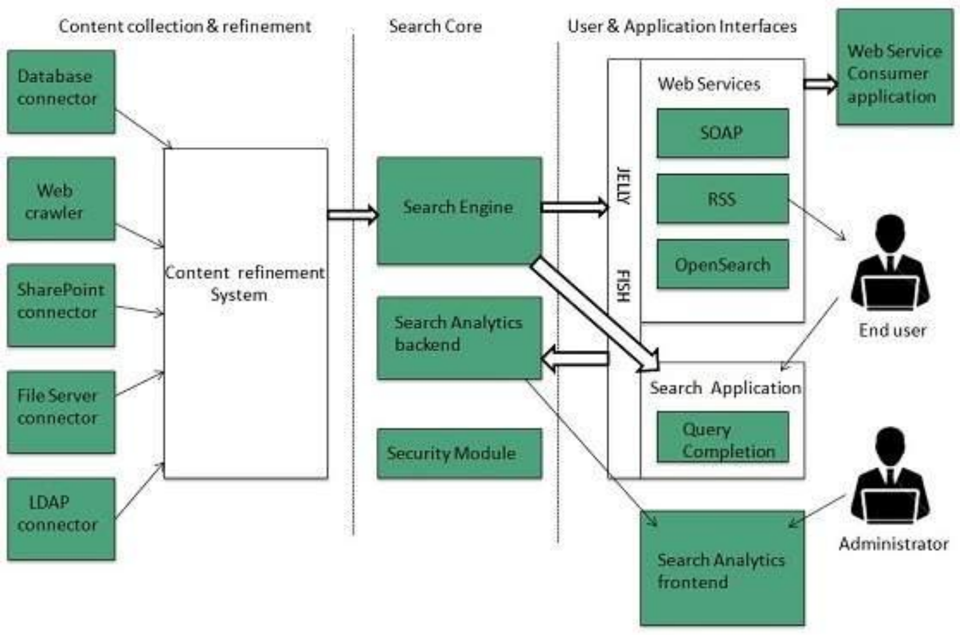
These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

- ✓ User can click on any of the search results to open it.

Architecture:

The search engine architecture comprises of the three basic layers listed below:

- ✓ Content collection and refinement.
- ✓ Search core
- ✓ User and application interfaces



Search Engine Processing

Indexing Process

Indexing process comprises of the following three tasks:

- Text acquisition
- Text transformation
- Index creation

1.1.1.5

1.1.1.6 TEXT ACQUISITION

It identifies and stores documents for indexing.

1.1.1.7 TEXT TRANSFORMATION

It transforms document into index terms or features.

1.1.1.8 INDEX CREATION

It takes index terms created by text transformations and create data structures to support fast searching.

Query Process

Query process comprises of the following three tasks:

- User interaction
- Ranking
- Evaluation

1.1.1.9

1.1.1.10 USER INTERACTION

It supports creation and refinement of user query and displays the results.

1.1.1.11 RANKING

It uses query and indexes to create ranked list of documents.

1.1.1.12 EVALUATION

It monitors and measures the effectiveness and efficiency. It is done offline.

Examples

Following are the several search engines available today:

Search Engine	**Description**
Google	It was originally called **BackRub.** It is the most popular search engine globally.
Bing	It was launched in 2009 by **Microsoft.** It is the latest web-based search engine that also delivers Yahoo's results.
Ask	It was launched in 1996 and was originally known as **Ask Jeeves.** It includes support for match, dictionary, and conversation question.
AltaVista	It was launched by **Digital Equipment Corporation** in 1995. Since 2003, it is powered by Yahoo technology.
AOL.Search	It is powered by Google.
LYCOS	It is top 5 internet portal and 13th largest online property according to Media Matrix.
Alexa	It is subsidiary of Amazon and used for providing website traffic information.

Web Browser

Web Browser is an application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar.Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.

Earlier the web browsers were text-based while now a days graphical-based or voice-based web browsers are also available. Following are the most common web browser available today:

Browser	**Vendor**
Internet Explorer	Microsoft
Google Chrome	Google
Mozilla Firefox	Mozilla
Netscape Navigator	Netscape Communications Corp.
Opera	Opera Software
Safari	Apple
Sea Monkey	Mozilla Foundation
K-meleon	K-meleon

Architecture

There are a lot of web browsers available in the market. All of them interpret and display information on the screen however their capabilities and structure varies depending upon implementation. But the most basic component that all web browser must exhibit are listed below:

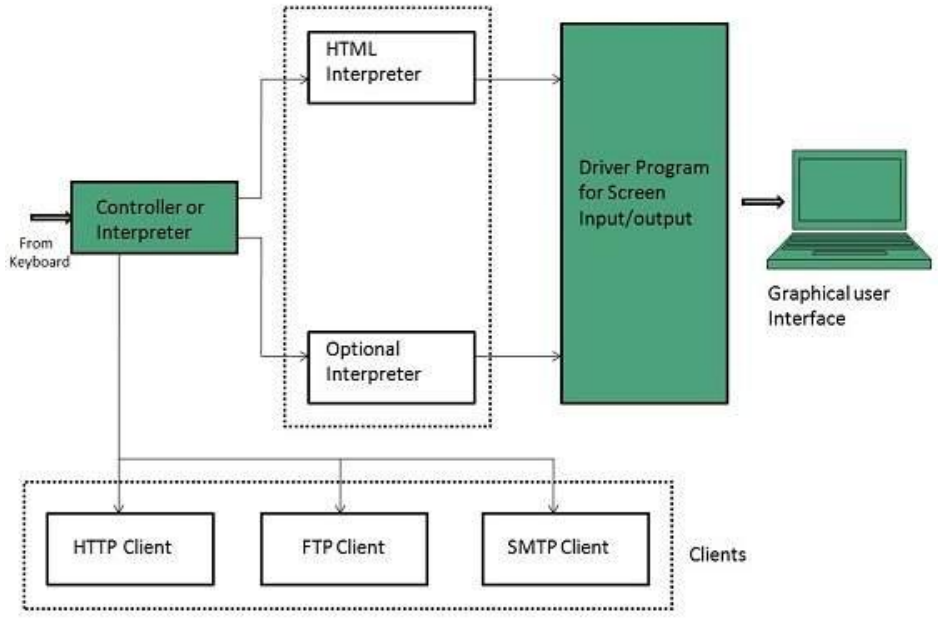
- Controller/Dispatcher
- Interpreter
- Client Programs

Controller works as a control unit in CPU. It takes input from the keyboard or mouse, interpret it and make other services to work on the basis of input it receives.

Interpreter receives the information from the controller and execute the instruction line by line. Some interpreter are mandatory while some are optional For example, HTML interpreter program is mandatory and java interpreter is optional.

Client Program describes the specific protocol that will be used to access a particular service. Following are the client programs tat are commonly used:

- HTTP
- SMTP
- FTP
- NNTP
- POP

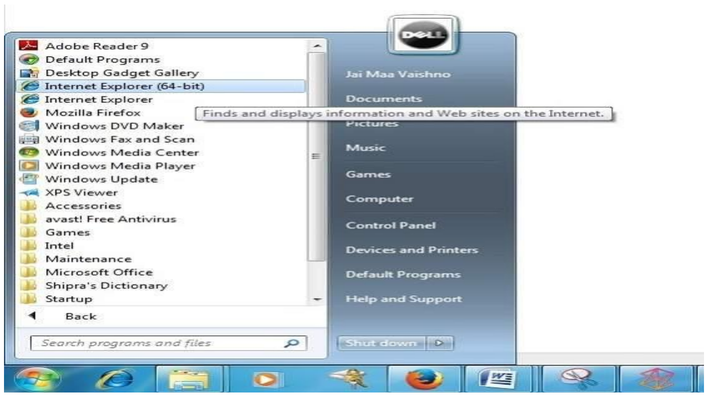


Starting Internet Explorer

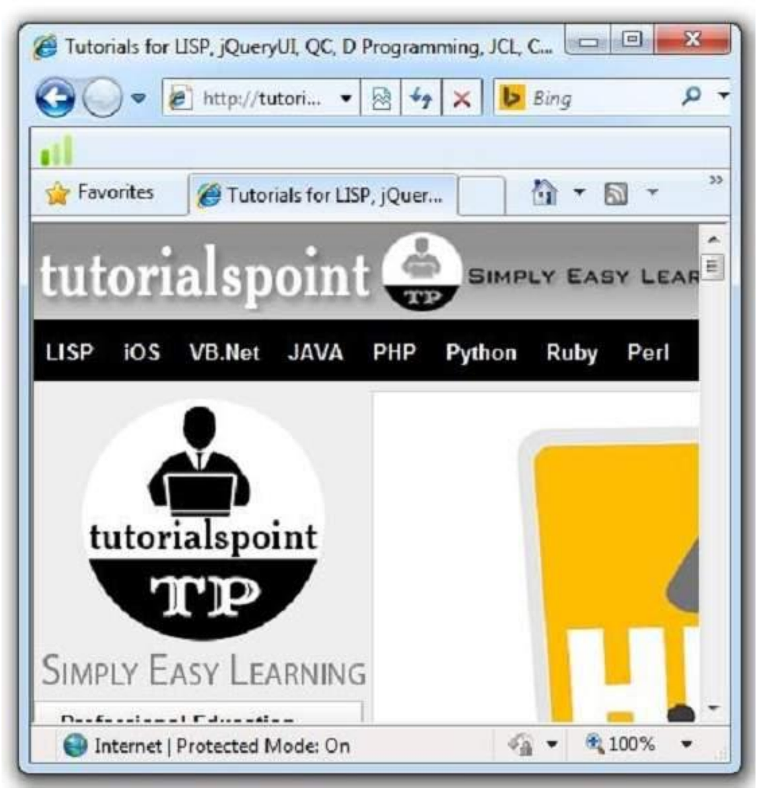
Internet explorer is a web browser developed by Microsoft. It is installed by default with the windows operating system however, it can be downloaded and be upgraded.

To start internet explorer, follow the following steps:

- Go to **Start** button and click **Internet Explorer.**



The **Internet Explorer** window will appear as shown in the following diagram:



Accessing Web Page

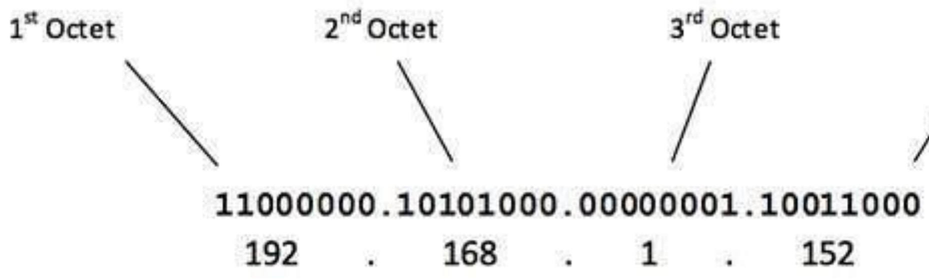
Accessing web page is very simple. Just enter the **URL** in the address bar as shown the following diagram:

IP Addresses

Internet Protocol hierarchy contains several classes of IP Addresses to be used efficiently in various situations as per the requirement of hosts per network. Broadly, the IPv4 Addressing system is divided into five classes of IP Addresses. All the five classes are identified by the first octet of IP Address.

Internet Corporation for Assigned Names and Numbers is responsible for assigning IP addresses.

The first octet referred here is the left most of all. The octets numbered as follows depicting dotted decimal notation of IP Address:

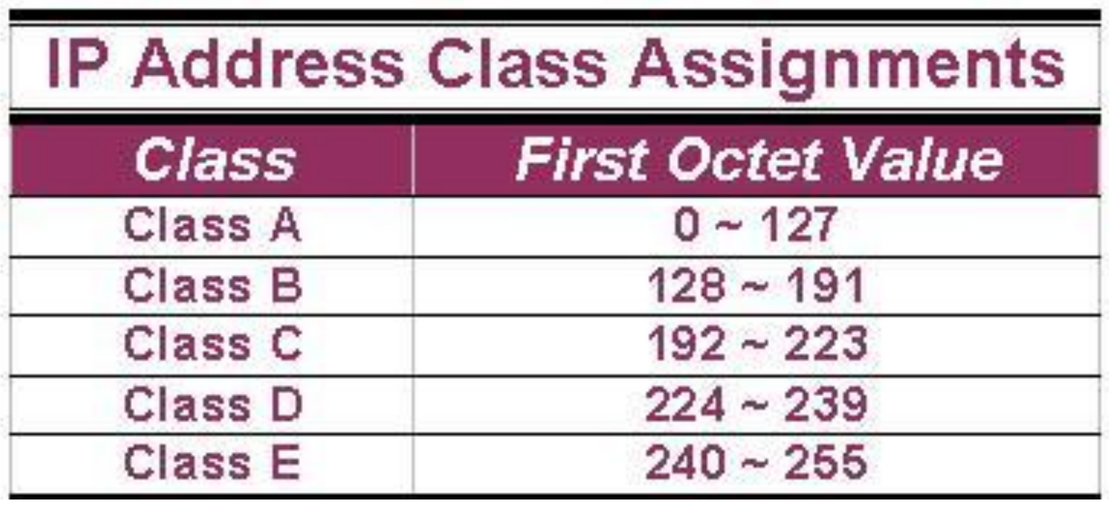


The number of networks and the number of hosts per class can be derived by this formula:

Number of networks	= 2^network_bits
Number of Hosts/Network	= 2^host_bits – 2

When calculating hosts' IP addresses, 2 IP addresses are decreased because they cannot be assigned to hosts, i.e. the first IP of a network is network number and the last IP is reserved for Broadcast IP.

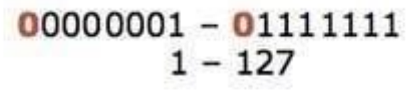
The 5 IP classes are split up based on the value in the 1st octet:



IP Address Class Assignments	
Class	**First Octet Value**
Class A	0 ~ 127
Class B	128 ~ 191
Class C	192 ~ 223
Class D	224 ~ 239
Class E	240 ~ 255

Class A Address

The first bit of the first octet is always set to 0 (zero). Thus the first octet ranges from 1 – 127, i.e.



00000001 - 01111111
1 - 127

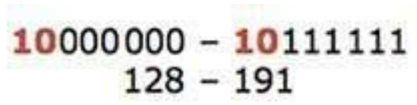
Class A addresses only include IP starting from 1.x.x.x to 126.x.x.x only. The IP range 127.x.x.x is reserved for loopback IP addresses.

The default subnet mask for Class A IP address is 255.0.0.0 which implies that Class A addressing can have 126 networks (2^7-2) and 16777214 hosts (2^{24}-2).

Class A IP address format is thus: **0NNNNNNN**.HHHHHHHH.HHHHHHHH.HHHHHHHH

Class B Address

An IP address which belongs to class B has the first two bits in the first octet set to 10, i.e.



10000000 - 10111111
128 - 191

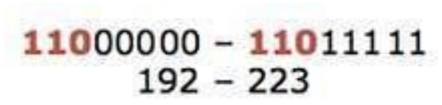
Class B IP Addresses range from 128.0.x.x to 191.255.x.x. The default subnet mask for Class B is 255.255.x.x.

Class B has 16384 (2^{14}) Network addresses and 65534 (2^{16}-2) Host addresses.

Class B IP address format is: **10NNNNNN.NNNNNNNN**.HHHHHHHH.HHHHHHHH

Class C Address

The first octet of Class C IP address has its first 3 bits set to 110, that is:


11000000 - 11011111
192 - 223

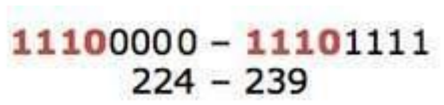
Class C IP addresses range from 192.0.0.x to 223.255.255.x. The default subnet mask for Class C is 255.255.255.x.

Class C gives 2097152 (2^{21}) Network addresses and 254 (2^{8}-2) Host addresses.

Class C IP address format is: **110NNNNN.NNNNNNNN.NNNNNNNN**.HHHHHHHH

Class D Address

Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of:


11100000 - 11101111
224 - 239

Class D has IP address rage from 224.0.0.0 to 239.255.255.255. Class D is reserved for Multicasting. In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask.

Class E Address

This IP Class is reserved for experimental purposes only for R&D or Study. IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254. Like Class D, this class too is not equipped with any subnet mask.

Internet Protocol version 4

The Internet Protocol version 4 was designed to be allocated to approx. imately 4.3 billion addresses. At the beginning of Internet this was considered a much wider address space for which there was nothing to worry about.

The sudden growth in internet users and its wide spread use has exponentially increased the number of devices which needs real and unique IP to be able to communicate. Gradually, an IPS is required by almost every digital equipment which were made to ease human life, such as Mobile Phones, Cars and other electronic devices. The number of devices (other than computers/routers) expanded the demand for extra IP addresses, which were not considered earlier.

Allocation of IPv4 is globally managed by Internet Assigned Numbers Authority (IANA) under coordination with the Internet Corporation for Assigned Names and Numbers (ICANN). IANA works closely with Regional Internet Registries, which in turns are responsible for efficiently distributing IP addresses in their territories. There are five such RIRS. According to IANA reports, all the IPv4 address blocks have been allocated. To cope up with the situation, the following practices were being done:

- **Private IPs:** Few blocks of IPs were declared for private use within a LAN so that the requirement for public IP addresses can be reduced.
- **NAT:** Network address translation is a mechanism by which multiple PCs/hosts with private IP addresses are enabled to access using one or few public IP addresses.
- Unused Public IPs were reclaimed by RIRs.

Internet Protocol v6 (IPv6)

IETF (Internet Engineering Task Force) has redesigned IP addresses to mitigate the drawbacks of IPv4. The new IP address is version 6 which is 128-bit address, by which every single inch of the earth can be given millions of IP addresses.

Today majority of devices running on Internet are using IPv4 and it is not possible to shift them to IPv6 in the coming days. There are mechanisms provided by IPv6, by which IPv4 and IPv6 can co-exist unless the Internet entirely shifts to IPv6:

- Dual IP Stack
- Tunneling (6to4 and 4to6)
- NAT Protocol Translation

Internet Protocols

TCP / IP

TCP / IP is a suite, or family, of protocols that govern the way data is transmitted across networks. TCP / IP protocols work together to break the data into small pieces that can be efficiently handled by the network, communicate the destination of the data to the network, verify the receipt of the data on the other end of the transmission, and reconstruct the data in its original form.

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

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

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

Many Internet users are familiar with the even higher layer application protocols that use TCP/IP to get to the Internet. These include the World Wide Web's Hypertext Transfer Protocol (HTTP), the File Transfer Protocol (FTP), Telnet (Telnet) which lets you logon to remote computers, and the Simple Mail Transfer Protocol (SMTP). These and other protocols are often packaged together with TCP/IP as a "suite."

Personal computer users with an analog phone modem connection to the Internet usually get to the Internet through the Serial Line Internet Protocol (SLIP) or the Point-to-Point Protocol (PPP). These protocols encapsulate the IP packets so that they can be sent over the dial-up phone connection to an access provider's modem.

Protocols related to TCP/IP include the User Datagram Protocol (UDP), which is used instead of TCP for special purposes. Other protocols are used by network host computers for exchanging router information. These include the Internet Control Message Protocol (ICMP), the Interior Gateway Protocol (IGP), the Exterior Gateway Protocol (EGP), and the Border Gateway Protocol (BGP).

FTP

FTP (File Transfer Protocol) is the protocol, or set of rules, which enables files to be transferred from one computer to another. It is part of the TCP / IP protocol suite. Files that are available for FTP are stored on computers called FTP servers. An FTP client program is an interface that allows the user to locate the file(s) to be transferred and initiate the transfer process. It is a good idea to have current virus checking software and compression / decompression software before downloading files. Through anonymous FTP, users have access to many types of files including shareware, freeware, upgrades and documents.

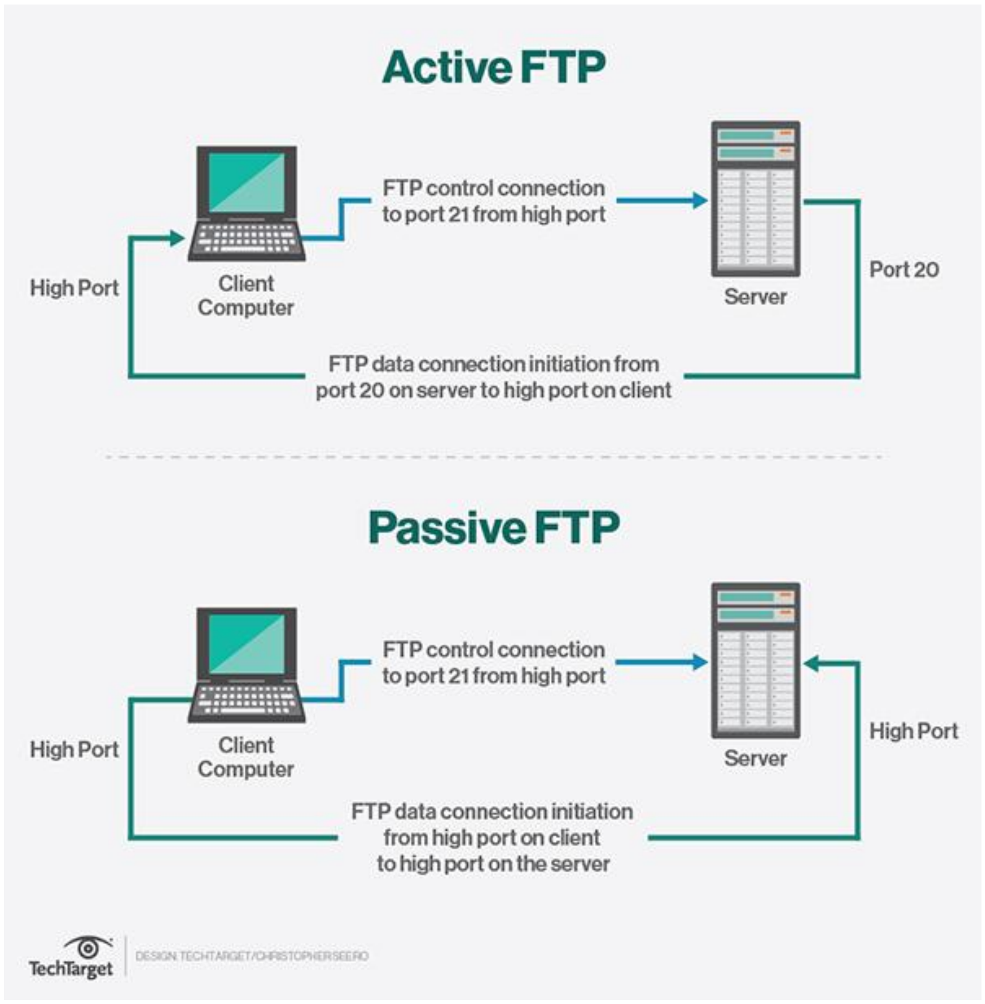
File Transfer Protocol (FTP) is a client-server standard used to transfer files between computers over the Internet using control and data channels.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet over TCP/IP connections.

FTP is a client-server protocol that relies on two communications channels between client and server: a command channel for controlling the conversation and a data channel for transmitting file content. Clients initiate conversations with servers by requesting to download a file. Using FTP, a client can upload, download, delete, rename, move and copy files on a server. A user typically needs to log on to the FTP server, although some servers make some or all of their content available without login, also known as anonymous FTP.

FTP sessions work in passive or active modes. In active mode, after a client initiates a session via a command channel request, the server initiates a data connection back to the client and begins transferring data. In passive mode, the server instead uses the command channel to send the client the information it needs to open a data channel. Because passive mode has the client initiating all connections, it works well across firewalls and Network Address Translation (NAT) gateways.


Active FTP
FTP control connection to port 21 from high port
High Port
Client Computer
Server
Port 20
FTP data connection initiation from port 20 on server to high port on client
Passive FTP
FTP control connection to port 21 from high port
High Port
Client Computer
Server
High Port
FTP data connection initiation from high port on client to high port on the server
TechTarget
DESIGN TECHTARGET/CHRISTOPHER SEERO

Active FTP and passive FTP compared

FTP was originally defined in 1971, prior to the definition of TCP and IP, and has been redefined many times -- e.g., to use TCP/IP (RFC 765 and RFC 959), and then Internet Procotol Version 6 (IPv6), (RFC 2428). Also, because it was defined without much concern for security, it has been extended many times to improve security: for example, versions that encrypt via a TLS connection (FTPS) or that work with Secure File Transfer Protocol (SFTP), also known as SSH File Transfer Protocol.

Users can work with FTP via a simple command line interface (for example, from a console or terminal window in Microsoft Windows, Apple OS X or Linux) or with a dedicated graphical user interface (GUI). Web browsers can also serve as FTP clients.

Although a lot of file transfer is now handled using HTTP, FTP is still commonly used to transfer files "behind the scenes" for other applications -- e.g., hidden behind the user interfaces of banking, a service that helps build a website, such as Wix or SquareSpace, or other services. It is also used, via Web browsers, to download new applications.

HTTP

HTTP is an acronym for Hypertext Transfer Protocol. HTTP is the set of rules, or protocol, that enables hypertext data to be transferred form one computer to another, and is based on the client / server principle. Hypertext is text that is coded using the Hypertext is text that is coded using the Hypertext Markup Language (HTML). HTTP enables users to retrieve a wide variety of resources such as text, graphics, sound, animation and other hypertext documents, and allows hypertext access to other Internet protocols.

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet).

HTTP concepts include (as the Hypertext part of the name implies) the idea that files can contain references to other files whose selection will elicit additional transfer requests. Any Web server machine contains, in addition to the Web page files it can serve, an HTTP daemon, a program that is designed to wait for HTTP requests and handle them when they arrive. Your Web browser is an HTTP client, sending requests to server machines. When the browser user enters file requests by either "opening" a Web file (typing in a Uniform Resource Locator or URL) or clicking on a hypertext link, the browser builds an HTTP request and sends it to the Internet Protocol address (IP address) indicated by the URL. The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request. (A Web page often consists of more than one file.)

The latest version of HTTP is HTTP 1.1.

TelNet:

Telnet is the protocol which enables one computer to establish a connection to another computer. The computer establishing the connection is referred to as the local computer; the computer accepting the connection is referred to as the remote, or host, computer. Although some computer may require an account and password, many computers allow users to access resources stored on them without an account and a password. Telnet can provide access to many resources around the world, such as library catalogs, databases, and other Internet tools and applications.

What is Telnet?

Telnet is a user command and an underlying TCP/IP protocol for accessing remote computers. Through Telnet, an administrator or another user can access someone else's computer remotely. On the Web, HTTP and FTP protocols allow you to request specific files from remote computers, but not to actually be logged on as a user of that computer. With Telnet, you log on as a regular user with whatever privileges you may have been granted to the specific application and data on that computer.

A Telnet command request looks like this (the computer name is made-up):

```
telnet the.libraryat.whatis.edu
```

The result of this request would be an invitation to log on with a userid and a prompt for a password. If accepted, you would be logged on like any user who used this computer every day.

Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer.

Gopher

Gopher is a protocol designed to search, retrieve, and display documents from remote sites on the Internet. In addition to document display, document retrieval, it is possible to initiate on-line connections with other systems via Gopher. Information accessible via Gopher is stored on many computers all over the Internet called Gopher servers. Gopher works on the client / server model and to retrieve and search the information stored on the gopher servers, you need to run a Gopher client application on your computer. Gopher can work with many other Internet protocols.

WAIS

WAIS is an Internet search tool that has the capability of searching many databases at one time. The databases to be searched can be determined by the user. When WAIS completes a search, it is actually searching an index of the database. A WAIS database index is created by a person. WAIS retrieves all items from the chosen databases that contain any of the words in the search phrase, provided that the words in the search phrase appear in the indexes of the selected databases. A relevancy ranking is assigned to each retrieved item to help the user determine which items may be most useful. WAIS can be accessed via Telnet, Gopher or a WAIS client program, and increasingly WAIS indexed databases are accessible through the World Wide Web. Web users can use WAIS by either downloading a WAIS client and a "gateway" to the Web browser or by using Telnet to connect to a public WAIS client.

WAIS (Wide Area Information Servers) is an Internet system in which specialized subject databases are created at multiple server locations, kept track of by a directory of servers at one location, and made accessible for searching by users with **WAIS** client programs. The user of WAIS is provided with or obtains a list of distributed database s. The user enters a search argument for a selected database and the client then accesses all the servers on which the database is distributed. The results provide a description of each text that meets the search requirements. The user can then retrieve the full text.

Because of the abundance of content and search engines now available on the Web, few if any WAIS servers remain in operation.

GPRS

General Packet Radio Service (**GPRS**) is a packet-based mobile data service on the global system for mobile communications (GSM) of 3G and 2G cellular communication systems.

A packet-switched technology that enables data communications.

- ✓ GPRS is used for various data applications on phones, including wireless Internet (WAP), MMS, and software that connects to the Internet. Basically, any network connection that is not voice or text messaging uses a data connection like GPRS.
- ✓ GPRS offers a tenfold increase in data speed over previous (circuit-switched) technologies, up to 115kbit/s (in theory). Typical real-world speeds are around 30-40 Kbps.

General Packet Radio Services (GPRS) is a packet-based wireless communication service that promises data rates from 56 up to 114 Kbps and continuous connection to the Internet for mobile phone and computer users. The higher data rates allow users to take part in video conferences and interact with multimedia Web sites and similar applications using mobile handheld devices as well as notebook computers. GPRS is based on Global System for Mobile (GSM) communication and complements existing services such circuit-switched cellular phone connections and the Short Message Service (SMS).

In theory, GPRS packet-based services cost users less than circuit-switched services since communication channels are being used on a shared-use, as-packets-are-needed basis rather than dedicated to only one user at a time. It is also easier to make applications available to mobile users because the faster data rate means that middleware currently needed to adapt applications to the slower speed of wireless systems are no longer be needed. As GPRS has become more widely available, along with other 2.5G and 3G services, mobile users of virtual private networks (VPNs) have been able to access the private network continuously over wireless rather than through a rooted dial-up connection.

GPRS also complements Bluetooth, a standard for replacing wired connections between devices with wireless radio connections. In addition to the Internet Protocol (IP), GPRS supports X.25, a packet-based protocol that is used mainly in Europe. GPRS is an evolutionary step toward Enhanced Data GSM Environment (EDGE) and Universal Mobile Telephone Service (UMTS).

Newer technologies like EDGE and 3G are much faster.

Using a packet switching, subscribers are always connected and always on-line, so services will be easy and quick to access. GPRS is considered a "2.5G" technology, meaning it is more advanced than standard 2G digital technology, but does not meet the requirements of a full-fledged 3G technology.

EDGE (Enhanced Data for Global Evolution):

- ✓ An upgrade for GSM/GPRS networks that triples data rates (speed) over standard GPRS.
- ✓ EDGE is used automatically when both the phone and network support it. EDGE phones will automatically revert to the slower GPRS standard when EDGE service is not available.
- ✓ Although many EDGE phones and devices are theoretically capable of up to 236 Kbps, most EDGE networks are only configured to allow up to 135 Kbps, to conserve spectrum resources. Real-world data rates are usually lower than the maximum.
- ✓ Because it is based on existing GSM and GPRS technology, EDGE is a smooth upgrade for GSM network operators.
- ✓ Although EDGE works at a low level within the GSM standard that includes voice, the main benefit is to increase GPRS data rates. GPRS operating over EDGE is called EGPRS.
- ✓ Although EDGE is faster than GPRS, it is not as fast as 3G technologies such as HSDPA and EVDO.

2G, 2.5G, 2.75G, 3G, 4G

Short-form for generations of wireless technologies:

- 2G stands for second generation primarily meaning GSM only;
- 2.5G primarily means GSM and GPRS;
- 2.75G was used to refer to EDGE (Enhanced Data Rates for GSM Evolution) Networks;
- 3G means third generation (W-CDMA) technology;
- 4G includes the Mobile WiMAX, and the first-release Long-Term Evolution (LTE) standards.

What 5G will consist of is not yet clearly established, and it is unlikley to be deployed before the 2020s. The EU is sponsoring projects to better define and develop 5G including METIS, 5GNOW, iJOIN, TROPIC, Mobile Cloud Networking, COMBO, MOTO and PHYLAWS.

2.75G is a term occasionally used to refer to EDGE data connectivity, implying that is is faster than GPRS (sometimes called 2.5G), but slower than typical 3G networks. The truth of the matter, however, is that EDGE is an official ITU ratified 3G technology.

G in **2G**, 3G and 4G stands for the "Generation" of the mobile **network**. Today, mobile operators have started offering 4G services in the country. A higher number before the 'G' means more power to send out and receive more information and therefore the ability to achieve a higher efficiency through the wireless **network**.

What are **1G**, **2G**, **3G** and **4G** networks ? ... The "G" in wireless networks refers to the "generation" of the underlying wireless network technology. Technically generations are defined as follows: **1G** networks (NMT, C-Nets, AMPS, TACS) are considered to be the first analog cellular systems, which started early 1980s.

The term "**3G** internet" refers to the third generation of mobile phone standards as set by the International Telecommunications Union (ITU). **3G** technologies allow mobile operators to offer more service options to their users, including mobile broadband.

Download speeds compare on 2G, 3G and 4G

Generation	**Technology**	**Maximum Download Speed**
2G	HSPA	7.2 Mbit/s
3G	HSPA+	21Mbit/s
	DC-HSPA+	42Mbit/s
4G	LTE	100Mbit/s

Meaning of 3g spectrum

For example, in the case of a voice signal having a minimum **frequency** of 200 hertz (Hz) and a maximum **frequency** of 3,000 Hz, the bandwidth is 2,800 Hz (3 KHz). The amount of bandwidth needed for **3G** services could be as much as 15-20 Mhz, whereas for 2G services a bandwidth of 30-200 KHz is used.Sep 29, 2006

1G (or 1-G) refers to the first generation of wireless telephone **technology** (mobile telecommunications). ... The main difference between the two mobile telephone systems (**1G**

and 2G), is that the radio signals used by **1G** networks are analog, while 2G networks are digital.

Difference between 2G,3G and 4G

G in **2G**, **3G** and **4G** stands for the “Generation” of the mobile network. Today, mobile operators have started offering **4G** services in the country. A higher number before the 'G' means more power to send out and receive more information and therefore the ability to achieve a higher efficiency through the wireless network.

1.2 Introduction to HTML

HTML stands for **H**yper**t**ext **M**arkup **L**anguage, and it is the most widely used language to write Web Pages.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus the link available on a webpage are called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

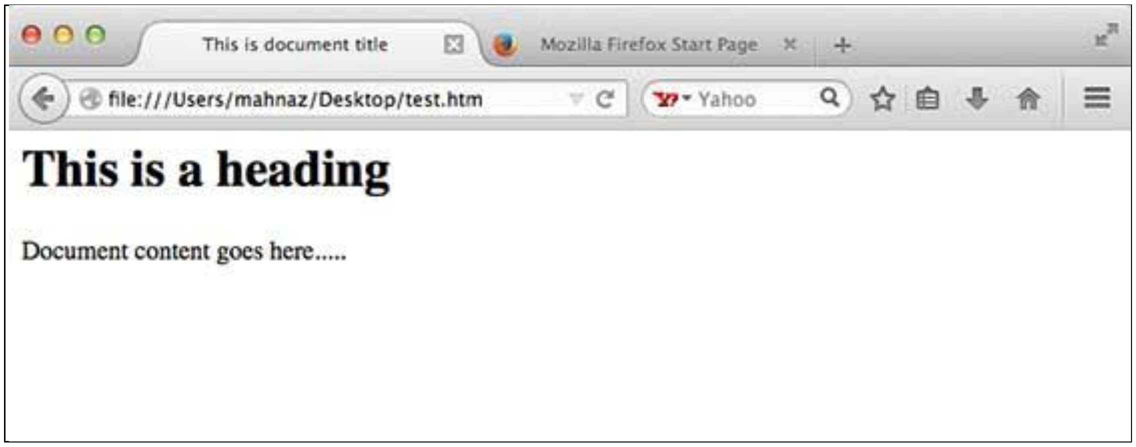
Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

Basic HTML Document

In its simplest form, following is an example of an HTML document:

```
<!DOCTYPE html>

<html>

<head>

<title>This is document title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>Document content goes here.....</p>

</body> </html>
```

Either you can use **Try it** option available at the top right corner of the code box to check the result of this HTML code, or let's save it in an HTML file **test.htm** using your favorite text editor. Finally open it using a web browser like Internet Explorer or Google Chrome, or Firefox etc. It must show the following output:


This is document title
Mozilla Firefox Start Page
file:///Users/mahnaz/Desktop/test.htm
Yahoo

This is a heading

Document content goes here.....

HTML Tags

As told earlier, HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example **<html>**has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

Above example of HTML document uses following tags:

Tag	**Description**
<!DOCTYPE...>	This tag defines the document type and HTML version.
<html>	This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
<head>	This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
<title>	The <title> tag is used inside the <head> tag to mention the document title.
<body>	This tag represents the document's body which keeps other HTML tags like <h1>, <div>, <p> etc.
<h1>	This tag represents the heading.
<p>	This tag represents a paragraph.

To learn HTML, you will need to study various tags and understand how do they behave while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful webpage.

World Wide Web Consortium (W3C) recommends to use lowercase tags starting from HTML 4.

HTML Document Structure

A typical HTML document will have following structure:

```
Document declaration tag

<html>
<head>
    Document header related tags
</head>
<body>
    Document body related tags
</body>
</html>
```

We will see what is document declaration tag.

The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration:

```
<!DOCTYPE html>
```

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used. We will see more details on this while discussing <!DOCTYPE...> tag along with other HTML tags.

Headers

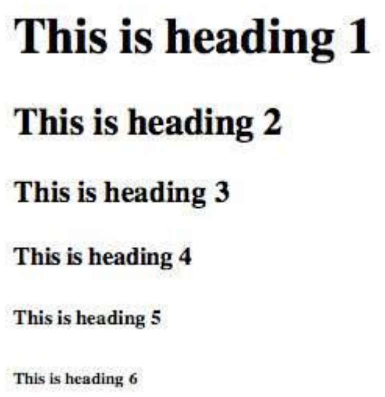
Heading Tags

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements **<h1>, <h2>, <h3>, <h4>, <h5>, and <h6>**. While displaying any heading, browser adds one line before and one line after that heading.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Heading Example</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

This will produce following result:



This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Formatting Tags:

Bold Text

Anything that appears within **<b>...</b>** element, is displayed in bold as shown below:

Example:

```
<!DOCTYPE html>

<html>

<head>

<title>Bold Text Example</title>

</head>

<body>

<p>The following word uses a <b>bold</b> typeface.</p>
```

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

This will produce following result:

The following word uses a **bold** typeface.

Italic Text:

Anything that appears within **<i>...</i>** element is displayed in italicized.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Italic Text Example</title>
</head>
<body>
<p>The following word uses a <i>italicized</i> typeface.</p>
</body>
</html>
```

This will produce following result:

The following word uses a *italicized* typeface.

Underlined Text:

Anything that appears within **<u>...</u>** element, is displayed with underline as shown below:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Underlined Text Example</title>
</head>
<body>
<p>The following word uses a <u>underlined</u> typeface.</p>
</body>
</html>
```

This will produce following result:

The following word uses a underlined typeface.

HTML Paragraphs

The HTML **<p>** element defines a **paragraph**:

Example

```
<!DOCTYPE html>

<html>

<body>

<p>This is a paragraph.</p>

<p>This is a paragraph.</p>

<p>This is a paragraph.</p>

</body>

</html>
```

This will produce following result:

This is a paragraph.

This is a paragraph.

This is a paragraph.

TT Tag:

The <tt> tag is not supported in HTML5.

If <tt> was used for marking up keyboard input, consider the <kbd> element; for variables, consider the <var> element; for computer code, consider the <code> element; and for computer output, consider the <samp> element, or use CSS instead.

The <tt> tag defines teletype text.

```
<!DOCTYPE html>

<html>
```

```
<body>

<p>This text is normal.</p>

<p><tt>Thistext is teletype text.</tt></p>

</body>

</html>
```

Output ;

This text is normal.

This text is teletype text.

Strike Text

Anything that appears within **<strike>...</strike>** element is displayed with strikethrough, which is a thin line through the text as shown below:

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Strike Text Example</title>

</head>

<body>

<p>The following word uses a <strike>strikethrough</strike> typeface.</p>

</body>

</html>
```

This will produce following result:

The following word uses a ~~strikethrough~~ typeface.

Monospaced Font:

The content of a **<tt>...</tt>** element is written in monospaced font. Most of the fonts are known as variable-width fonts because different letters are of different widths (for example, the letter 'm' is wider than the letter 'i'). In a monospaced font, however, each letter has the same width.

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Monospaced Font Example</title>

</head>

<body>

<p>The following word uses a <tt>monospaced</tt> typeface.</p>

</body>

</html>
```

This will produce following result:

The following word uses a monospaced typeface.

BR TAG:

The <br> tag inserts a single line break.

The <br> tag is an empty tag which means that it has no end tag.

```
<!DOCTYPE html>

<html>

<body>

<p>To break lines<br>in a text,<br>use the br element.</p>

</body></html>
```

This will produce following result:

To break lines
in a text,
use the br element.

EM Tag

The <em> tag is a phrase tag. It renders as emphasized text.

Tip: This tag is not deprecated, but it is possible to achieve richer effect with CSS.

```
<html>
<head>
<style>
em {
font-style: italic;
}</style>
</head>
<body>
<p>An em element is displayed like this:</p>
<em>Some emphasized text</em>
<p>Change the default CSS settings to see the effect.</p>
</body>
</html>
```

This will produce following result:

An em element is displayed like this:

Some emphasized text

Change the default CSS settings to see the effect.

Example

```
<!DOCTYPE html>
<html>
<body>
<em>Emphasized text</em><br>
<strong>Strong text</strong><br>
<code>A piece of computer code</code><br>
<samp>Sample output from a computer program</samp><br>
<kbd>Keyboard input</kbd><br>
<var>Variable</var>
</body>  </html>
```

This will produce following result:

Emphasized text
Strong text
A piece of computer code
Sample output from a computer program
Keyboard input
Variable

Grouping Content

The **<div>** and **<span>** elements allow you to group together several elements to create sections or subsections of a page.

For example, you might want to put all of the footnotes on a page within a <div> element to indicate that all of the elements within that <div> element relate to the footnotes. You might then attach a style to this <div> element so that they appear using a special set of style rules.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Div Tag Example</title>
</head>
<body>
<divid="menu"align="middle">
<ahref="/index.htm">HOME</a> |
<ahref="/about/contact_us.htm">CONTACT</a> |
<ahref="/about/index.htm">ABOUT</a>
</div>
<divid="content"align="left"bgcolor="white">
<h5>Content Articles</h5>
<p>Actual content goes here.....</p>
</div>
</body>
</html>
```

This will produce following result:

HOME | CONTACT | ABOUT

1.1.1.12.1 CONTENT ARTICLES

Actual content goes here.....

The <span> element, on the other hand, can be used to group inline elements only. So, if you have a part of a sentence or paragraph which you want to group together, you could use the <span> element as follows

Example

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Span Tag Example</title>
</head>
<body>
<p>This is the example of <spanstyle="color:yellow">span tag</span> and the
<spanstyle="color:red">div tag</span>alongwith CSS</p>
</body>
</html>
```

This will produce following result:

This is the example of span tag and the div tag along with CSS

These tags are commonly used with CSS to allow you to attach a style to a section of a page.

HR TAG

The <hr> tag defines a thematic break in an HTML page (e.g. a shift of topic).

The <hr> element is used to separate content (or define a change) in an HTML page.

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML</h1>
<p>HTML is a language for describing web pages.</p>
<hr>
<h1>CSS</h1>
<p>CSS defines how to display HTML elements.</p>
</body>
</html>
```

This will produce following result:

HTML

HTML is a language for describing web pages.

CSS

CSS defines how to display HTML elements.

PRE TAG

The <pre> tag defines preformatted text.

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

```
<!DOCTYPE html>
<html>
<body>
<pre>
Text in a pre element
is displayed in a fixed-width
font, and it preserves
bothspaces and
line breaks
</pre>
</body>
</html>
```

This will produce following result:

Text in a pre element is displayed in a fixed-width font, and it preserves both spaces and line breaks

Fonts Tag

Fonts play very important role in making a website more user friendly and increasing content readability. Font face and color depends entirely on the computer and browser that is being used to view your page but you can use HTML **<font>** tag to add style, size, and color to the text on your website. You can use a **<basefont>** tag to set all of your text to the same size, face, and color.

The font tag is having three attributes called **size, color**, and **face** to customize your fonts. To change any of the font attributes at any time within your webpage, simply use the <font> tag. The text that follows will remain changed until you close with the </font> tag. You can change one or all of the font attributes within one <font> tag.

Note: The font and base font tags are deprecated and it is supposed to be removed in a future version of HTML. So they should not be used rather, it's suggested to use CSS styles to manipulate your fonts. But still for learning purpose, this chapter will explain font and base font tags in detail.

Set Font Size

You can set content font size using **size** attribute. The range of accepted values is from 1(smallest) to 7(largest). The default size of a font is 3.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Font Size</title>
</head>
<body>
<font size="1">Font size="1"</font><br/>
<font size="2">Font size="2"</font><br/>
<font size="3">Font size="3"</font><br/>
<font size="4">Font size="4"</font><br/>
<font size="5">Font size="5"</font><br/>
<font size="6">Font size="6"</font><br/>
<font size="7">Font size="7"</font>
</body>
</html>
```

This will produce following result:

```
Font size="1"
Font size="2"
Font size="3"
Font size="4"
Font size="5"
Font size="6"
Font size="7"
```

Relative Font Size

You can specify how many sizes larger or how many sizes smaller than the preset font size should be. You can specify it like **<font size="+n"> or <font size="-n">**

Example:

```
<!DOCTYPE html>
```

```
<html>

<head>

<title>Relative Font Size</title>

</head>

<body>

<font size="-1">Font size="-1"</font><br/>

<font size="+1">Font size="+1"</font><br/>

<font size="+2">Font size="+2"</font><br/>

<font size="+3">Font size="+3"</font><br/>

<font size="+4">Font size="+4"</font>

</body>

</html>
```

This will produce following result:

```
Font size="-1"
Font size="+1"
Font size="+2"
Font size="+3"
Font size="+4"
```

Setting Font Face

You can set font face using *face* attribute but be aware that if the user viewing the page doesn't have the font installed, they will not be able to see it. Instead user will see the default font face applicable to the user's computer.

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Font Face</title>

</head>

<body>

<font face="Times New Roman"size="5">Times New Roman</font><br/>
```

```
<font face="Verdana"size="5">Verdana</font><br/>
<font face="Comic sans MS"size="5">Comic Sans MS</font><br/>
<font face="WildWest"size="5">WildWest</font><br/>
<font face="Bedrock"size="5">Bedrock</font><br/>
</body>
</html>
```

This will produce following result:

```
Times New Roman
Verdana
Comic Sans MS
WildWest
Bedrock
```

Specify alternate font faces

A visitor will only be able to see your font if they have that font installed on their computer. So, it is possible to specify two or more font face alternatives by listing the font face names, separated by a comma.

```
<fontface="arial,helvetica">
<fontface="Lucida Calligraphy,Comic Sans MS,Lucida Console">
```

When your page is loaded, their browser will display the first font face available. If none of the given fonts are installed, then it will display the default font face *Times New Roman*.

Setting Font Color

We can set any font color you like using *color* attribute. You can specify the color that you want by either the color name or hexadecimal code for that color.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Font Color</title>
</head>
<body>
<font color="#FF00FF">This text is in pink</font><br/>
<font color="red">This text is red</font>
```

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

This will produce following result:

This text is in pink
This text is red

The <basefont> Element:

The <basefont> element is supposed to set a default font size, color, and typeface for any parts of the document that are not otherwise contained within a <font> tag. You can use the <font> elements to override the <basefont> settings.

The <basefont> tag also takes color, size and face attributes and it will support relative font setting by giving size a value of +1 for a size larger or -2 for two sizes smaller.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Basefont Color</title>
</head>
<body>
<basefontface="arial, verdana, sans-serif"size="2"color="#ff0000">
<p>This is the page's default font.</p>
<h2>Example of the &lt;basefont&gt; Element</h2>
<p><fontsize="+2"color="darkgray">
This is darkgray text with two sizes larger
</font></p>
<p><fontface="courier"size="-1"color="#000000">
It is a courier font, a size smaller and black in color.
</font></p>
</body>
</html>
```

This will produce following result:

This is the page's default font.

Example of the <basefont> Element

This is darkgray text with two sizes larger

It is a courier font, a size smaller and black in color.

Images are very important to beautify as well as to depict many complex concepts in simple way on your web page. This tutorial will take you through simple steps to use images in your web pages.

Insert Image

We can insert any image in your web page by using **<img>** tag. Following is the simple syntax to use this tag.

```
<imgsrc="Image URL" ... attributes-list/>
```

The <img> tag is an empty tag, which means that it can contain only list of attributes and it has no closing tag.

Example

To try following example, let's keep our HTML file test.htm and image file test.png in the same directory:

```
<!DOCTYPE html>
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>
<imgsrc="/html/images/test.png"alt="Test Image"/>
</body>
</html>
```

This will produce following result:

Simple Image Insert

We can use PNG, JPEG or GIF image file based on your comfort but make sure you specify correct image file name in **src** attribute. Image name is always case sensitive.

The **alt** attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.

Set Image Location

Usually we keep our all the images in a separate directory. So let's keep HTML file test.htm in our home directory and create a subdirectory **images** inside the home directory where we will keep our image test.png.

Example

Assuming our image location is "/html/image/test.png", try the following example:

```
<!DOCTYPE html>
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>
<imgsrc="/html/images/test.png"alt="Test Image"/>
</body>
</html>
```

This will produce following result:

Simple Image Insert

Set Image Width/Height

We can set image width and height based on your requirement using **width** and **height** attributes. You can specify width and height of the image in terms of either pixels or percentage of its actual size.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Width and Height
</title>
</head>
<body>
<p>Setting image width and height</p>
<imgsrc="/html/images/test.png"alt="Test Image"width="150"height="100"/>
</body>
</html>
```

This will produce following result:

Setting image width and height

Set Image Border

By default image will have a border around it, you can specify border thickness in terms of pixels using **border** attribute. A thickness of 0 means, no border around the picture.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Border</title>
</head>
<body>
<p>Setting image Border</p>
<imgsrc="/html/images/test.png"alt="Test Image"border="3"/>
</body>
```

```
</html>
```

This will produce following result:

Setting image Border

Set Image Alignment

By default image will align at the left side of the page, but you can use **align** attribute to set it in the center or right.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Alignment</title>
</head>
<body>
<p>Setting image Alignment</p>
<imgsrc="/html/images/test.png"alt="Test Image"border="3"align="right"/>
</body>
</html>
```

This will produce following result:

Setting image Alignment

Some characters are reserved in HTML and they have special meaning when used in HTML document. For example, you cannot use the greater than and less than signs or angle brackets within your HTML text because the browser will treat them differently and will try to draw a meaning related to HTML tag.

HTML processors must support following five special characters listed in the table that follows.

Symbol	Description	Entity Name	Number Code
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos;	&#39;
&	Ampersand	&amp;	&#38;
<	less-than	&lt;	&#60;
>	greater-than	&gt;	&#62;

Example

If you want to write <div id="character"> as a code then you will have to write as follows:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Entities</title>
</head>
<body>
&lt;div id=&quot;character&quot;&gt;
</body>
</html>
```

This will produce following result:

```
<div id="character">
```

HTML META TAG

HTML lets you specify metadata - additional important information about a document in a variety of ways. The META elements can be used to include name/value pairs describing properties of the HTML document, such as author, expiry date, a list of keywords, document author etc.

The **<meta>** tag is used to provide such additional information. This tag is an empty element and so does not have a closing tag but it carries information within its attributes.

We can include one or more meta tags in your document based on what information you want to keep in your document but in general, meta tags do not impact physical appearance of the document so from appearance point of view, it does not matter if you include them or not.

Adding Meta Tags to Your Documents

We can add metadata to your web pages by placing <meta> tags inside the header of the document which is represented by **<head>** and **</head>** tags. A meta tag can have following attributes in addition to core attributes:

Attribute	**Description**
Name	Name for the property. Can be anything. Examples include, keywords, description, author, revised, generator etc.
content	Specifies the property's value.
scheme	Specifies a scheme to interpret the property's value (as declared in the content attribute).
http-equiv	Used for http response message headers. For example http-equiv can be used to refresh the page or to set a cookie. Values include content-type, expires, refresh and set-cookie.

Specifying Keywords

We can use <meta> tag to specify important keywords related to the document and later these keywords are used by the search engines while indexing your webpage for searching purpose.

Example

Following is an example where we are adding HTML, Meta Tags, Metadata as important keywords about the document.

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Meta Tags Example</title>
<metaname="keywords"content="HTML, Meta Tags, Metadata"/>
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

This will produce following result:

Hello HTML5!

Document Description

We can use <meta> tag to give a short description about the document. This again can be used by various search engines while indexing your webpage for searching purpose.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<metaname="keywords" content="HTML, Meta Tags, Metadata"/>
<metaname="description" content="Learning about Meta Tags."/>
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Document Revision Date

We can use <meta> tag to give information about when last time the document was updated. This information can be used by various web browsers while refreshing your webpage.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
```

```
<metaname="keywords"content="HTML, Meta Tags, Metadata"/>
<metaname="description"content="Learning about Meta Tags."/>
<metaname="revised"content="Tutorialspoint, 3/7/2014"/>
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Document Refreshing

A <meta> tag can be used to specify a duration after which your web page will keep refreshing automatically.

Example

If you want your page keep refreshing after every 5 seconds then use the following syntax.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<metaname="keywords"content="HTML, Meta Tags, Metadata"/>
<metaname="description"content="Learning about Meta Tags."/>
<metaname="revised"content="Tutorialspoint, 3/7/2014"/>
<metahttp-equiv="refresh"content="5"/>
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

Page Redirection

We can use <meta> tag to redirect your page to any other webpage. You can also specify a duration if you want to redirect the page after a certain number of seconds.

Example

Following is an example of redirecting current page to another page after 5 seconds. If you want to redirect page immediately then do not specify *content*attribute.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<metaname="keywords"content="HTML, Meta Tags, Metadata"/>
<metaname="description"content="Learning about Meta Tags."/>
<metaname="revised"content="Tutorialspoint, 3/7/2014"/>
<metahttp-equiv="refresh"content="5; url=https://www.tutorialspoint.com"/>
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

UNIT 2

HTML 5 AND CSS3(CASCADE STYLE SHEET)

2.1 HTML5 and Advanced HTML :

2.1.1 HTML5 :

What is HTML5?

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.

It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc. HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included. To natively include and handle multimedia and graphical content, the new <video>, <audio> and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and <menu> have been changed, redefined or standardized.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for any invalid documents.

The default character encoding in HTML5 is UTF-8
The DOCTYPE declaration for HTML5 is very simple

```
<!DOCTYPE html>
```

The character encoding (charset) declaration is also very simple

```
<meta charset="UTF-8">
```

HTML5 Example

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>
<body>
Content of the document......
</body>
</html>
```

Difference between HTML and HTML5

Html	Html5
Doctype declaration in Html is too longer <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">	DOCTYPE declaration in Html5 is very simple "<!DOCTYPE html>
character encoding in Html is also longer <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">	character encoding (charset) declaration is also very simple <meta charset="UTF-8">
Audio and Video are not part of HTML4	Audio and Videos are integral part of HTML5 e.g. <audio> and <video> tags.
Vector Graphics is possible with the help of technologies such as VML, Silverlight, Flash etc	Vector graphics is integral part of HTML5 e.g. SVG and canvas
It is almost impossible to get true GeoLocation of user browsing any website especially if it comes to mobile devices.	JS GeoLocation API in HTML5 helps identify location of user browsing any website (provided user allows it)
Html5 use cookies.	It provides local storage in place of cookies.
Not possible to draw shapes like circle, rectangle, triangle.	Using Html5 you can draw shapes like circle, rectangle, triangle.
Does not allow JavaScript to run in browser. JS runs in same thread as browser interface.	Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5
Works with all old browsers	Supported by all new browser.

New elements in HTML5

The most interesting new HTML5 elements are:

- ✓ New **semantic elements** like <header>, <footer>, <article>, and <section>.
- ✓ New **attributes of form elements** like number, date, time, calendar, and range.
- ✓ New **graphic elements**: <svg> and <canvas>.
- ✓ New **multimedia elements**: <audio> and <video>.

Canvas elements

The HTML <canvas> element is used to draw graphics on a web page. The HTML <canvas> element is used to draw graphics via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics. Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Media elements:

Multimedia on the web is sound, music, videos, movies, and animations. Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and formats.

Browser Support

The first web browsers had support for text only, limited to a single font in a single color Later came browsers with support for colors and fonts, and images. Audio, video, and animation have been handled differently by the major browsers. Different formats have been supported, and some formats require extra helper programs (plug-ins) to work.HTML5 multimedia promises an easier future for multimedia.

Multimedia Formats

Multimedia elements (like audio or video) are stored in media files. The most common way to discover the type of a file, is to look at the file extension.Multimedia files have formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Semantic and structural element

Semantics is the study of the meanings of words and phrases in a language.

Semantic elements = elements with a meaning.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

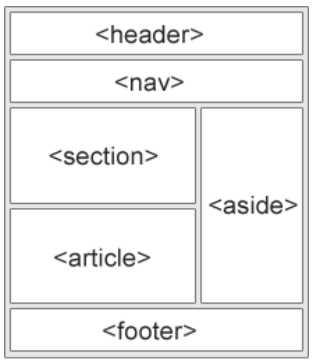
HTML5 semantic elements are supported in all modern browsers.In addition, you can "teach" older browsers how to handle "unknown elements".

New Semantic Elements in HTML5

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- ✓ <article>
- ✓ <aside>
- ✓ <details>
- ✓ <figcaption>
- ✓ <figure>
- ✓ <footer>
- ✓ <header>
- ✓ <main>
- ✓ <mark>
- ✓ <nav>
- ✓ <section>
- ✓ <summary>
- ✓ <time>



HTML5 <section> Element

The <section> element defines a section in a document. A section is a thematic grouping of content, typically with a heading.

A home page could normally be split into sections for introduction, content, and contact information.

Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

HTML5 <article> Element

The <article> element specifies independent, self-contained content.An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Nesting <article> in <section> or Vice Versa?

The <article> element specifies independent, self-contained content.

The <section> element defines section in a document.

On the Internet, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <sections> elements.You will also find pages with <section> elements containing <section> elements, and <article> elements containing <article> elements.

Example for a newspaper: The sport **articles** in the sport **section**, may have a technical **section** in each **article**.

HTML5 <header> Element

The <header> element specifies a header for a document or section.The <header> element should be used as a container for introductory content. We can have several <header> elements in one document. The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <footer> Element

The <footer> element specifies a footer for a document or section. A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc. You may have several <footer> elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

The aside content should be related to the surrounding content.

Example

```
<p>My family and I visited The Epcot center this summer.</p>
<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a <**figure**> element:

Example

```
<figure>
  <img src="pic_mountain.jpg" alt="The Pulpit Rock" width="304" height="228">
  <figcaption>Fig1.-The Pulpit Rock, Norway.</figcaption>
</figure>
```

The <**img**> element defines the image, the <**figcaption**> element defines the caption.

Why Semantic Elements?

With HTML4, developers used their own id/class names to style elements: header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, etc.

This made it impossible for search engines to identify the correct web page content. With the new HTML5 elements (<header> <footer> <nav> <section> <article>), this will become easier.A Semantic Web allows data to be shared and reused across applications, enterprises, and communities.

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

New graphic elements : <canvas> and <svg>

<canvas> element

The HTML <canvas> element is used to draw graphics via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics. Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content. The markup looks like the one given below

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute. Here is an example of a basic, empty canvas:

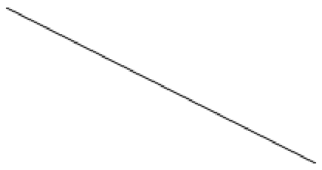
Example

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Draw a Line

Example

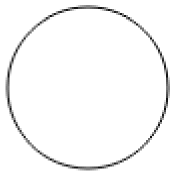
```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```



Draw a Circle:

Example

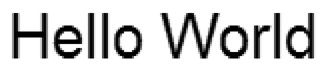
```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```



Draw a Text :

Example

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
```



HTML Canvas Can Draw Text

Canvas can draw colorful text, with or without animation.

HTML Canvas Can Draw Graphics

Canvas has great features for graphical data presentation with an imagery of graphs and charts.

HTML Canvas Can be Animated

Canvas objects can move. Everything is possible: from simple bouncing balls to complex animations.

HTML Canvas Can be Interactive

Canvas can respond to JavaScript events. Canvas can respond to any user action (key clicks, mouse clicks, button clicks, finger movement).

HTML Canvas Can be Used in Games

Canvas' methods for animations, offer a lot of possibilities for HTML gaming applications.

Canvas Example

In HTML, a <canvas> element looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

The <canvas> element must have an id attribute so it can be referred to by JavaScript.The width and height attribute is necessary to define the size of the canvas.

Tip: You can have multiple <canvas> elements on one HTML page.

By default, the <canvas> element has no border and no content.To add a border, use a style attribute:

Example

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
</canvas>
```

<svg> element

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web

<svg> tag

Html5 introduce new tag **<svg>**, SVG stands for **Scalable Vector Graphics**. It is used to define graphics for the Web. <svg> tag is container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images. It is mostly used for vector type diagrams like pie charts, 2-Dimensional graphs in an X,Y coordinate system etc.

SVG Circle

Example

```
<!DOCTYPE html>
<html>
<body>
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>
</body>
</html>
```

Differences Between SVG and Canvas

SVG is a language for describing 2D graphics in XML. Canvas draws 2D graphics, on the fly (with a JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. We can attach JavaScript event handlers for an element.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

Canvas	SVG
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

2.1.2 Advanced HTML

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.

Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

Links

HTML links are hyperlinks. Just click the link and jump to another document. When the mouse is moved over a link, the mouse arrow will turn into a little hand. A link does not have to be text. It can be an image or any other HTML element.

HTML Links - Syntax

In HTML, links are defined with the **<a>** tag:

<a href="*url*">*link text*</a>

Example

<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>

The **href** attribute specifies the destination address (https://www.w3schools.com/html/) of the link.The **link text** is the visible part (Visit our HTML tutorial).Clicking on the link text will send you to the specified address.

Local Links

The example above used an absolute URL (A full web address). A local link (link to the same web site) is specified with a relative URL (without http://www....).

Example

```
<a href="html_images.asp">HTML Images</a>
```

HTML Link Colors

By default, a link will appear like this (in all browsers):

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

The default colors can be changed by using styles.

Example

```
<style>
a:link {
   color: green;
   background-color: transparent;
   text-decoration: none;
}
a:visited {
   color: pink;
   background-color: transparent;
   text-decoration: none;
}
a:hover {
   color: red;
   background-color: transparent;
   text-decoration: underline;
}
a:active {
   color: yellow;
   background-color: transparent;
   text-decoration: underline;
}
</style>
```

HTML Links - The target Attribute

The **target** attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- _blank - Opens the linked document in a new window or tab
- _self - Opens the linked document in the same window/tab as it was clicked (this is default)
- _parent - Opens the linked document in the parent frame
- _top - Opens the linked document in the full body of the window
- framename - Opens the linked document in a named frame

This example will open the linked document in a new browser window/tab:

Example

```
<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

If the webpage is locked in a frame, use target="_top" to break out of the frame:

Example

```
<a href="https://www.w3schools.com/html/" target="_top">HTML5 tutorial!</a>
```

HTML Links - Image as Link

It is common to use images as links:

Example

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0;">
</a>
```

HTML Links - Create a Bookmark

HTML bookmarks are used to allow readers to jump to specific parts of a Web page.Bookmarks can be useful if webpage is very long. To make a bookmark, first create the bookmark, and then add a link to it.When the link is clicked, the page will scroll to the location with the bookmark.

Example

First, create a bookmark with the id attribute:

```
<h2 id="C4">Chapter 4</h2>
```

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

```
<a href="#C4">Jump to Chapter 4</a>
```

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:

Example

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

External Paths

External pages can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a web page:

Example

```
<a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>
```

This example links to a page located in the html folder on the current web site:

Example

```
<a href="/html/default.asp">HTML tutorial</a>
```

This example links to a page located in the same folder as the current page:

Example

```
<a href="default.asp">HTML tutorial</a>
```

Summary

- Use the **<a>** element to define a link
- Use the **href** attribute to define the link address
- Use the **target** attribute to define where to open the linked document
- Use the **<img>** element (inside <a>) to use an image as a link
- Use the **id** attribute (id="*value*") to define bookmarks in a page
- Use the **href** attribute (href="#*value*") to link to the bookmark

Lists

HTML offers web authors three ways for specifying lists of information. All lists must contain one or more list elements. Lists may contain:

- **<ul>** - An unordered list. This will list items using plain bullets.
- **<ol>** - An ordered list. This will use different schemes of numbers to list your items.

- **<dl>** - A definition list. This arranges your items in the same way as they are arranged in a dictionary.

HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

This will produce following result:

- Beetroot
- Ginger
- Potato
- Radish

The type Attribute

We can use type attribute for <ul> tag to specify the type of bullet you like. By default it is a disc. Following are the possible options:

```
<ul type="square">
<ul type="disc">
<ul type="circle">
```

Example

Following is an example where we used <ul type="square">

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul type="square">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

This will produce following result:

- Beetroot
- Ginger
- Potato
- Radish

Example

Following is an example where we used <ul type="disc"> :

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul type="disc">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
```

</ul>

</body>

</html>

This will produce following result:

- Beetroot
- Ginger
- Potato
- Radish

Example

Following is an example where we used <ul type="circle"> :

<!DOCTYPE html>

<html>

<head>

<title>HTML Unordered List</title>

</head>

<body>

<ul type="circle">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ul>

</body>

</html>

This will produce following result:

o Beetroot oGinger oPotato
o Radish

HTML Ordered Lists

If you are required to put your items in a numbered list instead of bulleted then HTML ordered list will be used. This list is created by using <ol> tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

This will produce following result:

1. Beetroot
2. Ginger
3. Potato
4. Radish

The type Attribute

We can use type attribute for <ol> tag to specify the type of numbering you like. By default it is a number. Following are the possible options:

<ol type="1"> - Default-Case Numerals.

<ol type="I"> - Upper-Case Numerals.

<ol type="i"> - Lower-Case Numerals.

<ol type="a"> - Lower-Case Letters.

<ol type="A"> - Upper-Case Letters.

Example

Following is an example where we used <ol type="1">

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="1">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

This will produce following result:

1. Beetroot
2. Ginger
3. Potato
4. Radish

Example

Following is an example where we used <ol type="I">

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="I">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
```

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

This will produce following result:

I. Beetroot
II. Ginger
III. Potato
IV. Radish

Example

Following is an example where we used <ol type="i">

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="i">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

This will produce following result:

i. Beetroot
ii. Ginger
iii. Potato
iv. Radish

Example

Following is an example where we used <ol type="A">

```
<!DOCTYPE html>
<html>
<head>
```

<title>HTML Ordered List</title>

</head>

<body>

<ol type="A">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce following result:

A. Beetroot
B. Ginger
C. Potato
D. Radish

Example

Following is an example where we used <ol type="a">

<!DOCTYPE html>

<html>

<head>

<title>HTML Ordered List</title>

</head>

<body>

<ol type="a">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce following result:

a. Beetroot
b. Ginger
c. Potato
d. Radish

The start Attribute

We can use start attribute for <ol> tag to specify the starting point of numbering you need. Following are the possible options:

<ol type="1" start="4"> - Numerals starts with 4.

<ol type="I" start="4"> - Numerals starts with IV.

<ol type="i" start="4"> - Numerals starts with iv.

<ol type="a" start="4"> - Letters starts with d.

<ol type="A" start="4"> - Letters starts with D.

Example

Following is an example where we used <ol type="i" start="4" ><!DOCTYPE html>

```
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="i" start="4">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

This will produce following result:

iv. Beetroot
v. Ginger
vi. Potato
vii. Radish

HTML Definition Lists

HTML and XHTML support a list style which is called definition lists where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list. Definition List makes use of following three tags.

- **<dl> - Defines the start of the list**
- **<dt> - A term**
- **<dd> - Term definition**
- **</dl> - Defines the end of the list**

Example

```
<!DOCTYPE html>

<html>

<head>

<title>HTML Definition List</title>

</head>

<body>

<dl>

<dt><b>HTML</b></dt>

<dd>This stands for Hyper Text Markup Language</dd>

<dt><b>HTTP</b></dt>

<dd>This stands for Hyper Text Transfer Protocol</dd>

</dl>

</body>

</html>
```

This will produce following result:

HTML

This stands for Hyper Text Markup Language

HTTP

This stands for Hyper Text Transfer Protocol

HTML tables

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

tr and td Tags

The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells.

Here **border** is an attribute of <table> tag and it is used to put a border across all the cells. If you do not need a border then you can use border="0". **Table Heading**

Table heading can be defined using **<th>** tag. This tag will be put to replace <td> tag, which is used to represent actual data cell. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element in any row.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Header</title>
</head>
<body>
<table border="1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

This will produce following result:

Name	**Salary**
Ramesh Raman	5000
Shabbir Hussein	7000

Cellpadding and Cellspacing Attributes

There are two attribiutes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cells. The cellspacing attribute defines the width of the border, while cellpadding represents the distance between cell borders and the content within a cell.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Cellpadding</title>
</head>
<body>
<table border="1" cellpadding="5" cellspacing="5">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

This will produce following result:

Name	**Salary**
Ramesh Raman	5000
Shabbir Hussein	7000

Colspan and Rowspan Attributes

You will use **colspan** attribute if you want to merge two or more columns into a single column. Similar way you will use **rowspan** if you want to merge two or more rows.

Example

```
<html>
<head>
<title>HTML Table Colspan/Rowspan</title>
</head>
<body>
<table border="1">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell
2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
</body>
</html>
```

This will produce following result:

Column 1	**Column 2**	**Column 3**
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Tables Backgrounds

We can set table background using one of the following two ways:

- **bgcolor** attribute - You can set background color for whole table or just for one cell.
- **background** attribute - You can set background image for whole table or just for one cell.

We can also set border color also using **bordercolor** attribute.

Example

```
<html>
<head>
<title>HTML Table Background</title>
</head>
<body>
<table border="1" bordercolor="green" bgcolor="yellow"><tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th></tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1  Cell
2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
</body>
</html>
```

This will produce following result:

Column 1	**Column 2**	**Column 3**
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Table Height and Width

You can set a table width and height using **width** and **height** attributes. You can specify table width or height in terms of pixels or in terms of percentage of available screen area.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Width/Height</title>
</head>
<body>
<table border="1" width="400" height="150">
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

This will produce following result:

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Table Caption

The caption tag will serve as a title or explanation for the table and it shows up at the top of the table. This tag is deprecated in newer version of HTML/XHTML.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Caption</title>
</head>
<body>
<table border="1" width="100%">
<caption>This is the caption</caption>
<tr>
<td>row 1, column 1</td><td>row 1, columnn 2</td>
</tr>
<tr>
<td>row 2, column 1</td><td>row 2, columnn 2</td>
</tr></table></body>
</html>
```

This will produce following result:

This is the caption	
row 1, column 1	row 1, columnn 2
row 2, column 1	row 2, columnn 2

Table Header, Body, and Footer

Tables can be divided into three portions: a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content holder of the table.

The three elements for separating the head, body, and foot of a table are:

- <thead> - to create a separate table header.
- <tbody> - to indicate the main body of the table.
- <tfoot> - to create a separate table footer.

A table may contain several <tbody> elements to indicate different *pages* or groups of data. But it is notable that <thead> and <tfoot> tags should appear before <tbody>

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table</title>
</head>
<body>
<table border="1" width="100%">
<thead>
<tr>
<td colspan="4">This is the head of the table</td>
</tr>
</thead>
<tfoot>
<tr>
<td colspan="4">This is the foot of the table</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
```

```
<td>Cell 4</td>
</tr>
</tbody>
</table>
</body>
</html>
```

This will produce following result:

This is the head of the table			
This is the foot of the table			
Cell 1	Cell 2	Cell 3	Cell 4

HTML frames

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your web pages:

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers that do not support frame technology.

Creating Frames

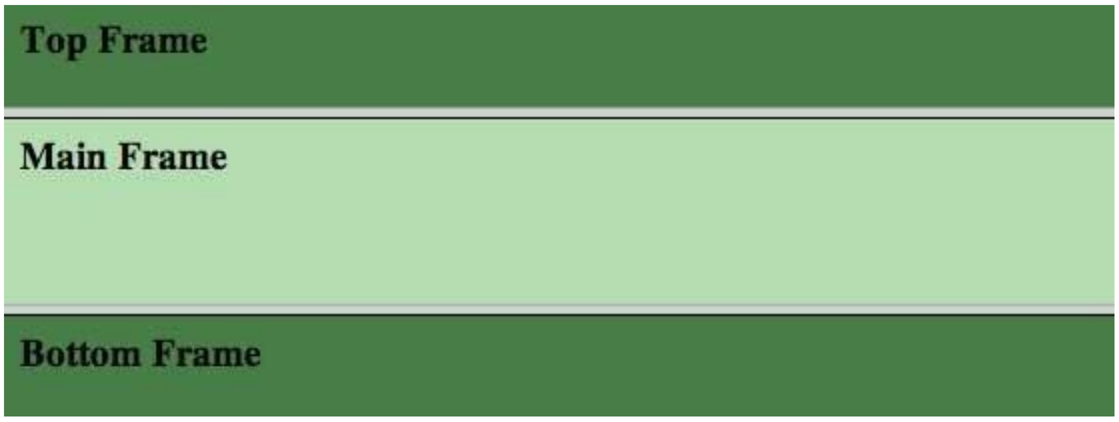
To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines how to divide the window into frames. The **rows** attribute of <frameset> tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

Example

Following is the example to create three horizontal frames:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset rows="10%,80%,10%">
<frame name="top" src="/html/top_frame.htm" />
<frame name="main" src="/html/main_frame.htm" />
<frame name="bottom" src="/html/bottom_frame.htm" />
<noframes>
<body>
    Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>
```

This will produce following result:

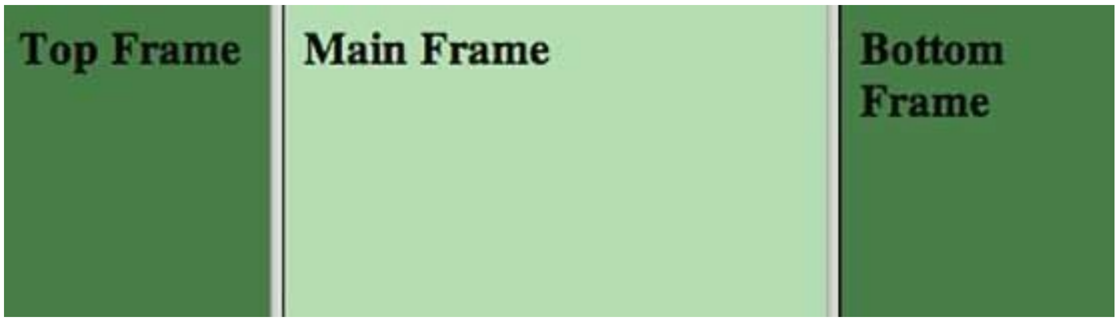


Example

Let's put above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Frames</title>
</head>
<frameset cols="25%,50%,25%">
<frame name="left" src="/html/top_frame.htm" />
<frame name="center" src="/html/main_frame.htm" />
<frame name="right" src="/html/bottom_frame.htm" />
<noframes>
<body>
    Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>
```

This will produce following result:



The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag:

Attribute	**Description**
cols	Specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of four ways: • Absolute values in pixels. For example to create three vertical frames, use *cols="100, 500,100"*. • A percentage of the browser window. For example to create three vertical frames, use *cols="10%, 80%,10%"*. • Using a wildcard symbol. For example to create three vertical frames, use *cols="10%,* • **,10%"*. In this case wildcard takes remainder of the window. • As relative widths of the browser window. For example to create three vertical frames, use *cols="3*,2*,1*"*. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
rows	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example to create two horizontal frames, use *rows="10%, 90%"*. You can specify the height of each row in the same way as explained above for columns.
border	This attribute specifies the width of the border of each frame in pixels. For example border="5". A value of zero means no border.
frameborder	This attribute specifies whether a threedimensional border should be displayed between frames. This attrubute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border.
framespacing	This attribute specifies the amount of space between frames in a frameset. This can take any integer value. For example framespacing="10" means there should be 10 pixels spacing between each frames.

The <frame> Tag Attributes

Following are important attributes of <frame> tag:

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src="/html/top_frame.htm" will load an HTML file available in html directory.
name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10".
noresize	By default you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm"

Browser Support for Frames

If a user is using any old browser or any browser which does not support frames then <noframes> element should be displayed to the user.

So you must place a <body> element inside the **<noframes>** element because the <frameset> element is supposed to replace the <body> element, but if a browser does not understand <frameset> element then it should understand what is inside the <body> element which is contained in a **<noframes>** element.

You can put some nice message for your user having old browsers. For example *Sorry!! your browser does not support frames.* as shown in the above example.

Frame's name and target attributes

One of the most popular uses of frames is to place navigation bars in one frame and then load main pages into a separate frame.

Let's see following example where a test.htm file has following code:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Target Frames</title>
</head>
<frameset cols="200, *">
<frame src="/html/menu.htm" name="menu_page" />
<frame src="/html/main.htm" name="main_page" />
<noframes>
<body>
    Your browser does not support frames.
</body>
</noframes>
</frameset>
</html>
```

Here we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by **menu.htm** file. The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.htm** file. For all the three links available in menu bar, we have mentioned target frame as **main_page**, so whenever you click any of the links in menu bar, available link will open in main_page.

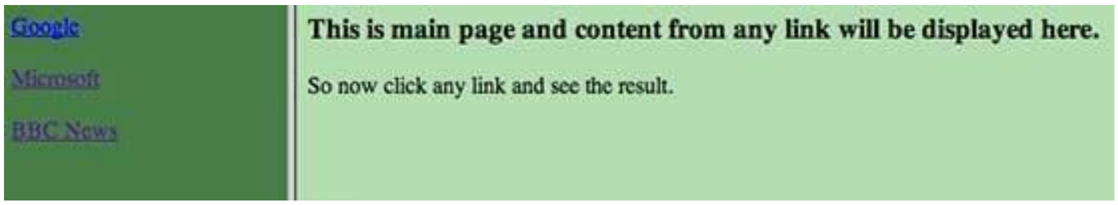
Following is the content of menu.htm file

```
<!DOCTYPE html>
<html>
<body bgcolor="#4a7d49">
<a href="https://www.google.com" target="main_page">Google</a><br /><br />
<a href="https://www.microsoft.com" target="main_page">Microsoft</a>
<br /><br />
<a href="https://news.bbc.co.uk" target="main_page">BBC
News</a>
</body>
</html>
```

Following is the content of main.htm file:

```
<!DOCTYPE html>
<html>
<body bgcolor="#b5dcb3">
<h3>This is main page and content from any link will be displayed here.</h3>
<p>So now click any link and see the result.</p>
</body>
</html>
```

When we load **test.htm** file, it produces following result:



This is main page and content from any link will be displayed here.

So now click any link and see the result.

Now you can try to click links available in the left panel and see the result. The *target* attribute can also take one of the following values:

Option	Description
_self	Loads the page into the current frame.
_blank	Loads a page into a new browser window.opening a new window.

_parent	Loads the page into the parent window, which in the case of a single frameset is the main browser window.
_top	Loads the page into the browser window, replacing any current frames.
target frame	Loads the page into a named targetframe.

HTML Forms

HTML Forms are required when you want to collect some data from the site visitor. For example during user registration you would like to collect information such as name, email address, credit card, etc. A form will take input from the site visitor and then will post it to a back-end application such as CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

There are various form elements available like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.

The HTML **<form>** tag is used to create an HTML form and it has following syntax:

<form action="Script URL" method="GET|POST"> form elements like input, textarea etc.
</form>

Form Attributes

Apart from common attributes, following is a list of the most frequently used form attributes:

Attribute	Description
action	Backend script ready to process your passed data.
method	Method to be used to upload data. The most frequently used are GET and POST methods.

target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
enctype	You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are: • **application/x-www-form-urlencoded** - This is the standard method most forms use in simple scenarios. • **mutlipart/form-data** - This is used when you want to upload binary data in the form of files like image, word file etc.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form:

- ✓ Text Input Controls
- ✓ Checkboxes Controls
- ✓ Radio Box Controls
- ✓ Select Box Controls
- ✓ File Select boxes
- ✓ Hidden Controls
- ✓ Clickable Buttons
- ✓ Submit and Reset Button

Text Input Controls

There are three types of text input used on forms:

- **Single-line text input controls -** This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.
- **Password input controls -** This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTMl <input> tag.
- **Multi-line text input controls -** This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

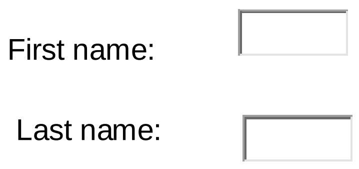
Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag. **Example**

Here is a basic example of a single-line text input used to take first name and last name:

```
<!DOCTYPE html>
<html>
<head>
<title>Text Input Control</title>
</head>
<body>
<form >
First name:  <input type="text" name="first_name" />
<br>
Last name:  <input type="text" name="last_name" />
</form>
</body>
</html>
```

This will produce following result:



```
First name:
Last name:
```

Attributes

Following is the list of attributes for <input> tag for creating text field.

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to **text**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.

value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

Password input controls

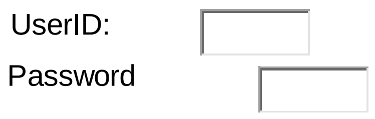
This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag but type attribute is set to **password**.

Example

Here is a basic example of a single-line password input used to take user password:

```
<!DOCTYPE html>
<html>
<head>
<title>Password Input Control</title>
</head>
<body>
<form >
User ID :  <input type="text" name="user_id" /><br>
Password:  <input type="password" name="password" />
</form>
</body>
</html>
```

This will produce following result:



UserID:
Password

Attributes

Following is the list of attributes for <input> tag for creating password field.

Attribute	Description
type	Indicates the type of input control and for password input control it will be set to **password**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text-input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text box.

Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

Example

Here is a basic example of a multi-line text input used to take item description:

```
<!DOCTYPE html>
<html>
<head>
<title>Multiple-Line Input Control</title>
</head>
<body>
<form>
Description : <br />
<textarea   rows="5"   cols="50"   name="description">   Enter   description
here...
```

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

This will produce following result:

Attributes

Following is the list of attributes for <textarea> tag.

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox**. Example

Here is an example HTML code for a form with two checkboxes: <!DOCTYPE html>

```
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
```

```
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
</form>
</body>
</html>
```

This will produce following result:

☐ ☐
Maths Physics

Attributes

Following is the list of attributes for <checkbox> tag.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to **checkbox**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the checkbox is selected.
checked	Set to *checked* if you want to select it by default.

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Example

Here is example HTML code for a form with two radio buttons:

```
<!DOCTYPE html>
<html>
<head>
<title>Radio Box Control</title>
</head>
<body>
```

```
<form>
<input type="radio" name="subject" value="maths"> Maths
<input type="radio" name="subject" value="physics"> Physics </form>
</body>
</html>
```

This will produce following result:

○ Maths ○ Physics

Attributes

Following is the list of attributes for radio button.

Attribute	Description
type	Indicates the type of input control and for checkbox input control it will be set to **radio**.
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	The value that will be used if the radio box is selected.
checked	Set to *checked* if you want to select it by default.

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

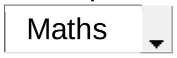
Example

Here is example HTML code for a form with one drop down box

```
<!DOCTYPE html>
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown">
```

```
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

This will produce following result:


Maths

Attributes

Following is the list of important attributes of <select> tag:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
size	This can be used to present a scrolling list box.
multiple	If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option> tag:

Attribute	Description
value	The value that will be used if an option in the select box is selected.
selected	Specifies that this option should be the initially selected value when the page loads.
label	An alternative way of labeling options

File Upload Box

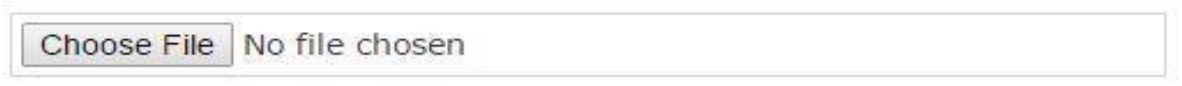
If a file is be uploaded to a web site ,use a file upload box which is also known as file select box. This is also created using the <input> element but type attribute is set to **file**.

Example

Here is example HTML code for a form with one file upload box:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="file" name="fileupload" accept="image/*" /></form>
</body>
</html>
```

This will produce following result:



Attributes

Following is the list of important attributes of file upload box:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
accept	Specifies the types of files that the server accepts.

Button Controls

There are various ways in HTML to create clickable buttons. It is also possible to create a clickable button using <input> tag by setting its type attribute to **button**. The type attribute can take the following values:

Type	Description
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their initial values.
button	This creates a button that is used to trigger a client-side script when the user clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

Example

Here is example HTML code for a form with three types of buttons:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
<input  type="button"  name="ok"  value="OK"    /><input  type="image"
name="imagebutton" src="/html/images/logo.png" />
</form>
</body>
</html>
```

This will produce following result:

Submit Reset

Hidden Form Controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page has be displayed next based on the passed current page.

Example

Here is example HTML code to show the usage of hidden control:

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<p>This is page 10</p>
<input type="hidden" name="pagename" value="10" />
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset"  value="Reset" />
</form>
</body>
</html>
```

This will produce following result: This is page 10

Submit Reset

COLGROUP

HTML <colgroup> tag is used for specifying properties for a group of columns within a table. Different properties can be applied to a column within a colgroup using the HTML col tag within the colgroup tag.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML colgroup Tag</title>
</head>
<body>
<p>This example shows a colgroup that has three columns of different widths:</p><table
border="1">
<colgroup span="3">
<col width="50"></col>
<col width="100"></col>
<col width="200"></col>
</colgroup>
<tr>
<td>col 1</td>
<td>col 2</td>
<td>col 3</td>
</tr>
</table>
</body>
</html>
```

This will produce following result:

This example shows a colgroup that has three columns of different widths:

col 1	col 2	col 3

HTML <tbody>,</thead>&<footer>

The HTML <tbody> tag is used in adding a body to a table. The tbody tag is used in conjunction with the thead tag and the tfoot tag in determining each part of the table (header, footer, body).

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML tbody Tag</title>
</head>
<body>
<table style="width:100%" border="1">
<thead>
<tr>
<td colspan="4">This is the head of the table</td>
</tr>
</thead><tfoot><tr>
<td colspan="4">This is the foot of the table</td></tr>
</tfoot>
<tbody>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
<td>Cell 4</td>
</tr>
<tr>
...more rows here containing four cells...
</tr></tbody>
<tbody>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
```

```
<td>Cell 3</td>
<td>Cell 4</td>
</tr>
<tr>
...more rows here containing four cells...
</tr>
</tbody>
</table>
</body>
</html>
```

This will produce following result:

...more rows here containing four cells... ...more rows here containing four cells...

This is the head of the table
This is the foot of the table

<iframe>

An inline frame can be defined with HTML tag **<iframe>**.The <iframe> tag is not somehow related to <frameset> tag, instead, it can appear anywhere in the document. The <iframe> tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders. The **src** attribute is used to specify the URL of the document that occupies the inline frame.

Example

Following is the example to show how to use the <iframe>:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Iframes</title>
</head>
<body>
<p>Document content goes here...</p>
<iframesrc="/html/menu.htm"width="555"height="200">     Sorry your browser does not support
inline frames.
```

```
</iframe>
<p>Document content also go here...</p>
</body>
</html>
```

This will produce following result:

Document content goes here...

Google

Microsoft

BBC News

Document content can also go here...

1.1.1.13 The <Iframe> Tag Attributes

Most of the attributes of the <iframe> tag, including name, class, frameborder, id, longdesc, marginheight, marginwidth, name, scrolling, style, and title behave exactly like the corresponding attributes for the <frame> tag.

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src="/html/top_frame.htm" will load an HTML file avalaible in html directory.
name	This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
marginwidth	This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
marginheight	This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight="10".

noresize	By default you can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling="no" means it should not have scroll bars.
longdesc	This attribute allows you to provide a link to another page containing a long description of the contents of the frame. For example longdesc="framedescription.htm"

2.2 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) describe how documents are presented on screens, in print, or perhaps how they are pronounced.

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, a number of style properties for a given HTML element can be specified. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

Example

First , consider an example of HTML document which makes use of <font> tag and associated attributes to specify text color and font size:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML CSS</title>
</head>
<body>
<p><font color="green" size="5">Hello, World!</font></p>
</body>
</html>
```

We can re-write above example with the help of Style Sheet as follows:

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>HTML CSS</title>
</head>
<body>
<p style="color:green;font-size:24px;">Hello, World!</p></body>
</html>
```

This will produce following result:

Hello, World!

CSS can be used in three ways in a HTML document:

- **External Style Sheet** - Define style sheet rules in a separate .css file and then include that file in the HTML document using HTML <link> tag.
- **Internal Style Sheet** - Define style sheet rules in header section of the HTML document using <style> tag.
- **Inline Style Sheet** - Define style sheet rules directly along-with the HTML elements using **style** attribute.

Let's see all the three cases one by one with the help of suitable examples.

External Style Sheet

If a style sheet is to be used to various pages, then it is always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as **.css** and it will be included in HTML files using <link> tag.

Example

Consider we define a style sheet file **style.css** which has following rules:

```
.red{    color: red; }
.thick{    font-size:20px; }
.green{ olor:green; }
```

Here we defined three CSS rules which will be applicable to three different classes defined for the HTML tags. Now let's make use of the above external CSS file in our following HTML document:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML External CSS</title>
```

```
<link rel="stylesheet" type="text/css" href="/html/style.css">
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body>
</html>
```

This will produce following result:

```
This is red
This is thick
This is green
This is thick and green
```

Internal Style Sheet

If a style sheet rules is to be applied to a single document , then include those rules in header section of the HTML document using <style> tag. Rules defined in internal style sheet overrides the rules defined in an external CSS file.

Example

Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using <style> tag:

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Internal CSS</title>
<style type="text/css">
.red{
color: red;
}
.thick{
  font-size:20px;
}
```

```
.green{
  color:green;
}
</style>
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body>
</html>
```

This will produce following result:

```
This is red
This is thick
This is green
This is thick and green
```

Inline Style Sheet

tyle sheet rules can be applied directly to any HTML element using **style** attribute of the relevant tag. This should be done only when a particular change in any HTML element only is required. Rules defined inline with the element overrides the rules defined in an external CSS file as well as the rules defined in <style> element.

Example

Let's re-write above example once again, but here we will write style sheet rules along with the HTML elements using **style** attribute of those elements.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Inline CSS</title>
</head>
<body>
<p style="color:red;">This is red</p>
<p style="font-size:20px;">This is thick</p>
<p style="color:green;">This is green</p>
```

```
<p style="color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

This will produce following result:

```
This is red
This is thick
This is green
This is thick and green
```

What is CSS3?

CSS3 is the latest evolution of the Cascading Style Sheets language and aims at extending CSS2.1. It brings a lot of long-awaited novelties, like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns, flexible box or grid layouts.The main difference between css2 and css3 is follows

- ✓ Media Queries
- ✓ Namespaces
- ✓ Selectors Level 3
- ✓ Color

CSS3 modules

CSS3 is collaboration of CSS2 specifications and new specifications.This collaboration is called as **module**. Some of the modules are shown below

- ✓ Selectors
- ✓ Box Model
- ✓ Backgrounds
- ✓ Image Values and Replaced Content
- ✓ Text Effects
- ✓ 2D Transformations
- ✓ 3D Transformations
- ✓ Animations
- ✓ Multiple Column Layout
- ✓ User Interface

CSS3 - Animation

Animation is process of making shape changes and creating motions with elements.

@keyframes

Keyframes will control the intermediate animation steps in CSS3.

Example of key frames with left animation

```
@keyframes animation {
  from {background-color: pink;}
  to {background-color: green;}
}
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: animation;
  animation-duration: 5s;
}
```

The above example shows height, width, color, name and duration of animation with keyframes syntax.

CSS3- Borders

Elements have a border that sits comfortably between the element's padding and margin. By default the border has a size of 0 — making it invisible but the thickness, style and color of the border can be set to make it appear.

Border shorthand

The **border** CSS property is a shorthand property for setting all individual border property values at once: border-width, border-style, and border-color. As with all shorthand properties, any individual value that is not specified is set to its corresponding initial value. Importantly, border cannot be used to specify a custom value for border-image, but instead sets it to its initial value, i.e., none.

Use border when you want to set all border properties simultaneously. While the border-width, border-style, and border-color shorthand properties accept up to four values, allowing to set different values for each edge, border only accepts a single value for each property. Thus, the same style is applied to all four edges.

Syntax

The border property is specified using one or more of the <br-width>, <br-style>, and <color> values listed below.

Values

<br-width>   Thickness of the border. Defaults to medium if absent.

<br-style>   Line style of the border. Defaults to none if absent.

<color> Color of the border. Defaults to the value of the element's color property.

Formal syntax

<br-width> || <br-style> || <color>

where

<br-width> = <length> | thin | medium | thick

<br-style> = none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

<color> = <rgb()> | <rgba()> | <hsl()> | <hsla()> | <hex-color> | <named-color> | currentcolor | <deprecated-system-color>

HTML

```
<p>You can edit the CSS below to experiment with border styles!</p>
<style content editable>
  .fun-border {
    border: 2px solid red;
  }
</style>
```

CSS

```
style {
  display: block;
  border: 1px dashed black;
}
```

Result

You can edit the CSS below to experiment with border styles!

fun-border { border: 2px solid red; }

CSS3 Backgrounds

The background of an element is the area that sits underneath an element's content, padding, and border. By default this is the case anyway — in newer browsers, the area that the background takes up can be altered using the background-clip property.

There are many other different properties used to manipulate the element's background.

- background-color: Sets a solid color for the background.
- background-image: Specifies a background image to appear in the background of the element. This can be a static file, or a generated gradient.
- background-position: Specifies the position that the background should appear inside the element background.
- background-repeat: Specifies whether the background should be repeated (tiled) or not.
- background-attachment: Specifies the behaviour of an element's background when its content scrolls, e.g. does it scroll with the content, or is it fixed?
- background: Shorthand for specifying the above five properties on one line.
- background-size: Allows a background image to be resized dynamically.

Background color

Use the background-color property. The default background color of most elements is not white but transparent .Therefore to set an element's background color to something interesting, it must be set explicitly.

In addition, it is important to set a background color as a fallback. Background colors are supported pretty much everywhere, whereas more exotic features such as background gradients are supported only in newer browsers, plus a background image might fail to load for some reason. It is therefore a good idea to set a basic background color as well as specifying such features, so the element's content is readable no matter what.

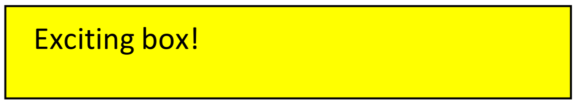
Example

```
<p>Exciting box!</p>
```

Let's give it a background color:

```
p {
  font-family: sans-serif;
  padding: 20px;
  /* background properties */
  background-color: yellow;
}
```

The result is as follows:



Exciting box!

Background image

The background-image property specifies a background image to display in the background of an element. The simplest usage of this property involves using the url() function — which takes the path to an image as a parameter — to fetch a static image file to insert.

Example

```
p {
  font-family: sans-serif;
  padding: 20px;
  /* background properties */
  background-color: yellow;
  background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-icon.png);
}
```

One important thing to bear in mind is that since background images are set using CSS and appear in the background of content, they will be invisible to assistive technologies like screen readers. They are not content images — they are just for decoration — if an image is to included in a web page that is part of the content, then it should be done with an <img> element.

Background repeat

Background-repeat allows to specify how the background image is repeated. The default value is repeat which makes the image keep repeating until the whole element background is filled.

Other common and widely supported values are

- no-repeat: The image will not repeat at all: it will only be shown once.
- repeat-x: The image will repeat horizontally all the way across the background.
- repeat-y: The image will repeat vertically all the way down the background.

Example

```
p {
  font-family: sans-serif;
  padding: 20px;
  /* background properties */
  background-color: yellow;
  background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-icon.png);
  background-repeat: no-repeat; }
```

Background position

Background-position allows us to position our background image wherever we want inside the background. Generally the property will take two values, separated by a space, which specify the horizontal (x) and vertical (y) coordinates of the image — the top left corner of the image, to be exact. Think of the background as a graph, with the x coordinate going across from left to right, and the y coordinate going from top to bottom.

The property can accept many different value types

- Absolute values like pixels — for example background-position: 200px 25px.
- Relative values like rems — for example background-position: 20rem 2.5rem.
- Percentages — for example background-position: 90% 25%.
- Keywords — for example background-position: right center. These two values are intuitive, and can take values of left, center, right, and top, center, bottom, respectively.

The values can be mixed and matched - for example background-position: 99% center. Also note that if one value is specified, that value will be assumed to be the horizontal value, and the vertical value will default to center.

Example:

```
p {
  font-family: sans-serif;
  padding: 20px;
  /* background properties */
  background-color: yellow;
  background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-icon.png);
  background-repeat: no-repeat;
  background-position: 99% center;
}
```

CSS3 - Web fonts

Web fonts are used to allows the fonts in CSS, which are not installed on local system.

Different web fonts formats

Fonts	Description
TrueType Fonts (TTF)	TrueType is an outline font standard developed by Apple and Microsoft in the late 1980s, It became most common fonts for both windows and MAC operating systems
OpenType Fonts (OTF)	OpenType is a format for scalable computer fonts and developed by Microsoft

The Web Open Font Format (WOFF)	WOFF is used for develop web page and developed in the year of 2009. Now it is using by W3C recommendation.
SVG Fonts/Shapes	SVG allow SVG fonts within SVG documentation. We can also apply CSS to SVG with font face property
Embedded OpenType Fonts (EOT)	EOT is used to develop the web pages and it has embedded in webpages so no need to allow 3rd party fonts

following code shows the sample code of font face

```
<html>
  <head>
    <style>
      @font-face {
        font-family: myFirstFont;
        src: url(/css/font/SansationLight.woff);
      }
      div {
        font-family: myFirstFont;
      }
    </Style>
  </head>
  <body>
    <div>This is the example of font face with CSS3.</div>
        <p><b>Original Text :</b>This is the example of font face with CSS3.</p>
  </body>
</html>
```

It will produce the following result –

This is the example of font face with CSS3.

Original Text : This is the example of font face with CSS3.

Fonts description

The following list contained all the fonts description which are placed in the @font-face rule −

Values	Description
font-family	Used to defines the name of font

src	Used to defines the URL
font-stretch	Used to find, how font should be stretched
font-style	Used to defines the fonts style
font-weight	Used to defines the font weight(boldness)

CSS3 - Multi columns

CSS3 supported multi columns to arrange the text as news paper structure.Some of most common used multi columns properties as shown below −

Values	Description
column-count	Used to count the number of columns that element should be divided
column-fill	Used to decide, how to fill the columns
column-gap	Used to decide the gap between the columns
column-rule	Used to specifies the number of rules
rule-color	Used to specifies the column rule color
rule-style	Used to specifies the style rule for column
rule-width	Used to specifies the width
column-span	Used to specifies the span between columns

Example

Below example shows the arrangement of text as news paper structure.

```
<html>
 <head>
 <style>
 .multi {
  /* Column count property */
  -webkit-column-count: 4;
-moz-column-count: 4;
column-count: 4;
 /* Column gap property */
 -webkit-column-gap: 40px;
-moz-column-gap: 40px;
 column-gap: 40px;
```

```
  /* Column style property */
-webkit-column-rule-style: solid;
 -moz-column-rule-style: solid;
 column-rule-style: solid;
 }
</style>
</head>
 <body>
 <div class="multi">
```

Tutorials Point originated from the idea that there exists a class of readers who respond better to online content and prefer to learn new skills at their own pace from the comforts of their drawing rooms. The journey commenced with a single tutorial on HTML in 2006 and elated by the response it generated, we worked our way to adding fresh tutorials to our repository which now proudly flaunts a wealth of tutorials and allied articles on topics ranging from programming languages to web designing to academics and much more.

```
    </div>
    </body>
</html>
```

If user wants to make text as **new paper without line**, it can be done as shown below.

```
.multi {
  /* Column count property */
  -webkit-column-count: 4;
  -moz-column-count: 4;
  column-count: 4;
  /* Column gap property */
  -webkit-column-gap: 40px;
  -moz-column-gap: 40px;
  column-gap: 40px;
}
```

It will produce the following result −

Tutorials Point originated from the idea that there exists a class of readers who respond better to online content and prefer to learn new skills at their own pace from the comforts of their drawing rooms. The journey commenced with a single tutorial on HTML in 2006 and elated by the response it generated, we worked our way to adding fresh tutorials to our repository which now proudly flaunts a wealth of tutorials and allied articles on topics ranging from programming languages to web designing to academics and much more

CSS3 - Text

CSS3 contains several extra features.

- text-overflow
- word-wrap
- word-break

The most commonly used property in CSS3

Values	Description
text-align-last	Used to align the last line of the text
text-emphasis	Used to emphasis text and color
text-overflow	used to determines how overflowed content that is not displayed is signaled to users
word-break	Used to break the line based on word
word-wrap	Used to break the line and wrap onto next line

CSS3 Text Overflow

The CSS3 text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.It can be clipped.

This is some long text that

or it can be rendered as an ellipsis (...):

This is some long text that….

The CSS code is as follows:

Example

```
p.test1 {
   white-space: nowrap;
   width: 200px;
   border: 1px solid #000000;
   overflow: hidden;
   text-overflow: clip;
}

p.test2 {
   white-space: nowrap;
```

```
    width: 200px;
    border: 1px solid #000000;
    overflow: hidden;
    text-overflow: ellipsis;
}
```

The following example shows how the overflowed content can be displayed when hovering over the element:

Example

```
div.test:hover {
    text-overflow: inherit;
    overflow: visible;
}
```

CSS3 Word Wrapping

The CSS3 word-wrap property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside:

This paragraph
contains a very long

word:

thisisaveryveryveryveryveryverylongword.

The long word will

break and wrap to

the next line.

The word-wrap property allows to force the text to wrap - even if it means splitting it in the middle of a word:

This paragraph
contains a very long

word: thisisaveryveryve

ryveryveryverylongword.

The long word will

break and wrap to

the next line.

The CSS code is as follows:

Example

Allow long words to be able to be broken and wrap onto the next line:

```
p {
   word-wrap: break-word;
}
```

CSS3 Word Breaking

The CSS3 word-break property specifies line breaking rules.

This paragraph
contains some
text. This line
will-break-at-hyphens.

This paragraph co
ntains some text. The lines will break
at any character.

The CSS code is as follows:

Example

```
p.test1 {
   word-break: keep-all;
}
p.test2 {
   word-break: break-all;
}
```

2.3 Formatting Text and Fonts

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

CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Font Family

The font family of a text is set with the font-family property.The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on. If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".More than one font family is specified in a comma-separated list.

Example

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

Font Style

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

This property has three values:

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

Example

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

Font Size

The font-size property sets the size of the text.Being able to manage the text size is important in web design. However, font size adjustments should not be used to make paragraphs look like headings, or headings look like paragraphs. The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Set Font Size With Pixels

Setting the text size with pixels gives full control over the text size.

Example

```
h1 {
   font-size: 40px;
}
h2 {
   font-size: 30px;
}
p {
   font-size: 14px;
}
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use **em** instead of pixels. 1**em** is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.The size can be calculated from pixels to **em** using this formula: *pixels*/16=**em**

Example

```
h1 {
   font-size: 2.5em; /* 40px/16=2.5em */
}
h2 {
    font-size: 1.875em; /* 30px/16=1.875em */
}
p {
    font-size: 0.875em; /* 14px/16=0.875em */
}
```

In the example above, the text size in em which is the same as the one in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Font Weight

The font-weight property specifies the weight of a font:

Example

```
p.normal {
    font-weight: normal;
}
p.thick {
    font-weight: bold;
}
```

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
p.normal {
    font-variant: normal;
}
p.small {
    font-variant: small-caps;
}
```

Text Indentation

The text-indent property is used to set the indentation of the first line of text of an element. Possible values for the text-indent property are: ***percentage*** (%), ***length*** (specifying indent space) or inherit.

The following example demonstrates how to indent the first line of a paragraph.

```
p {
   text-indent: 100px;
}
```

Word Spacing

The word-spacing property used to sets the spacing between the words.

- Word spacing can be affected by text justification. The text-align property is used.
- When whitespace is preserved all space characters are affected by the word spacing. The CSS white-space property is used.

Possible values for the word-spacing property are: *length* (specifying the space to be inserted between words), normal and inherit.

Example

```
p.one {
   word-spacing: 20px;
}
p.two {
   word-spacing: 20px;
   text-align: justify;
}
p.three {
  word-spacing: 20px;
   white-space: pre;
}
```

Letter Spacing

The letter-spacing property is used to set extra spacing between the characters of text.Possible values for this property are: ***length*** (specifying the extra space to be inserted between characters in addition to the default inter-character space), normal and inherit.

Example

```
h1 {
   letter-spacing: -3px;
}
p {
   letter-spacing: 10px;
}
```

Line Height

The line-height property defines height (also called ***leading***) of a line of text.Possible values for this property are: ***percentage*** (%), ***length***, ***number***, normal and inherit.

Example

```
p {
   line-height: 1.2;
}
```

When the value is a number, the line height is calculated by multiplying the element's font size by the number. While, percentage values are relative to the element's font size. If the value of the line-height property is greater than the value of the font-size for an element, this difference (called the ***"leading"***) is cut in half (called the ***"half-leading"***) and distributed evenly on the top and bottom of the in-line box.

Example

```
p {
   font-size: 14px;
   line-height: 20px;
   background-color: #f0e68c;
}
```

Property Values

The following table describes the values of the property used with font.

Value	Description
font-style	Sets the font style.
font-variant	Sets the font variant.
font-weight	Sets the font weight.
font-size	Sets the font size.
line-height	Sets the line height.
font-family	Specifies the font family.
initial	Sets this property to its default value.
inherit	If specified, the associated element takes the computed value of its parent element font property.
1.1.1.13.1.1 And the following values refer to system fonts:	
caption	The font used for captioned controls (e.g., buttons, drop-downs, etc.).
icon	The font used to label icons.
menu	The font used in menus (e.g., dropdown menus and menu lists).
message-box	The font used in dialog boxes.
small-caption	The font used for labeling small controls.
status-bar	The font used in window status bars.

CSS Colors

Colors in CSS are most often specified by:

- a valid color name - like "red"
- an RGB value - like "rgb(255, 0, 0)"
- a HEX value - like "#ff0000"

Setting Color Property

The color property typically defines the color of the text content of an element. For instance, the color property specified in the body selector defines the default text color (foreground color in general) for the whole page.

Example

```
body {
   color: #000000;
}
```

Color Names

Colors set by using color names:

Example

Color	Name
🟥	Red
🟩	Green
🟦	Blue
🟧	Orange
🟨	Yellow
▨	Cyan
⬛	Black

RGB (Red, Green, Blue)

RGB color values can be specified using this formula: rgb(red, green, blue).Each parameter (red, green, blue) defines the intensity of the color between 0 and 255.For example, rgb(255,0,0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

Example

Color	RGB
🟥	rgb(255,0,0)
🟩	rgb(0,255,0)
🟦	rgb(0,0,255)
🟧	rgb(255,165,0)
🟨	rgb(255,255,0)
🟦	rgb(0,255,255)

Shades of grey are often defined using equal values for all the 3 light sources:

Example

Color	RGB
⬛	rgb(0,0,0)
▨	rgb(128,128,128)
⬜	rgb(255,255,255)

Hexadecimal Colors

RGB values can also be specified using **hexadecimal** color values in the form: #***RRGGBB***, where RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and FF (same as decimal 0-255).

For example, #FF0000 is displayed as red, because red is set to its highest value (FF) and the others are set to the lowest value (00). **Note:** HEX values are case-insensitive. "#ff0000" is the same as "FF0000".

Example

Color	HEX
🟥	#FF0000
🟩	#00FF00
🟦	#0000FF
🟧	#FFA500
🟨	#FFFF00
🟦	#00FFFF

CSS Background

Background properties are used to define background styles for the elements.

Background Properties

The CSS provide several properties for styling the background of an element, like: background-color, background-image, background-repeat, background-attachment and background-position.

Background Color

The background-color property is used to set the background color of an element.The background color of a page is set like this:

Example

```
body {
  background-color: lightblue;
}
```

With CSS, a color is most often specified by:

- ✓ a valid color name - like "red"
- ✓ a HEX value - like "#ff0000"
- ✓ an RGB value - like "rgb(255,0,0)"

In the example below, the <h1>, <p>, and <div> elements have different background colors:

Example

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

Background Image

The background-image property set an image as a background of an HTML element. By default, the image is repeated so it covers the entire element.

Example

```
body {
   background-image: url("paper.gif");
}
```

Background Repeat

By default the background-image property repeats an image both horizontally and vertically.

By using background-repeat property tilting of background image can be controlled for a background of an html element. The background image can be set to repeat vertically (y-axis), horizontally (x-axis), in both directions, or in neither direction. The example below which demonstrates how to set the gradient background for a web page by repeating the sliced image horizontally.

```
body {
   background-image: url("gradient.png");
   background-repeat: repeat-x;
}
```

Background Attachment

The background-attachment property determines whether the background image is fixed with regard to the viewport or scrolls along with the containing block.

Example

```
body {
  width: 250px;
   height: 200px;
  overflow: scroll;
   background-image: url("recycle.jpg");
   background-attachment: fixed;
}
```

Background Position

The background-position property is used to control the position of the background image.If no background-position has been specified, the image is placed at the default top-left position of the element i.e. at (0,0).

Example

```
   body {
 background-image: url("tree.png");
 background-repeat: no-repeat;
```

```
}
```

In the following example, the background image is positioned at top-right corner and the position of the image is specified by the background-position property.

Example

```
body {
    background-image: url("tree.png");
    background-repeat: no-repeat;
    background-position: 100% top;
}
```

The Background Shorthand Property

There are many properties to consider when dealing with the backgrounds. It is also possible to specify all these properties in one single property, to shorten the code. This is called a shorthand property.

The background property is a shorthand property for setting all the individual background properties (i.e., background-color, background-image, background-repeat, background-attachment and background-position) at once. For example:

```
body {
   background-color: #f0e68c;
   background-image: url("smiley.png");
   background-repeat: no-repeat;
   background-attachment: fixed;
   background-position: 250px 25px;
}
```

Using the shorthand notation the above example can be written as:

```
  body {
     background: #f0e68c url("smiley.png") no-repeat fixed 250px 25px;
  }
```

When using the background shorthand property the order of the property values should be.

background: *color image repeat attachment position*;

If the value for an individual background property is missing or not specified while using the shorthand notation, the default value for that property will be used instead, if any.

The background properties do not inherit, but the parent element's background will be visible through by default because of the initial (i.e. default) transparent value of the background CSS property.

2.4 Exploring CSS Class and ID Attributes:

Defining CSS Class attribute

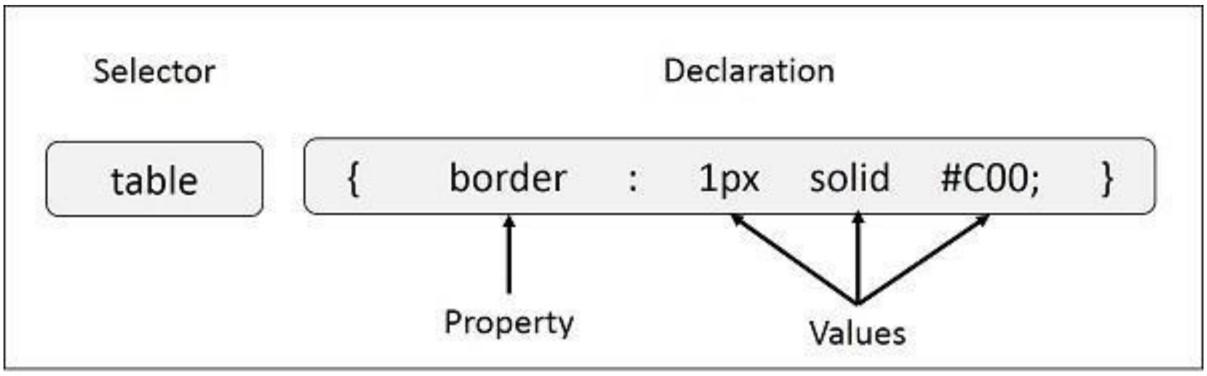
A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in a document. A style rule is made of three parts −

Selector − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

Property - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

Value - Values are assigned to properties.

For example, color property can have value either red or #F1F1F1 etc. CSS Style Rule Syntax can be as follows − selector { property: value }



Example:

A table border can be defined as follows −table{ border :1px solid #C00; } Here table is a selector and border is a property and given value 1px solid #C00 is the value of that property.

Selectors can be defined in various simple ways based on comfort.

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {
  color: #36CFFF;
}
```

The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −

```
* {
  color: #000000;
}
```

This rule renders the content of every element in our document in black.

The Descendant Selectors

A style rule can be applied to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <em> element only when it lies inside <ul> tag. ul em {

```
  color: #000000;
}
```

The Class Selectors

The style rules can be defined based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
  color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

```
h1.black        {
color:
#000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to black. More than one class selectors can be applied to a given element.

Consider the following example:

```
<p class="center bold">
  This para will be styled by the classes center and bold. </p>
```

The ID Selectors

Style rules can be defined based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black {

   color: #000000;
}
```

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example − h1#black {   color: #000000;

}

This rule renders the content in black for only <h1> elements with id attribute set to black. The true power of id selectors is when they are used as the foundation for descendant selectors, For example:

```
#black    h2    {
color:
#000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having id attribute set to black.

The Child Selectors

descendant selectors is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example − body > p { color: #000000;

}

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like <div> or <td> would not have any effect of this rule.

The Attribute Selectors

Styles can be applied to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of text − input[type = "text"]{   color: #000000;
}
The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

p[lang] - Selects all paragraph elements with a lang attribute.

p[lang="fr"] - Selects all paragraph elements whose lang attribute has a value of exactly "fr".

p[lang~="fr"] - Selects all paragraph elements whose lang attribute contains the word "fr".

p[lang|="en"] - Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

Multiple Style Rules

Multiple style rules for a single element can be defined. These rules are defined to combine multiple properties and corresponding values into a single block as defined in the following example − h1 { color: #36C; font-weight: normal; letter-spacing: .4em; margin-bottom: 1em; text-transform: lowercase;

}

Here all the property and value pairs are separated by a semi colon (;).They can be kept in a single line or multiple lines. For better readability it is kept into separate lines.

Grouping Selectors

A style can be applied to many selectors. Just separate the selectors with a comma, as given in the following example − h1, h2, h3 { color: #36C; font-weight: normal; letter-spacing: .4em; margin-bottom: 1em; text-transform: lowercase;

}

This defined style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

Various class selectors can be combined together as shown below −

```
#content, #footer, #supplement {
  position:        absolute;
left: 510px;
  width: 200px;
}
```

Defining Effects with CSS

With CSS3 shadow can be added to text and to elements with the following properties:

- text-shadow
- box-shadow

```
<!DOCTYPE html>
<html>
<head><style>
```

h1 {

```
    text-shadow: 2px 2px red;
}
</style>
</head>
<body>
<h1>Text-shadow effect!</h1>
<p><b>Note:</b> Internet Explorer 9 and earlier versions, do not support the text-shadow
property.</p>
</body>
</html>
```

Multiple Shadows

To add more than one shadow to the text, you can add a comma separated list of shadows.

```
h1
{
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

CSS3 BOX-SHADOW PROPERTY

The CSS3 box-shadow property applies shadow to elements. In its simplest use, only the horizontal shadow and the vertical shadow can be specified.

```
<!DOCTYPE html>
<html>
<head><style>  div {      width:
300px;            height:  100px;
padding:                     15px;
background-color:         yellow;
box-shadow: 10px 10px;
}
</style>
</head>
<body>
<div>This is a div element with a box-shadow</div>
</body>
</html>
```

CSS Lists

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow us to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

The **list-style-type** property specifies the type of list item marker.

The following example shows some of the available list item markers:

```
<!DOCTYPE html>
<html>
<head>
<style>
        ul.a {
           list-style-type: circle;
        }
        ul.b {
           list-style-type: square;
        }
        ol.c {
           list-style-type: upper-roman;
        }
        ol.d {
           list-style-type: lower-alpha;
        }
</style>
</head>
<body>

<p>Example of unordered lists:</p>
        <ul class="a">
```

```
          <li>Coffee</li>
          <li>Tea</li>
          <li>Coca Cola</li>
        </ul>
        <ul class="b">
          <li>Coffee</li>
          <li>Tea</li>
          <li>Coca Cola</li>
        </ul>
<p>Example of ordered lists:</p>
        <ol class="c">
          <li>Coffee</li>
          <li>Tea</li>
          <li>Coca Cola</li>
        </ol>
        <ol class="d">
          <li>Coffee</li>
          <li>Tea</li>
          <li>Coca Cola</li>
        </ol>
</body>
</html>
```

The out put is as follows

```
Example of unordered lists:

  ○  Coffee
  ○  Tea
  ○  Coca Cola

  ▪  Coffee
  ▪  Tea
  ▪  Coca Cola

Example of ordered lists:

         I.   Coffee
        II.   Tea
       III.   Coca Cola

         a.   Coffee
         b.   Tea
```

c. Coca Cola

List - Shorthand property

The **list-style** property is a shorthand property. It is used to set all the list properties in one declaration:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
   list-style: square inside url("sqpurple.gif");
   }
</style>
</head>
<body>
<ul>
      <li>Coffee</li>
      <li>Tea</li>
      <li>Coca Cola</li>
</ul>
</body>
</html>
```

The output is as follows

▨ Coffee
▨ Tea
▨ Coca Cola

When using the shorthand property, the order of the property values are:

- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)

If one of the property values above are missing, the default value for the missing property will be inserted, if any.

Styling List with Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the <ol> or <ul> tag, affects the entire list, while properties added to the <li> tag will affect the individual list items:

```
<!DOCTYPE html>
<html>
<head>
<style>
ol {
   background: #ff9999;
   padding: 20px;
}

ul {
   background: #3399ff;
   padding: 20px;
}

ol li {
   background: #ffe5e5;
   padding: 5px;
   margin-left: 35px;
}

ul li {
   background: #cce5ff;
   margin: 5px;
}
</style>
</head>
<body>
<h1>Styling Lists With Colors:</h1>
<ol>
     <li>Coffee</li>
     <li>Tea</li>
     <li>Coca Cola</li>
</ol>
<ul>
```

```
        <li>Coffee</li>
        <li>Tea</li>
        <li>Coca Cola</li>
</ul>
</body>
</html>
```

CSS Tables

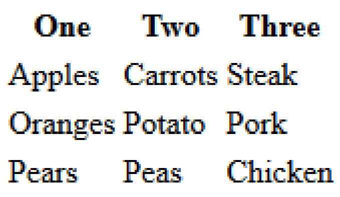
The look of an HTML table can be greatly improved with CSS.

Basic CSS Table Structure

Let's make a simple 3x4 table containing columns for different food groups. Our HTML would look something like:

```
<table>
<tr>
   <th>One</th>
   <th>Two</th>
   <th>Three</th>
</tr>
<tr>
   <td>Apples</td>
   <td>Carrots</td>
   <td>Steak</td>
</tr>
<tr>
   <td>Oranges</td>
   <td>Potato</td>
   <td>Pork</td>
</tr>
<tr>
   <td>Pears</td>
   <td>Peas</td>
   <td>Chicken</td>
</tr>
</table>
```

Within a brand new HTML document, before any resets or CSS, you'd probably have something like this on your hands:



One	Two	Three
Apples	Carrots	Steak
Oranges	Potato	Pork
Pears	Peas	Chicken

Table Borders

To specify table borders in CSS, use the **border** property. The example below specifies a black border for <table>, <th>, and <td> elements:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table, th, td {
  border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table and the <th> and <td> elements have separate borders.

Collapse Table Borders

The **border-collapse** property sets whether the table borders should be collapsed into a single border:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
   border-collapse: collapse;
}

table, td, th {
   border: 1px solid black;
}
</style>
</head>
```

```
<body>
<h2>Let the borders collapse:</h2>
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>
<p><b>Note:</b> If a !DOCTYPE is not specified, the border-collapse property can produce
unexpected results
in IE8 and earlier versions.</p>
</body>
</html>
```

The output is as follows

Let the borders collapse:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Note: If a !DOCTYPE is not specified, the border-collapse property can produce unexpected results in IE8 and earlier versions.

If only a border around the table is required then, specify only the border property for <table>:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
table {
    border-collapse: collapse;
    border: 1px solid black;
}
</style>
</head>
<body>
<h2>Single Border Around The Table:</h2>
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>
</body>
</html>
```

The output is as follows

Single Border Around The Table:

Firstname	**Lastname**
Peter	Griffin
Lois	Griffin

Table Width and Height

Width and height of a table are defined by the **width** and **height** properties.

The example below sets the width of the table to 100%, and the height of the <th> elements to 50px:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
    border: 1px solid black;
}
table {
    border-collapse: collapse;
    width: 100%;
}
th {
    height: 50px;
}
</style>
</head>
<body>
<h2>The width and height Properties</h2>
<p>Set the width of the table, and the height of the table header row:</p>
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
</tr>
</table>
</body>
</html>
```

The width and height Properties

Set the width of the table, and the height of the table header row:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Horizontal Alignment

The **text-align** property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

Example

```
th {
   text-align: left;
}
```

By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

Vertical Alignment

The **vertical-align** property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

Example

```
td {
    height: 50px;
    vertical-align: bottom;
}
```

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

Table Padding

To control the space between the border and the content in a table, use the **padding** property on <td> and <th> elements:

Example

```
th, td {
    padding: 15px;
    text-align: left;
}
```

Horizontal Dividers

Add the **border-bottom** property to <th> and <td> for horizontal dividers:

Example

```
th, td {
    border-bottom: 1px solid #ddd;
}
```

Hoverable Table

Use the **:hover selector** on <tr> to highlight table rows on mouse over:

Example

```
tr:hover {background-color: #f5f5f5}
<!DOCTYPE html>
<html>
<head>
<style>
table {
    border-collapse: collapse;
    width: 100%;
}
```

```
th, td {
   padding: 8px;
   text-align: left;
   border-bottom: 1px solid #ddd;
}

tr:hover{background-color:#f5f5f5}
</style>
</head>
<body>
<h2>Hoverable Table</h2>
<p>Move the mouse over the table rows to see the effect.</p>
<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
</table>
</body>
</html>
```

Striped Tables

For zebra-striped tables, use the **nth-child()** selector and add a **background-color** to all even (or odd) table rows.

Example

```
tr:nth-child(even) {background-color: #f2f2f2}
```

Table Color

The example below specifies the background color and text color of <th> elements.

Example

```
th {
   background-color: #4CAF50;
   color: white;
}
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content. Add a container element (like <div>) with **overflow-x:auto** around the <table> element to make it responsive.

Example

```
<div style="overflow-x:auto;">
<table>
... table content ...
</table>
</div>
```

CSS Forms

The look of an HTML form can be greatly improved with CSS.

Styling Input Fields

Use the **width** property to determine the width of the input field:

Example

```
input {
   width: 100%;
}
```

If a style is to be applied to a specific input type, use attribute selectors:

- input[type=text] - will only select text fields
- input[type=password] - will only select password fields
- input[type=number] - will only select number fields

Padded Inputs

Use the padding property to add space inside the text field. When there are many inputs after each other, add some margin, to add more space outside of them.

Example

```
input[type=text] {
   width: 100%;
   padding: 12px 20px;
   margin: 8px 0;
   box-sizing: border-box;
}
```

We have set the **box-sizing** property to **border-box**. This makes sure that the padding and eventually borders are included in the total width and height of the elements.

Bordered Inputs

Use the border property to change the border size and color, and use the **border-radius** property to add rounded corners.

Example

```
input[type=text] {
   border: 2px solid red;
   border-radius: 4px;
}
```

If a bottom border only is required , use the **border-bottom** property.

Example

```
input[type=text] {
   border: none;
   border-bottom: 2px solid red;
}
```

Colored Inputs

Use the **background-color** property to add a background color to the input, and the color property to change the text color.

Example

```
input[type=text] {
   background-color: #3CBC8D;
   color: white;
}
```

Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on).This behavior can be removed by adding **outline: none**; to the input.

Use the **: focus** selector to do something with the input field when it gets focus.

Input with icon/image

If an icon is required inside the input, use the **background-image** property and position it with the **background-position** property. Also notice that a large left padding is added to reserve the space of the icon.

Example

```
input[type=text] {
   background-color: white;
   background-image: url('searchicon.png');
   background-position: 10px 10px;
   background-repeat: no-repeat;
   padding-left: 40px;
}
```

Animated Search Input

transition property is used to animate the width of the search input when it gets focus.

Styling Textareas

The **resize** property is used to prevent textareas from being resized (disable the "grabber" in the bottom right corner).

Example

```
textarea {
   width: 100%;
   height: 150px;
   padding: 12px 20px;
   box-sizing: border-box;
   border: 2px solid #ccc;
   border-radius: 4px;
   background-color: #f8f8f8;
   resize: none;
}
```

UNIT-3

CLIENT SIDE SCRIPTING (JAVA SCRIPT)

Objectives:

At end the this unit, students can

- Define scripting language
- Need of scripting language in programming
- Differentiate between client side scripting and server side scripting
- Explain the life span of variables
- Discuss about different types of operators
- Describe the conditional statements
- Define functions
- Describe execution of deferred scripts
- Explain DOM
- State the types of predefined objects
- State the types of dialog boxes
- Define events and its handlers
- State the types of event handlers
- Define forms
- Describe forms element properties
- Define image map
- Differentiate between client side and server side image maps
- Describe status bar and cookies
- Define live connect
- Describe java console
- Explain the java to java script communications.

3.1 JAVA SCRIPT BASICS

A scripting language is a programming language which can be embedded within HTML, commonly are used to add functionality to a Web page. It employs a high-level construct to interpret and execute one command at a time. The functionalities it adds are , different menu styles or graphic displays or to serve dynamic advertisements. There are two types of scripting languages. They are 1. Client side scripting languages 2. Server side scripting language.

✓ The client-side scripting languages, affects the data that the end user sees in a browser window.

- ✓ The server-side scripting language manipulates the data, usually in a database, on the server.

Scripting languages came about largely because of the development of the Internet as a communications tool. JavaScript, ASP, JSP, PHP, Perl, and Python are examples of scripting languages.

3.1.1 Need of scripting Languages:

i) It is used for validating the input data given by the user in the forms

ii) The data type conversion takes automatically

iii) We can write event driven programs using scripting language. i.e mouse click event, keyboard event.

iv) We can create new variable without specifying its type.

v) A script is used in order to extend application functionalities, or to execute simple tasks by using pre-existing components

Structure of java script programs

Java script codes are written within **html** program using the **script tags** <script>…..</Script> as follows :

```
<html>
        <head>
                <Title> Script Example</Title> <head>
                <body>
                      <script language="Javascript" type="text/javascript">
                      Statement1
                      ………..
                      Statement n
                      </script>
            </body>
 </html>
```

Where

$$\left.\begin{array}{ll} \text{<html>……</html>} & \text{<head>……<head>} \\ \text{<title>…….</title>} & \text{<body>……</body>} \end{array}\right\} \text{HTML tags}$$

<script>……</script> -----------→Script tag

Language – Scripting language used

Statements – valid java script language statement

3.1.2 Variables and data types

Declaring variable

Variable is a name given to memory location to store data. The stored data changes during the execution of the program.

Variables are declared with the **var** keyword as follows.

var name = value ;

where

var – keyword and it is optional

name – valid user defined name

value – data to be stored

Example

var name = “Hello”

Life span of variable

Life span of variable means how long a variable retains a given value during the execution of the program. JavaScript variables have only two scopes. They are

i) Local variable

ii) Global variable

Local variable

Variables which are declared inside a function are called local variables. A local variable will be visible only within a function where it is defined.

Example:

```
<html>
        <body>
                <script type = "text/javascript">
                   function checkscope( )
                   {
                   var myvar = "local";  // Declare a local variable
                   document.write(myvar);
                   }
                </script>
        </body>
</html>
```

Global Variable

A global variable has global scope which means it can be defined anywhere in JavaScript code. These variables are alive and active throughout the program.

```
<html>
 <body>
             <script type = "text/javascript">
              var myVar = "global"; // Declare a global variable
              function checkscope( )
              {
             --------
             --------
              }
              </script> </body>
</html>
```

Data types

In java script there is no need to specify the data type of the variables. These variables can hold many **data types** such as numbers, strings and more:

Example:

i) var length = 16;   // Number

This assigns an integer data value to the variable length

ii) var lastName = "Johnson";   // String

This assigns a string data to the variable lastname.

JavaScript Types are Dynamic.

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example:

```
var x;              // Now x is undefined
var x = 5;           // Now x is a Number
var x = "Welcome";   // Now x is a String
```

Literals

Literals are fixed values that does not change during the script execution. The following types of literals are used in java script.

- Numbers
- Boolean
- String
- Null
- Undefined

3.1.3 Operators

An operator is a symbol which represents an operation that can be performed on data. There are six operators available in java script to carry out the operation. They are

- Assignment Operators
- Short hand Assignment Operators
- Comparison Operators
- Conditional (or ternary) Operators
- Logical (or Relational) Operators

1. Assignment operators

Assignment operators are used to assign values to JavaScript variables. This is done with the help of the assignment operator = .

The general form is

Variable = constant or variable or expression

Rules:

i) only one variable is allowed on the LHS of assignment operator.

ii) All operation must be written explicity.

Example:

1. var x = 100

The value 100 is assigned to the variable x.

2. Short hand assignment operators

Short hand assignment operators are operators which are used to simplify the coding of certain type of assignment statement. The general form is

Variable operator = expression

Example

Operator	Meaning	Example
+=	Value of LHS variable will be added to the RHS value and is assigned to LHS variable	x=x+10 can be written as x+=10
-=	RHS value will be subtracted from LHS variable and is assigned to LHS variable	X=x*10 can be written as x-=10
*=	Value of LHS variable will be multiplied by the RHS value and is assigned to LHS variable.	x=x*10 can be written as x*=10
/=	Value of LHS variable will be divided by the RHS value and is assigned to LHS variable	x=x/10 can be written as x/=10
%=	Value of LHS variable will be divided by the RHS value and remainder will be stored in LHS variable	x=x%10 can be written as x%=10

iii) Comparison operator

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Operator	Operation	Example
>	greater than	8>5
<	less than	2<7
>=	greater than equal to	8>=B
<=	less than equal to	10<=7
==	equal to	10==(A+B)

iv) Computational operators

Arithmetic operators are used to do arithmetic calculations. There are two types

i) Binary operators

ii) Unary operators

i) Binary operators:

Binary operators need two operands for operations. They are

Operator	Operation	Example
+	Addition	x=5+3
-	Subtraction	y=x-10
*	Multiplication	Z=5*3
/	Division	X=10/3
%	Modulo operator	X=5%3

ii) Unary operator

Unary operators need only one operand for operation. They are

Operator	Operation	Example
-	Unary minus	-10
++	Increment	++i
--	Decrement	--i

V) Logical Operator:

Logical operators are used to determine the logic between variables or values.

Operator	Meaning	Example
&&	AND	A>B&&x==y
||	OR	A>B||X<=Z
!	NOT	!X

3.1.4 Control Structure

Control Structures are used to transfer java script program control from one statement to any other statement. The different control statements are,

i) Conditional Statement

ii) Looping Statement

i) Conditional Statement:

Decision making statements are used to skip or to execute a group of statements based on the result of some condition. The decision making statements are,

1. if statement
2. if … else statement
3. nested if
4. Switch statement.

1. if statement

Logical if statement is used to execute or skip one statement or group of statements for a particular condition. The general form is

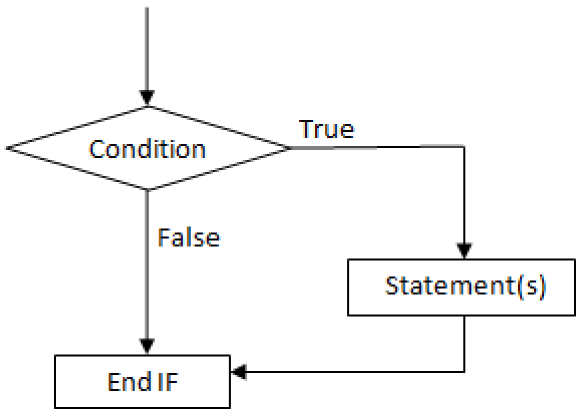
```
If (test condition)
        {
        Statement block;
        }
next statement;
```

When this statement is executed, the computer first evaluates the value of the test condition.

If the value is true, statement block and next statement are executed sequentially.

If the value is false, statement block is skipped and execution starts from next statement.

Flow diagram



Example:

```
If(mark ==30)
{
        mark=mark+10;
}
```

The above statements test the mark of the student. If the student mark is 30, then 10 mark will be added to his marks. For other marks no additional marks are added.

2. if--else statement

If--else statement is used to execute one group of statements if the test condition is true or other group if condition is false.

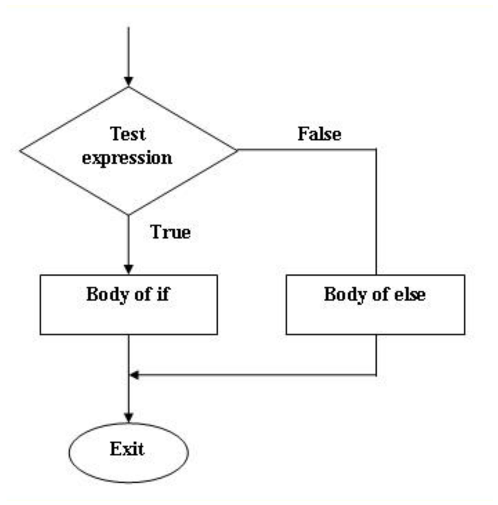
The general form is

```
If (test condition)
        {
        Statement block 1;
        }
else
        {
        Statement block 2;
        }
next statement;
```

When this statement is executed, the computer first evaluates the value of the test condition. if the value is true, statement block-1 is executed and the control is transferred to next statement.

If the value is false, the statement block-2 is executed and the control is transferred to next statement.

Flow diagram



Example:

```
If(mark>=30)
        {
        mark=mark+10;
        }
else
        {
        Mark=mark+5;
        }
```

This program tests are mark of the student. If it is greater than 35, 10 marks will be added. Else 5 marks will be added.

3. switch statement

The switch statement can have a number of possible execution paths. This permits any number of branches.

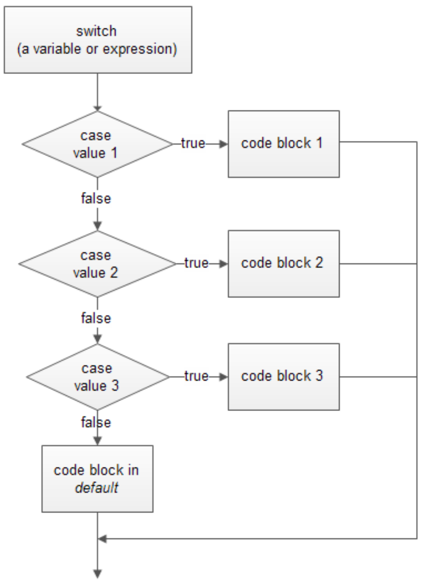
The general form is

```
switch(expression)
{
            case label         1:
        statement   block-1;
      break;

         --------
         --------
            case label         n:
        statement    block-n
                        break;

    default:
      break
}
next statement
```

When this statement is executed the computer first evaluates the values of the expression in the keyword switch. This values is successively compared with the case label 1, label 2, label n. if a case label matches with the value, the statement block associated with the case label is executed.

Then the control is transferred to the next statement. If none of the case matches with the value, the default statement block is executed.

Flow diagram



Example

```
html>
        <body>
                <script>
        var d = new Date();
        day=d.getDay();
                switch (day)
                {
                 case 0:
                        document.write(“SUNDAY”);
                 break;
                case 1:
                        document.write(“MONDAY”);


                  break;
                default:
                        document.write(“great week”);
                }
                </script>
        </body>
</html>
```

ii) Looping statement

Looping statements are the statements execute one or more statement repeatedly several number of times.

loop structures are used to execute a group of statements repeatedly until some condition is satisfied. In general, the statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. The looping structures are

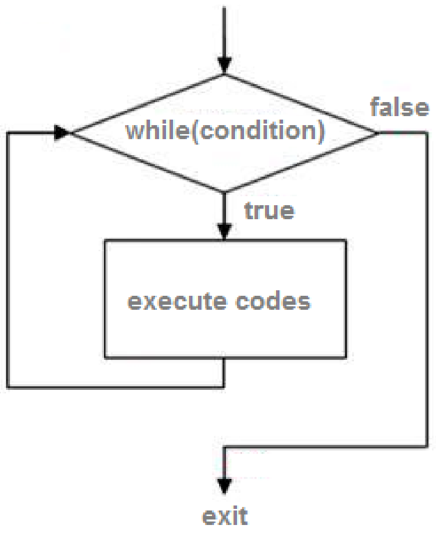
- while structure
- do-while structure
- for structure
- for-in structure

1. While structure

It repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.

```
While(test condition)
          {
          body of the loop;
          }
next statement
```

Flow diagram



Example

```
int i=1;
 while(i<5)
 {
document.write("Welcome");
  i++;
 }
```

These program statements are used to display the message welcome, four times. Initially the condition i<5 is tested. If it is false the program ends without executing the body of the loop. If it is true the body of the loop will be executed repeatedly four times.

2. do-while loop

A **do-while** loop is similar to a while loop, except that a do-while loop is execute at least one time.

A do while loop is a control flow statement that executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given condition at the end of the block (in while).

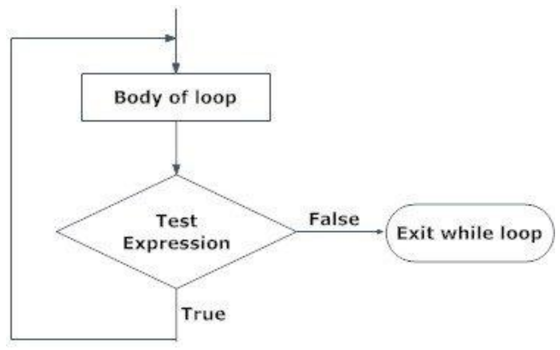
The general form is

```
do
{
      body of the loop;
}
while(test condition);
next statement;
```

When this statement is executed the body of the loop is executed first. Then the test condition is evaluated. If the value is false, the control is transferred to the next statement.

If the value is true the body of the loop is executed repeatedly until the test condition becomes false. When the test condition becomes false the control is transferred to the next statement

Flow diagram



Example

```
var i=1;
do
{
document.write("welcome");
i++;
}
while(i<5);
```

These program statements are used to display the message welcome, four times.

Initially the body of the loop is executed once and the condition i<5 is tested. If it is false the program ends.

If it is true the body of the loop is executed repeatedly four times

3. for statement

For statement is used to execute a statement or a group of statements repeatedly for a known number of times.

The general forms is

```
for(control variable; test condition; increment or decrement)

{

        body of the loop;

}

next statement;
```

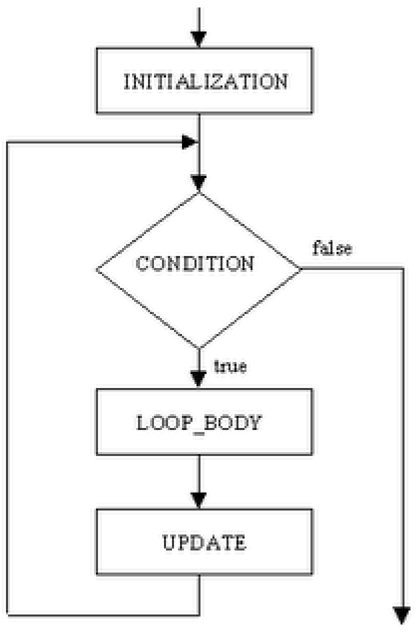
i) The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables.

ii) Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.

iii) After the body of the 'for' loop executes, the flow of control jumps back up to the **increment** statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

iv) The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

Flow diagram



Example

```
sum=0;

for(i=1; i<100; i++)

{

sum = sum+i;

}

document.write(“sum=”+sum);

}
```

4.for-in statement

The for/in statement loops through the properties of an object.

The block of code inside the loop will be executed once for each property.

The general form is

```
for (var in object)

{
statement to be executed
}
```

When this statement is executed the body of the loop will be executed number of times equal to the number of elements in the array.

Example

```
<html>

        <body>

                <script type=”text/javascript”>

                var x;

                var name = new Array();

                   name[0]= “welcome”

                   name[1]= “great day”

                   name[2]= “good morning”

                   for(x in name)

                   {

                   document.write[name[x]);

                   }

                </script>

        </body>

        </html>
```

During the execution of the above program the write statement will be executed three times.

Output

```
welcome

great day

good morning
```

Break statement

The break statement "jumps out" of a loop.

Break statement is used to exit from a loop while the test condition is true. This statement can be used within a for, while, do while or switch statement.

The general form is

```
break;
```

When the break statement is executed inside a loop, the execution of the loop is terminated and the program continues with the statement following the loop.

If break statement is used in nested loops, it will exit from the loop containing it.

Example

```
var i =0;

while(i<6)

{

If(i==3)

break;

i++;

}

document.write(i);   → i=3
```

the output is 3

Continue statement

The **continue statement** breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop. In other words, the statement after the keyword continue will be skipped and the control is transferred to the beginning of the loop.

The general form is

```
continue;
```

Example:

```
n=0;

for(i=1;i<5;i++)

{

----------

----------

If(i==3)

continue;

n=n+i;

}
```

During the execution if the value of i=3, the statement below the keyword continue is skipped and the control is transferred to the beginning of the loop.

Advantages of looping statements

- ✓ Reduce length of Code
- ✓ Take less memory space.
- ✓ Burden on the developer is reducing.
- ✓ Time consuming process to execute the program is reduced.

3.2 Object based programming and message boxes

Java script is a power full object based language. It is not hundred percentage object oriented language as java. Therefore with the help of java script we can create self contained java script objects so that it can be reused.

3.2.1 Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

The general form is

```
function functionname(parameters)
{
 local variable declarations
valid statements
return();
}
```

Where

function, return-keyword

functionname – user defined name

parameters – variable to receive values from function call

Example

```
function area(w1, w2, h)

{

var area = (w1+w2)*h/2;

document.write (area);

}
```

In this **area** is name of the function. It has three arguments w1, w2, h. This can be called from anywhere on the page with the values of w1, w2 and h.

Calling the function

The defined function can be called from anywhere in our java script code

The general form for calling the function is

```
functionname(values)
```

functionname – name of the already defined function

values-values given to parameters in the calling function

Example

```
<html>

<body>

<script>

function myFunction(p1, p2) → function definition

{

var d = p1*p2;

document.write("product="+d);

}

myFunction(4, 3); → function calling
```

```
</script>

</body>

</html>
```

Return statement

Return statement is used to return a value from the function. The general form is

```
return[OR] return(expression)
```

Example

```
<html>

        <body>

                <script language = “javascript”>

                function myFunction(a, b)

                 {

                        var d = a * b;     return(d);

                }

                var s = myFunction(4, 3);

                document.write(“product = “+s);   </script>

        </body>

</html>
```

Output

12

3.2.2 Executing deferred scripts

The **defer** is only for external **scripts.** Deferred scripts(functions) are java scripts that will not do anything immediately. This can be executed by using deferred commands from outside the deferred scripts.

Three ways of calling deferred scripts

1.By using ordinary function calling method from inside or outside the scripts.

2. By user initiated events.

3. By clicking on links.

Example

```
<html>
<body>
<script language="javascript">
function sum(a,b,c)
{
var d=a+b+c;                          → deferred script
document.write("sum="+d);
}
sum(5,5,5);  → deferred script calling
</script>
</body>
</html>
```

3.2.3 Objects

JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or ***key***) and a value. A property's value can be a function, in which case the property is known as a method.

General form to create objects.

```
var objectname=new object()
```

where

object name = user defined name

new, var – keyword

object – name of the class or object

Document object

Every web page resides inside a browser window which can be considered as an object.

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The way document content is accessed and modified is called the **Document Object Model**, or **DOM**.

Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.

i) Document object properties

alinkColor - The color of active links.

bgColor - Sets the background color of the web page.

domain - The domain name of the document server.

fgColor - The text color attribute set in the <body> tag.

linkColor - The color of HTML links in the document. It is specified in the

<body> tag.

vlinkColor - The color of visited links as specified in the <body> tag

cookie - Used to identify the value of a cookie.

applets- An array containing an entry for each applet in the document.

images - An array containing an entry for each image in the document.

ii) Document object methods

Methods are useful for everything from displaying the contents of the object to the screen.

1. open()

This method is used to open the output stream to the document object for writing

2. close()

This methods is used to close the output stream.

3. write()

This method is used to write any content on the document.

4. writeln()

This method is used to write the values passed to the document object followed by a new line

Simple programs

1. Write a program to add two numbers.

```
<script>
   var num1, num2, sum;
   num1 = prompt("Enter first number");
   num2 = prompt("Enter second number");
   sum = parseInt(num1) + parseInt(num2) ;
   alert("Sum = " + sum)   </script>;
```

2. write a program to find maximum number until sees zero

```
<script>
   var max = 0;
   var num;
   num = prompt("Enter new value, or 0 to end");
   while (num != 0) {
           if (parseFloat(num) > max)
         max = num;
           num = prompt("Enter new value, or 0 to end");
   }
   document.write("<P> Max = " + max);
</script>
```

3. Write a program to concatenates two names

```
<script>
   var firstname, secondname, result;
   firstname = prompt("Enter first name");
   secondname = prompt("Enter last name");
   result = firstname + secondname ;
   alert("hello, " + result);
</script>
```

3.2.4 Predefined objects

The predefined objects do not represent any part of the web page. But these are used for writing programming functions. The important predefined objects are;

1. Array object
2. History object
3. Location object

1. Array object

An array object stores multiple values in a single variable. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Syntax

```
var arrayname = new Array(size);
```

Where

var, new	-	keyword
arrayname	-	user defined name
size	-	size of the array

Assigning values to the created array

This is use to giving initial values to the created array is called assigning values.

Syntax

```
arrayname(index) =”value”;
```

Example

var fruits = new array("banana", "orange", "grapes");

Methods in array object

i) length() – this method is used to find the number of elements in an array

ii) concat()-Returns a new array comprised of this array joined with other array(s).

iii)filter() - Creates a new array with all of the elements of this array for which the provided filtering function returns true.

iv) join()-Joins all elements of an array into a string.

v) foreach()-Calls a function for each element in an array.

vi) reverse() – It is used to reverse the order of elements in an array.

vii) shift() - Removes the first element from an array and returns that element.

viii) sort()-Sorts the elements of an array

Example

1.Writ e a program to concate two arrays

```
<html>
   <head>
               <title>JavaScript Array concat Method</title>
   </head>
              <body>
                          <script type="text/javascript">
                          var vehicle = ["car", "auto", "bike"];
                         var wheel = [4, 3 2];
                          var vehiclewheel = vehicle.concat(wheel);
                         document.write("vehiclewheel: " vehiclewheel);
                          </script>
               </body>
</html>
```

Output

```
            vechicle wheel : car,auto,bike,4,3,2
```

2. Write a program to shift array element

```
<html>
 <head>
 <title>JavaScript Array shift Method</title>
 </head>
  <body>
  <script type="text/javascript">
  var element = [105, 1, 2, 3].shift();
   document.write("Removed element is : " + element );
```

```
</script>
 </body> </html>
```

Output

```
        Removed element is : 105
```

3. write a program to sort an array

```
<html>
 <head>
 <title>JavaScript Array sort Method</title>
 </head>
 <body>
 <script type="text/javascript">
 var fruits = new Array("orange", "mango", "banana");
 var sorted = fruits.sort();
 document.write("sorted array elements : " + sorted );
 </script>
 </body>
</html>
```

Output

```
        sorted array elements: banana,mango,orange
```

2. History object

The history object represents an array of URLs visited by the user. By using this object, you can load previous, forward or any particular page.

i) History objects properties

a) current - specifies the URL of the current entry in the object

b) length - specifies the number of elements contained in the object

c) next - The URL of the next document in the history object.

d) previous - The URL of the last document in the history object.

The general form to use the above property is

```
location.property name;
```

where;

location-object name ,   property name-name of the property

ii) History objects methods

The following are the important methods in history objects. They are;

a) back()

b) forward()

c) go()

a) back()

This method is used to Go to the previous URL entry in the history list.

Example

```
<form>
<input type="button" value="go back" onclick="history.back()">
</form>
```

b) forward()

Specifies a method that loads the next URL from the history list

Example

```
<form>
<input type="button" value="go forward" onclick="history.forward()">
</form>
```

c) go()

This method is used to loads a specific URL from the history list

Example

```
    <form>
    <input type="button" value="go back" onclick="history.go(-1)">
    </form>
```

Example

```
<html>
<body>
```

```
        <script type="text/javascript">
                var ex = history.length;
                document.write("The number of pages visited
                before this page is" +ex+ " pages!!!.")
        </script>
</body>   </html>
```

3) Location Object

This location object is used to holds information on the current URL**.**

Can not be created by the user. It is a predefined object.

Can be accessed by the location property of a window object.

i) Location object properties

a) host – specifies the host and domain name or IP address of a network host.

b) hostname- Specifies the host: port portion of the URL.

c) href - Specifies the entire URL

d) pathname- Specifies the URL-path portion of the URL.

e)protocol- Specifies the beginning of the URL, including the colon.

ii) Location object methods

a) reload()-Forces a reload of the window's current document

b) replace- Loads the specified URL over the current history entry

Example

```
<html>
<body>
<button onclick="myFunction()">Load new document</button>
<script>
function myFunction()
 {
 window.location.assign("http://www.google.com”);
}
</script>
</body>
</html>
```

3.2.5 Dialog boxes

JavaScript supports three important types of dialog boxes. These dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users.

Syntax

```
Window.showmodeldialog("message")
```

Where

Window	-	object
Showmodeldialog	-	function name
Message	-	action to take

Example

```
<html>
<head>
        <script language = "java script">
                function foropen()
                {
                window.showmodeldialog("d:\test.html")
                }
        </script>
</head>
<body>
        <form name=webpgm>
        <input type="button" value="push me" onclick="foropen()">
</body>
</html>
```

Alert boxes

An alert dialog box is mostly used to give a warning message to the users.

```
alert("message")
```

Where

alert –	function name
message –	to be displayed

Example 1:

```
<html>
<body>
        <button onclick="myFunction()">Try it</button>
        <script>
                function myFunction() {
                alert("Hello! I am an alert box!");
                }
        </script>
</body>
</html>
```

Example 2:

```
<html>
<head>
                <script type="text/javascript">
                <!--
                        function Warn() {
                        alert ("This is a warning message!");
                        document.write ("This is a warning message!");
                 }
                //-->
                </script>
</head>
<body>
<p>Click the following button to see the result: </p>
  <form>
<input type="button" value="Click Me" onclick="Warn();" />
</form>
</body>
</html>
```

Confirm boxes

A confirmation dialog box is mostly used to take user's consent on any option. If the user clicks on the OK button, the window method **confirm ()** will return true. If the user clicks on the Cancel button, then **confirm ()** returns false.

Syntax:

```
confirm(“message”)
```

Example

```
<html>
<head>
<button onclick="myFunction()">Try it</button>
        <script>
                function msg()
                {
                  var x;
                 if (confirm("Press a button!") == true)
                {
                  x = "You pressed OK!";
                   }
                else
                {
                   x = "You pressed Cancel!";
                 }
                   document,write( x);
                }
        </script>
</head>
<body>
        <input type=”button” onclick = “msg()” value=”Display a confirm box”>
</body>
</html>
```

Prompt boxes

The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.

This dialog box is displayed using a method called **prompt()**

Syntax

```
window.prompt("message","defaultText")
;
```

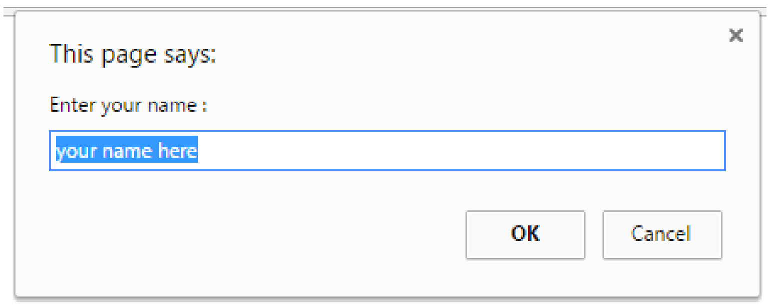
Where

window	-	object
prompt	-	function
default text	-	no need to give this value. So it can be left blank.
Message	-	optional. The message to display in the dialog box.

Example

```
<html>
<head>
        <script type="text/javascript">
        <!--
                function getValue(){
                var retVal = prompt("Enter your name : ", "your name here");
                document.write("You have entered : " + retVal);
        }
        //-->
        </script>
  </head>
  <body>
<form>
        <input type="button" value="Click Me" onclick="getValue();" />
</form>
</body>   </html>
```

Output



This page says:

Enter your name :

your name here

3.3 Java script with XML

Events

When the page loads, it is called an event. When the user clicks a button, that click too is an event.

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

Examples

i) mouse click

ii) pressing any key

iii) closing a window

iv) resizing window

v) image loading

vi) submitting form

The functions will be executed only when event occurs.

3.3.1 Event handlers

Event handlers are functions which are used to handle the events. There are two types of event handlers.

i) interactive event handler

ii) non interactive event handler

An interactive event handler that depends upon users interaction. On mouse click is an interactive event handler because, it depends on users action on the mouse.

Non interactive event handler is the one that will not depends on user action. onload is an non interactive event handler because, it executes automatically without the interaction of the user.

Important event handlers

i) onload

ii) onclick

iii) onfocus

iv) onunload

v)onblur

vi)onerror

vii)onmouseover

viii)onmouseout.

i) onload

The onload event occurs only when the document loaded.

```
onload=”function name”
```

where

onload – event handler

function name – function to handle to the event

Example

```
<html>
<head>
        <script>
                function myFunction()
                 {
                          alert("Page is loaded");
                }
        </script>
</head>
<body onload="myFunction()">
<h1>Hello World!</h1>
</body>
</html>
```

ii) onclick

The onclick event Occurs only on a mouse click. This is the most frequently used event type which occurs when a user clicks the left button of the mouse.

```
onclick=”function name”
```

Example

```
<html>
<head>
        <script type="text/javascript">
                function sayHello()
                {
                alert("Hello World")
                 }
        </script>
</head>
<body>

        <form>
        <input type="button" onclick="sayHello()" value="Say Hello" />
        </form>
</body>
</html>
```

iii) onunload

The onunload event occurs only when user exists a page.

```
onunload=”function name”
```

Example

```
<html>
<head>
        <script>
        function myFunction() {
          alert("welcome");
        }
        </script>
</head>
<body onunload="myFunction()">
</body>
</html>
```

iv) onfocus

The onfocus event occurs only when the window gets focus

```
onfocus=”function name”
```

Example

```
<html>
<body>
Enter your name: <input type="text" onfocus="focus(this)">
        <script>
        function focus(x) {
          x.style.background = "red";
        }
        </script>
</body>
</html>
```

Output

Enter your name:

When the input field gets focus, a function is triggered which changes the background-color(red).

v) onblur

The onblur event triggers only when the window loses focus.

```
onblur=”function name”
```

Example

```
<html>
<head>
        <script>
        function sample() {
          var x = document.getElementById("fname");
         x.value = x.value.toUpperCase();
        }
        </script>
</head>
<body>
        Enter your name: <input type="text" id="fname" onblur="sample()">
</body>
</html>
```

vi) onerror

The onerror eventTriggers when an error occurs.

```
Onerror=”function name”
```

Example

```
<html>
<head>
        <script>
        function imgError() {
          alert('The image could not be loaded.');
        }
        </script>
</head>
<body>
<img src="d:\butterfly.gif" onerror="imgError()">
</body>
</html>
```

3.3.2 Forms

Each form in the document creates a form object. Form fields do not necessarily have to appear in a <form> tag. Forms can be put anywhere in a page.

Form array

Form array is an array used to store the form objects. Using this array we can store the form objects. It is used to access the form elements for doing some validations on the page.

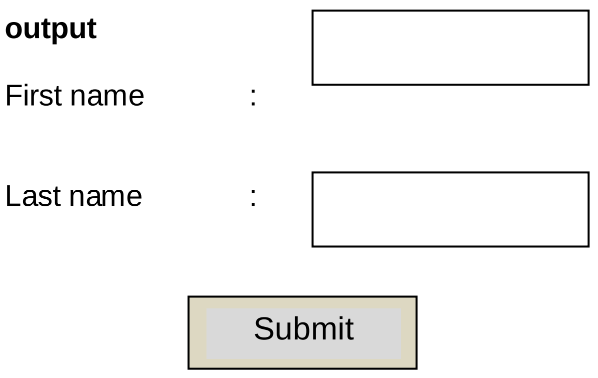
Form element properties

- ✓ action - browser to handle the form when it is submitted to a CGI program running on the server.
- ✓ method="GET" defines the method data is passed to the server when the form is submitted.
- ✓ input type="text" defines the text box object. This is standard HTML markup.
- ✓ id- sets or returns the id of a form
- ✓ target – set or returns where to open the action URL in a form.
- ✓ enctype- it is MIME type used to encode the content of a form

Example 1

```
<html>

<form action="C:\Documents and Settings\admin\Desktop\html\forms.html" method="get">

  First name: <input type="text" name="fname"><br>

  Last name: <input type="text" name="lname"><br>

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

</form>

</html>
```

output

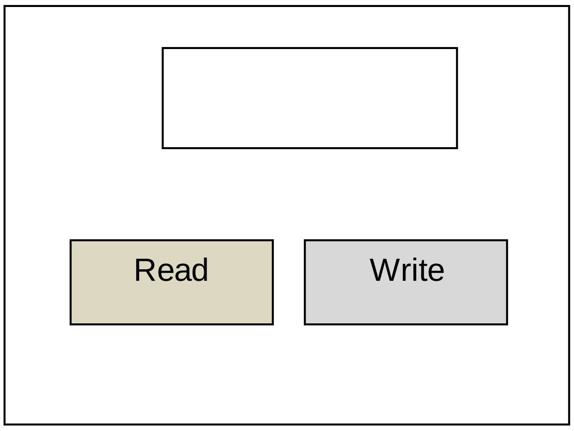


Example2

```
<HTML>

<HEAD>

<TITLE>Test Input </TITLE>

        <SCRIPT LANGUAGE="JavaScript">

        function readText (form) {

         TestVar =form.inputbox.value;

         alert ("You typed: " + TestVar);

        }

        function writeText (form) {

        form.inputbox.value = "Have a nice day!"

        }

        </SCRIPT>

</HEAD>

<BODY>

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

Enter something in the box: <BR>

<INPUT TYPE="text" NAME="inputbox" VALUE=""><P>

<INPUT TYPE="button" NAME="button1" Value="Read" onClick="readText(this.form)">

<INPUT TYPE="button" NAME="button2" Value="Write" onClick="writeText(this.form)">

</FORM>

</BODY>

</HTML>
```

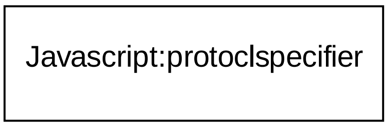
When you click the "Read" button, JavaScript calls the readText function, which reads and displays the value you entered into the text box.

When you click the "Write" button, JavaScript calls the writeText function, which writes "Have a nice day!" in the text box.



3.4 Using java script URLs

Javascript URLs are used to include javasrcipt code on the client side.



Where

Javascript - keyword

Protocol specifier – to specify the javascript code

Java script interpreter executes the codes present in the URL while execution.

Example

i) javascript: var now = new Date();

<h1> time is: </h1>"+now";

3.4.1 Image maps

The <map> tag is used to define a client-side image-map. An image-map is an image with clickable areas in the image, which has a link associated with it. There are two types of image maps. They are,

i) Client side image map

ii) server side image map

i) Client side image map

With a client side image map, specify a list of areas that will be used as the links. The information required to run the image map must be given in our html document. Since all the information regarding the map functions are provided on the client side, it’s called client side image map.

There are four types of these areas; rectangles, circles, polygons and default. Its down with the help of <MAP> tag

Syntax

```
<map name=”name of the map”>

<area shape=”name of the shape” cords=”x,y,w,h”
href=”name of the link” alt=”image”>

</map>
```

Where

Name	-	name of the map used
Cords	-	x,y co-ordinates to fix the map area, w- width of the map, h-height of map area.
Shape	-	shape to be defined. It can be rect, circle, poly
Href	-	url to be linked by this area
Alt	-	alternate message to display on the screen

Example

```
<html>
<body>
<img src="d:\planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap">
        <map name="planetmap">

             <area shape="rect" coords="0,0,82,126" alt="Sun" href="d:\sun.html">

             <area shape="circle" coords="90,58,3" alt="Mercury" href="d:\mercury.html">

            <area shape="circle" coords="124,58,8" alt="Venus" href="d:\venus.html">

        </map>
</body>
</html>
```

Server side image map

Server side image map is an image map run by common gateway interface. simply put the image inside a hyper link and use **ismap** attribute which makes it special image and when the user clicks some place within the image, the browser passes the coordinates of the mouse pointer along with the URL specified in the <a> tag to the web server.

The server uses the mouse-pointer coordinates to determine which document to deliver back to the browser.

```
<a href=”url address”>

<img src=”peacock.jpg” width=”200” height=”250”is map>

</a>
```

The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image, beginning with (0,0).

Example

```
<html>
<body>
        <a href="f/cse.jpg " target="_self">
        <img ismap src="d/images/logo.png" alt="Tutorials Point" border="0"/>
        </a>
</body>
</html>
```

3.4.2 status bar

The status bar sets the text in the status bar at the bottom of the browser window. It is used to display the any information on the status window.

Properties

a) status information about the browser window

b) default status

we try to open a particular web page, the status of the operation such as connecting host, opening page and other information will be displayed on the status bar.

Example

```
<script type="text/javascript">
        function statwords(message)
        {
        window.status = message;
        }
</script>
<center>
<form>
<input type="button" VALUE="Write a status bar message" OnClick="statwords('This is my
message');">
</form>
</center>
```

The javascript event **onclick** activates the function called **statwords** and passes a text string to a variable called a message. The function then activates the status bar area and insert the text string.

3.4.3 Cookies

Cookies are information , stored in a small files on our browser. It contains some data.

1. A name-value pair containing the actual data

2. An expiry date after which it is no longer valid

3. The domain and path of the server it should be sent to

When a user visit a web page, his username stored in a cookie. Next time user visit a web page , the cookie just remember his username

Cookies contains the following components.

i) name -Cookies are set and retrieved in the form of key-value pairs

ii) expires -The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.

iii) Domain − The domain name of your site.

iv) Path − The path to the directory or web page that set the cookie

v) Secure − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.

vi) Name=Value − Cookies are set and retrieved in the form of key-value pairs

Use of cookies

1. Web servers to perform functions such as tracking information about user visits to websites.

2. it is used to store information between web pages

3. The main use of the cookie is to identify the user and customize web page for them according to the information stored in the cookie

3.4.4 Live connect

Java applet can communicate with Java script in a same HTML page through a technology called Live Connect which is supported by all major web browsers. LiveConnect was first implemented in the Netscape browser, and currently Mozilla Firefox fully supports this feature.

There are two main aspects of LiveConnect**:**

- ✓ Calling Java methods from JavaScript
- ✓ Using JavaScript objects in Java

Java console

It is browser window that is used to display java messages. This is used mainly for simple debugging operations. Java Console is a simple debugging aid that redirects any System.out and System.err to the console window. It is available for applets running with Java Plug-in and applications running with Java Web Start.

The steps given below are followed to enable java console in internet explorer.

Internet Explorer

- ✓ Click Tools and then Internet Options
- ✓ Select the Security tab, and select the Custom Level button
- ✓ Scroll down to Scripting of Java applets
- ✓ Make sure the Enable radio button is checked
- ✓ Click OK to save your preference

Java script to java communication

Whenever you need to access a Java object, class, array, or package just use one of the following four LiveConnect objects.

- ✓ JavaObject – Used to access a Java object from JavaScript.
- ✓ JavaClass – Used as a reference to a Java class.
- ✓ JavaArray – Used to access Java arrays.
- ✓ JavaPackage – Used as a reference to a Java package.

Whenever your JavaScript code refers to a Java package, the JavaScript runtime automatically creates a JavaPackageobject.

In this case, we are creating an instance of class MyClass which is inside the mypackagepackage.

```
var myVar=new Packages.mypackage.MyClass();
```

Java to java script communication

To communicate from the java to javascript, import package netscape.javascript package in to the java file. It contains following classes.

i) netscape.javascript.Jsobject- *JSObject* allows Java to manipulate objects that are defined in JavaScript. Values passed from Java to JavaScript

ii) netscape.javascript.JSException- its allow java code to handle script errors.

To access the live connect classes, place the jar file on the jdk compiler class path.

SUMMARY

- ✓ Scripting languages are programming languages used to write web based application
- ✓ Life span of variable means how long a variable retains a given value during the execution of the program.
- ✓ Variables are of two types 1. Global 2. Local
- ✓ A global variable has global scope which means it can be defined anywhere in JavaScript code.
- ✓ Variables which are declared inside a function are called local variables.
- ✓ JavaScript variable has dynamic data types. This means that the same variable can be used to hold different data types
- ✓ Literals are fixed values that does not change during the script execution.
- ✓ An operator is a symbol which represents an operation that can be performed on data.
- ✓ Assignment operators are used to assign values to JavaScript variables
- ✓ Short hand assignment operators are operators which are used to simplify the coding of certain type of assignment statement.
- ✓ Comparison operators are used in logical statements to determine equality or difference between variables or values.
- ✓ Logical operators are used to determine the logic between variables or values.
- ✓ Arithmetic operators are used to do arithmetic calculations. There are two types i) binary operator ii) unary operator
- ✓ Control Structures are used to transfer java script program control from one statement to any other statement.
- ✓ Looping statements are the statements execute one or more statement repeatedly several number of times.
- ✓ If statement is used to execute or skip one statement or group of statements for a particular condition.

- ✓ If--else statement is used to execute one group of statements if the test condition is true or other group if condition is false.
- ✓ The switch statement can have a number of possible execution paths. This permits any number of branches.
- ✓ While loop repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body
- ✓ A do-while loop is similar to a while loop, except that a do-while loop is execute at least one time.
- ✓ For statement is used to execute a statement or a group of statements repeatedly for a known number of times.
- ✓ The for/in statement loops through the properties of an object.
- ✓ The break statement "jumps out" of a loop.
- ✓ The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.
- ✓ A JavaScript function is a block of code designed to perform a particular task.
- ✓ Return statement is used to return a value from the function.
- ✓ Every web page resides inside a browser window which can be considered as an object.
- ✓ An array object stores multiple values in a single variable
- ✓ The history object represents an array of URLs visited by the user.
- ✓ The location object is used to holds information on the current URL**.**
- ✓ The dialog boxes can be used to raise and alert, or to get confirmation on any input or to have a kind of input from the users.
- ✓ When the page loads, it is called an event
- ✓ Event handlers are functions which are used to handle the events.
- ✓ Each form in the document creates a form object.
- ✓ Form array is an array used to store the form objects
- ✓ Java script URLs are used to include java script code on the client side
- ✓ An image-map is an image with clickable areas in the image, we will be taken to the link associated with it.
- ✓ The status bar sets the text in the status bar at the bottom of the browser window.
- ✓ Cookies are information, stored in small files on our browser.
- ✓ Java applet can communicate with Java script in a same HTML page through a technology called Live Connect which is supported by all major web browsers.
- ✓ Java console is a browser window that is used to display java messages

Review Questions

Part - A

1. Define: Scripting language
2. Why need for scripting language?
3. Define: Life span of variable
4. List out the two types of variables
5. Define: Local variable
6. Define: Global variable
7. Define: Literals
8. List out the different types of operators
9. Give the general form of for in statement
10. What is the use of break and continue statement?
11. Define Document object
12. Give the general form to create array
13. What are the three ways of calling deferred scripts?
14. Define: predefined objects
15. Define location objects
16. Define: dialog box
17. Define: Events
18. What are the two types of event handlers?
19. What is the use of form array?
20. What is the use of java script URLs?
21. Define: image map
22. Define cookies
23. What are the uses of cookies?
24. What is live connect?
25. List out the different types of live connect objects

PART – B

1. Give the structure of java script program
2. Explain about variables
3. Explain the different types of literals.
4. Discuss about logical operators
5. Give the structure of nested if and explain
6. What is a function?

7. Discuss about return statement
8. Write different types of DOM properties.
9. What is the use of forward () method in history object?
10. Differentiate between interactive and non-interactive event handler.
11. Write about form element properties.
12. Differentiate between client side and server side image map.

PART – C

1. Explain about different types of operators.
2. Explain about different types of looping statements.
3. Write a function to add two numbers.
4. Explain about different types of Document object methods.
5. Explain the following: i) Array object ii) History object
6. Explain about different types of dialog boxes.
7. Discuss about different types of event handlers.
8. Explain different types of image maps.
9. Write about status bar
10. Discuss about Cookies
11. Explain the communication between java and java script.

UNIT-4

SERVER SIDE SCRIPTING (JSP)

Objectives :

At end the this unit, students can

- Explain uses of JSP
- Differentiate between client side scripting and server side scripting
- Differentiate between JSP and Java Script
- State the advantages and disadvantages of JSP
- Describe the client and server responsibilities
- Explain the JSP Architecture
- Describe the JSP Life Cycle
- Differentiate between JSP and Servlet
- State the types of JSP Servers
- Define and explain the types of Comments in JSP
- Define Directives and its types
- Explain scripting Element
- Describe the types of scripting Elements.
- Explain different types of implicit objects

4.1 Introductions to JSP

JSP technology is used to create dynamic web applications. JSP pages are easier to maintain than a Servlet. JSP enables us to write HTML pages containing tags, inside which we can include powerful Java programs. Jsp pages are platform independent.

JSP does not require additional files like, java class files, web.xml etc

Any change in the JSP code is handled by Web Container (Application server like tomcat), and doesn't require re-compilation.

Script

A Script is a set of instructions. For web pages they are instructions either to the web browser (client side scripting) or to the server(server side scripting)

Client Side scripting Language

The client is the system on which the Web browser is running. JavaScript is the main client-side scripting language for the Web. Client-side scripts are interpreted by the browser. The process with client-side scripting is:

- The user requests a Web page from the server
- The server finds the page and sends it to the user
- The page is displayed on the browser with any scripts running during or after display

Server side scripting Language

The server is where the Web page and other content lives. The server sends pages to the user/client on request. The process is:

- The user requests a Web page from the server
- The script in the page is interpreted by the server creating or changing the page content to suit the user and the occasion and/or passing data around
- The page in its final form is sent to the user and then cannot be changed using server-side scripting

Java script

JavaScript is a lightweight, interpreted programming language. It is designed for creating static web pages. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

JSP

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP, ASP and React's JSX, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

Servlet

A servlet is a Java program that runs on a Web server. It is similar to an applet, but is processed on the server rather than a client's machine. Servlets are often run when the user clicks a link, submits a form, or performs another type of action on a website

4.1.1 Client side versus server side scripting

Client Side Scripting

- Client side scripting is used when the user's browser already has all the code and the page is altered on the basis of the users input.
- Client side scripting is used to validate the data send by the client to the server.
- It is used to create cookies.

- Client side scripting cannot be used to connect to the databases on the web server.
- Client side scripting can't access the file system that resides at the web server.
- Client side scripting is possible to be blocked by the user.
- Examples of Client side scripting languages : Java script, VB script.

Server Side Scripting

- Server side scripting is used to create dynamic pages based a number of conditions when the users browser makes a request to the server.
- Server executes server-side scripts to send out a page but it does not execute client-side scripts.
- Server side scripting is used to connect to the databases that reside on the web server.
- Server side scripting can access the file system residing at the web server.
- Server side scripting can't be blocked by the user.
- Examples of Server side scripting languages : PHP, JSP, ASP, ASP.Net, Ruby, Perl

JSP versus java script

Sno.	JSP	Java script
1.	It is executed by the server which hosts the site	It is executed by the browser on our local machine.
2.	It is server side scripting which enables us to write code in java and do perform database interaction etc.	It is client side scripting language which is used to interact with the client at front end ie browser.
3.	JSP uses full java technology	Java script uses little java technology

Advantages of JSP

- Easy to maintain and code.
- High Performance and Scalability.
- JSP is built on Java technology, so it is platform independent.
- It is more convenient to write and modify the code
- JSP programming is easy to learn and easy to implement for Non-Java programmers also.
- JSP programming eliminates the repeated deployment problems
- JSP programming environment provides page compilation automatically.

Disadvantages of JSP

- Java knowledge is necessary to use JSP effectively.
- It is difficult to trace errors occurred in JSP pages.
- Database connectivity is not as easy as it should be.
  It needs servlet engine

4.1.2 Client and server responsibilities

Client

A client is a piece of computer hardware or software that accesses a service made available by a server.

Server

A server is a computer program that provides services to other computer programs (and their users) in the same or other computers.

Client responsibilities

1. Client initiates the request to the server for a resource.
2. After that request, it waits for the reply from the client.
3. It receives the reply form the server
4. It can request to more than one service at a time
5. It can interact with the end user using graphical user interface.

Server responsibilities

1. Server never initiates a request or any activity.
2. It replies only for the request from connected more than one clients.
3. It can install/uninstall application and transfer data to client systems remotely.

4.1.3 Installing and configuring Tomcat server

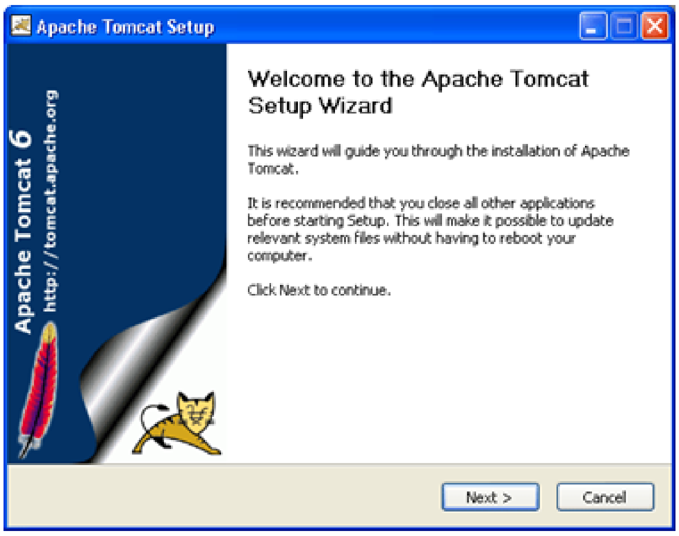
Step 1: Installation of JDK: Don't forget to install JDK on your system (if not installed) because any tomcat requires the Java 1.5 (Java 5) and Java 1.6 (Java 6) and then set the class path (environment variable) of JDK.

Step 2: Setting the class path variable for JDK

The other way of setting the class path variable is:

First right click on the MyComputer->properties->Advanced->EnvironmentVariables->Classpath.
Set bin directory path of JDK in the path variable.

Step 3: Here we are exploring the installation process by using the **.**exe file. The directory C:\apache-tomcat-6.0.10 is the common installation directory as it is pre-specified C:\ as the top-level directory. First unpack the zipped file and simply execute the **.**exe file.



Welcome to the Apache Tomcat Setup Wizard

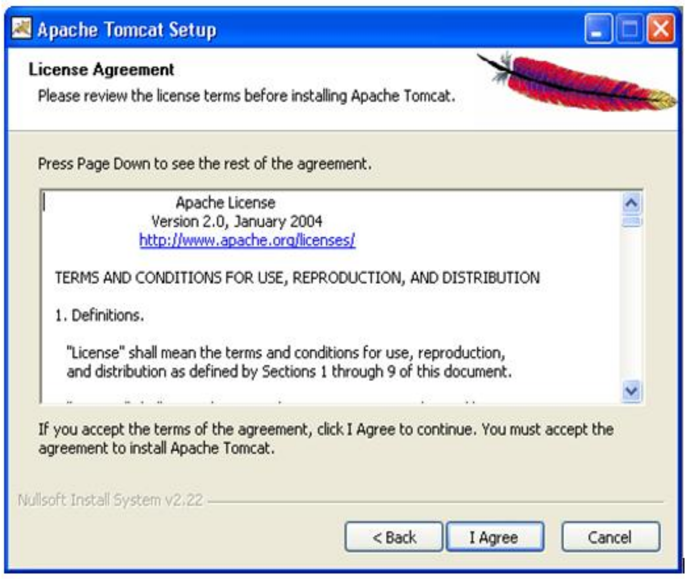
This wizard will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Fig: 4.1.1

The above shown screen shot is the first one shown in the installation process. Just click on the Next button to proceed the installation process.



License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

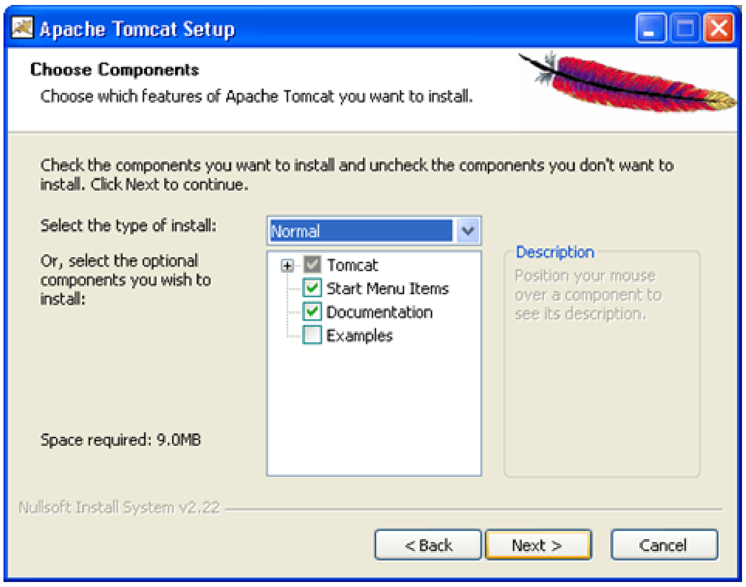
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Fig: 4.1.2

click "I Agree" button to continue the installation process.


Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Normal

Or, select the optional components you wish to install:

- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [ ] Examples

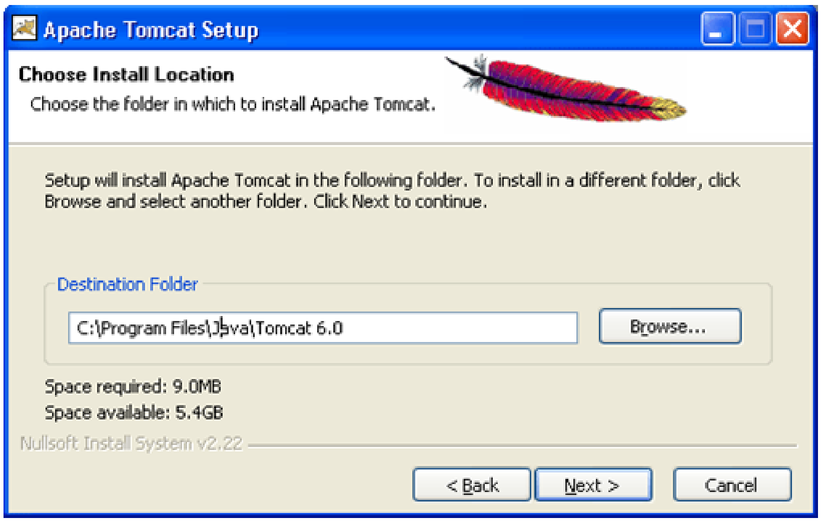
Description: Position your mouse over a component to see its description.

Space required: 9.0MB

Nullsoft Install System v2.22

Fig: 4.1.3

Click next to go with the default components choosen.


Apache Tomcat Setup

Choose Install Location

Choose the folder in which to install Apache Tomcat.

Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files\Java\Tomcat 6.0

Space required: 9.0MB
Space available: 5.4GB

Nullsoft Install System v2.22

Fig: 4.1.4

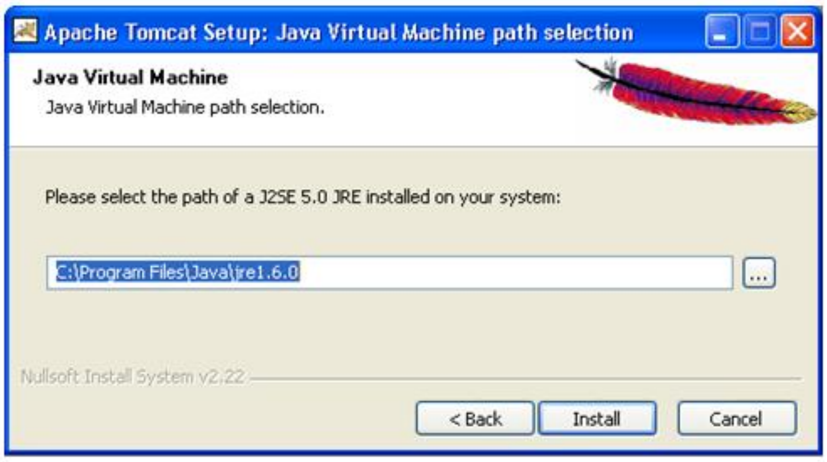
Now choose the port number on which you want to run the tomcat server. Tomcat uses the port number 8080 as its default value. Choose the user name and password as per your convenience.

e.g While using the port number 8080, give the following request in the address bar as:

Default Port: http//localhost:8080/index.jsp

Note that we do no need to specify any port number in the URL.

Now click on the Next button to proceed the installation process.



Apache Tomcat Setup: Java Virtual Machine path selection

Java Virtual Machine

Java Virtual Machine path selection.

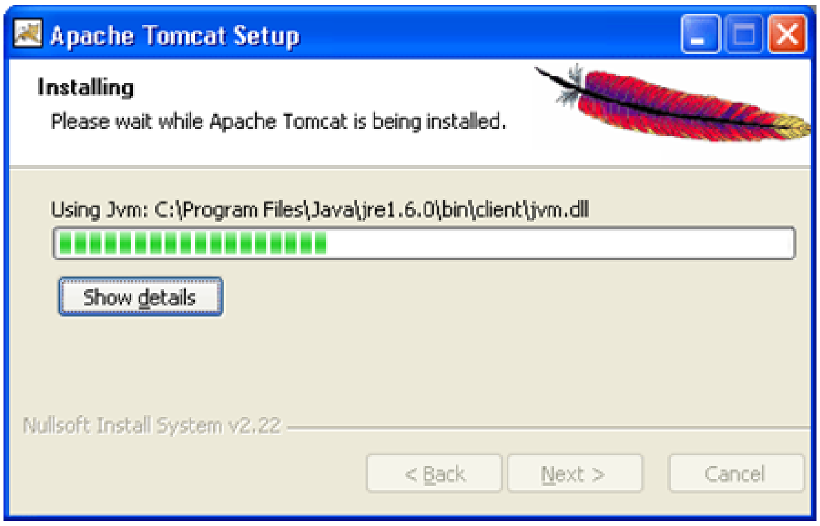
Please select the path of a J2SE 5.0 JRE installed on your system:

C:\Program Files\Java\jre1.6.0

Nullsoft Install System v2.22

Fig: 4.1.5

The installation process shows the above screen as the next window. This window asks for the location of the installed Java Virtual Machine. Browse the location of the JRE folder and click on the Install button. This will install the Apache tomcat at the specified location.



Apache Tomcat Setup

Installing

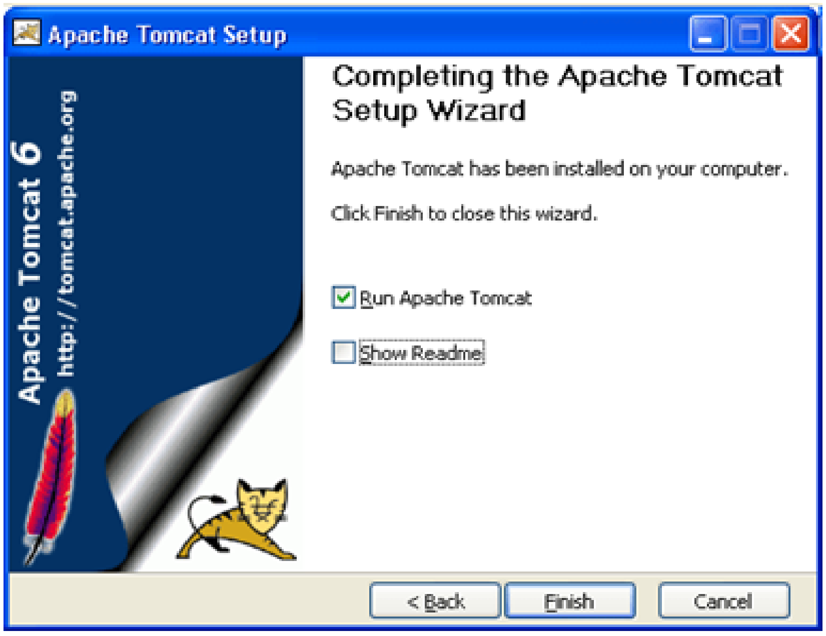
Please wait while Apache Tomcat is being installed.

Using Jvm: C:\Program Files\Java\jre1.6.0\bin\client\jvm.dll

Nullsoft Install System v2.22

Fig:4.1.6

To get the information about installer click on the "Show details" button.


Apache Tomcat Setup

Completing the Apache Tomcat Setup Wizard

Apache Tomcat has been installed on your computer.

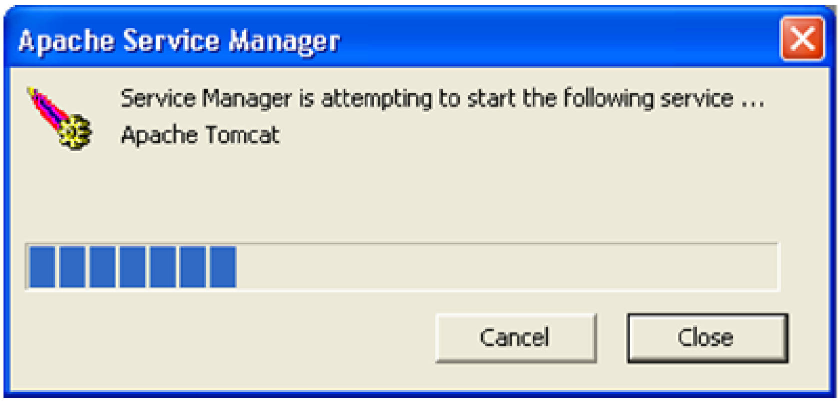
Click Finish to close this wizard.

[✓] Run Apache Tomcat

[ ] Show Readme

Fig: 4.1.7

After completion of installation process it will display the window like the above one.


Apache Service Manager

Service Manager is attempting to start the following service ...
Apache Tomcat

Fig:4.1.8

On clicking at Finish button, a window like the above one will display a message printed on the window given below.

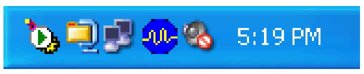
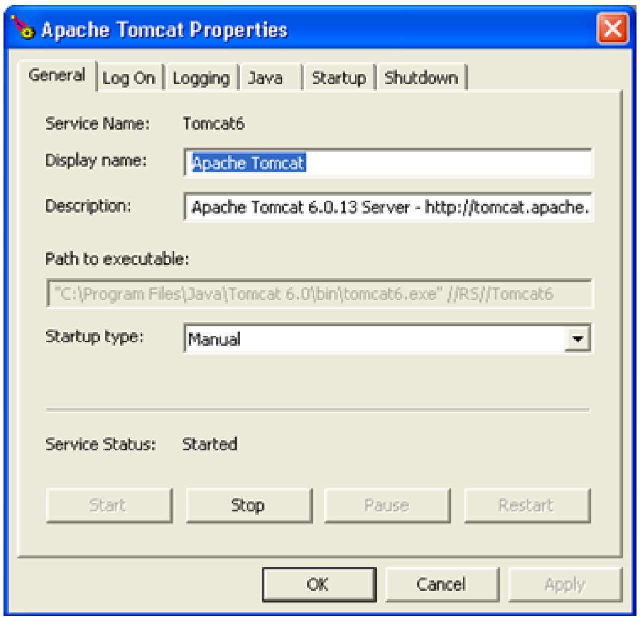


Fig:4.1.9

After successfully installing, a shortcut icon to start the tomcat server appears in the icon tray of the task bar as shown above. Double clicking the icon, displays the window of Apache Manager for Tomcat. It will show the "Startup type" as manual since we have changed the destination folder for tomcat during the installation process. Now we can configure the other options like "Display name" and "Description" .We can also start, stop and restart the service from here.



Service Name: Tomcat6

Display name: Apache Tomcat

Description: Apache Tomcat 6.0.13 Server - http://tomcat.apache.

Path to executable:

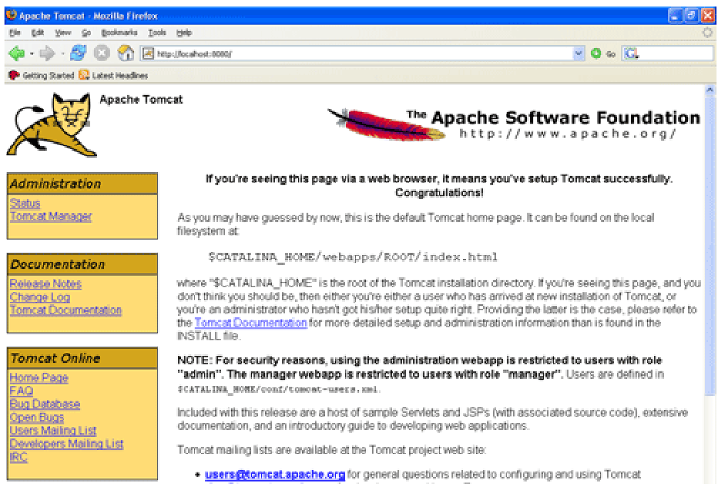
"C:\Program Files\Java\Tomcat 6.0\bin\tomcat6.exe" //RS//Tomcat6

Startup type: Manual

Service Status: Started

Fig: 4.1.10

If installation process completes successfully then a window as shown below will appear.



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat

Fig: 4.1.11

4.1.4 JSP Architecture

The web server needs a JSP engine ie. container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages.

A JSP container works with the Web server to provide the runtime environment and other services a JSP needs

A typical request / response phase of a JSP is defined below

a) Request is initiated for a jsp file by client using browser

b) Webs server (JSP engine) loads the JSP file and translate the JSP file into a java code . The Generated Java code will be a Servlet.

c) Once Servlet (Java code) is generated, JSP engine compiles the servlet. Any compilation errors will be detected in this phase.

d) Now servlet class is loaded by the container and executes it.

e) JSP Engine sends the response back to client.

Translation and compilation phase is done only when

a) First request came for the jsp file

b) The generated servlet is older than the JSP file. This is the case when JSP file is

modified

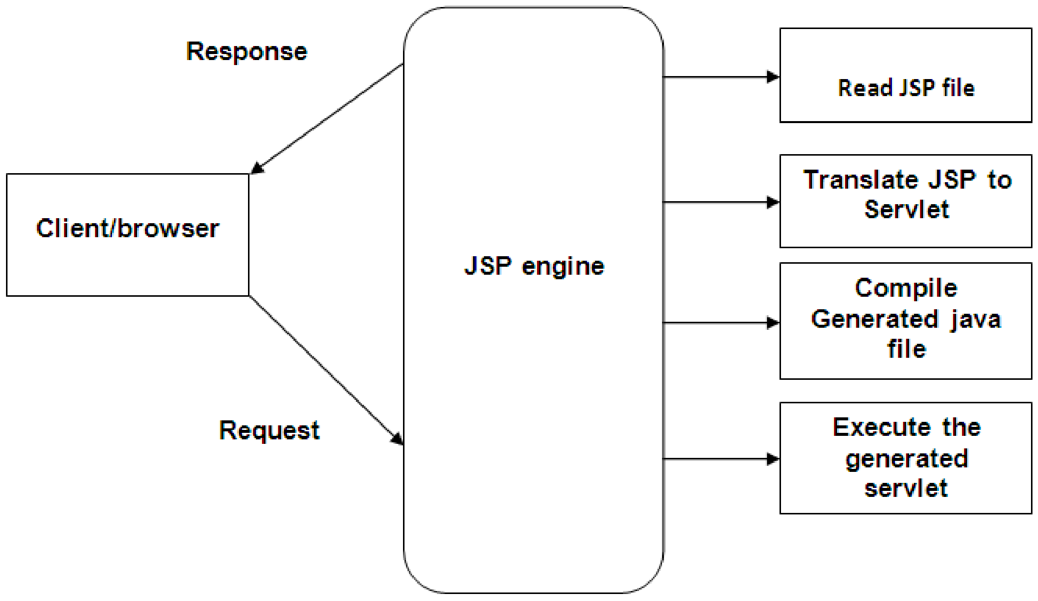


Fig: 4.1.12

4.1.5 Life Cycle of a JSP Page

The steps given below gives the life cycle of a JSP page.

The client requests t he web server which contains different types of web pages such as SAP,JSP etc.

Web server recognizes the request. If the request is for JSP page, it is directed to the servlet engine.

Servlet engine directs it to the JSP engine.

JSP engine process the JSP pages and sends back the response to web server through servlet.

The web server responses the client with the needed page.

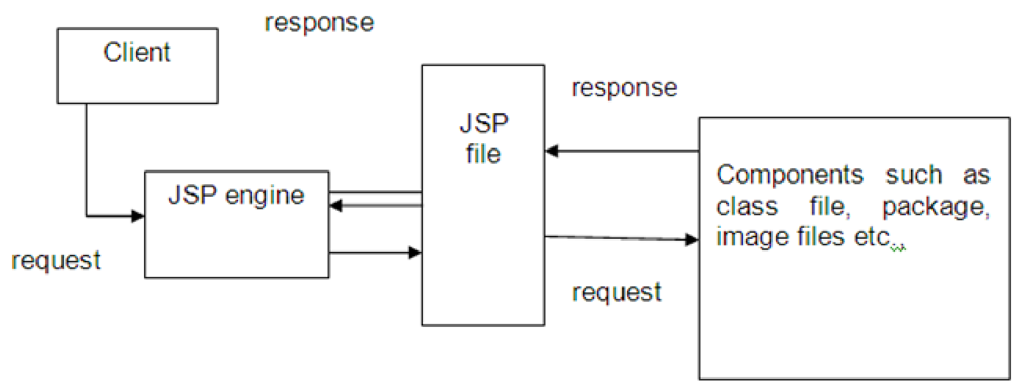


Fig: 4.1.13

4.1.6 JSP vs Servlets

JSP	Servlets
JSP is a webpage scripting language that can generate dynamic content.	Servlets are Java programs that are already compiled which also creates dynamic web content.
JSP run slower compared to Servlet as it takes compilation time to convert into Java Servlets.	Servlets run faster compared to JSP.
The advantage of JSP programming over servlets is that we can build custom tags which can directly call Java beans.	There is no such custom tag facility in servlets.
JSP is document centric	Servlet is like programs.

4.1.7 JSP vs Asp.NET

ASP.NET is an open-source server-side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web applications.

- Both are designed to create interactive (GUI based) web pages for web based applications.
- JSP and ASP support application development and deployment faster and easier.
- JSP and ASP are both server side scripting languages
- JSP is from Sun Microsystems while ASP is from Microsoft

- ASP costs money while JSP is free.
- ASP code is interpreted while JSP code is compiled at run time
- JSP code can run faster than ASP if there are fewer changes
- Majority of Windows users use ASP while users of open source operating systems like Linux use JSP among others.
- Jsp is secured and Asp is not secured
- JSP support only java and supports VB Script, Jscript

List of JSP servers

The most commonly used JSP servers are

- Apache tomcat
- Web logic
- J2EE server
- Apache HTTP server
- Netscape Server
- IIS Server

4.2 JSP Elements

4.2.1 Comments

- Comments are explanations included in a JSP program for better understanding. These will not be executed.
- In JSP two types of comments are allowed in the JSP page. They are
  - HTML Comments
  - Hidden Comments
- `<!-- comment -->` is not a JSP comment. This is HTML comment.

1. HTML Comments

HTML comments are explanation included in a HTML part of the JSP page. These comments will not be displayed on the page. But this can be viewed in the source.

Syntax

```
<!... valid comments….>
(or)
<!... valid comments….>
[<%=expression%>]
```

Where

<! - Ope

>   - Closi

<% - Scriplet expression Opening tag

%> - Scriplet expression Closing tag

[<%=expression%>] is optional one

Example

- <%-- This is a JSP comment--%>
- <!—This is a HMTL comment-->

2. Hidden Comments

A comment that documents the JSP page but is not sent to the client. The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags. A hidden comment is not sent to the client. The hidden comment is useful when which want to hide or "comment out" part of our JSP page.

It use any characters in the body of the comment except the closing --%> combination.

Syntax

<%-- comment --%>

Examples

```
<%@ page language="java" %>
<html>
<head><title>A Hidden Comment </title></head>
        <body>
              <%-- This comment will not be visible to the client in the page source --%>
        </body>
</html>
```

4.2.2 Directives

Directives are prewritten files in JSP. The **jsp directives** are a message that tells the web container how to translate a JSP page into the corresponding servlet.

There are **three types** of directives:

1. Page directive
2. Include directive
3. Taglib directive

- The directives must begin with <%@ and end with%>.
- Directives are used to design the entire JSP page.

Syntax

```
<%@ directive attribute="value" %>
```

1. Page directive.

The page directive defines attributes that apply to an entire JSP page.

```
<%@ page attribute1="value1"
attribute2="value2”,  attributen="value n" %>
```

Attributes of JSP page directive

The following are the attributes of the page directives.

- import
- extends
- info
- buffer
- language
- isThreadSafe
- autoFlush
- session
- errorPage
- isErrorPage

1) import

The import attribute is used to import class, interface or all the members of a package. It is similar to import keyword in java class or interface.

2) extends

The extends attribute defines the parent class that will be inherited by the generated servlet. It is rarely used.

3) info

Info attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

4) buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page. The default size of the buffer is 8Kb.

5) language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

6) isThreadSafe

Servlet and JSP both are multithreaded. If we want to control this behavior of JSP page, we can use isThreadSafe attribute of page directive. The value of isThreadSafe value is true. If make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.

7) errorPage

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

8) isErrorPage

The isErrorPage attribute is used to declare that the current page is the error page.

9) autoflush

The default value is true. To decide whether to clear the output buffer or not. If it is true, the buffer will be cleaned automatically else an exception will be thrown when the buffer overflows.

10) Session

The default value is true. It is used to extend the session of the JSP page to the bean application used inside this page.

Example

```
<%@ page import="java.util.Date" %>

         Today is: <%= new Date() %>

<%@ page info="welcome " %>

<%@ page buffer="16kb" %>

<%@ page errorPage="myerrorpage.jsp" %>
```

2. Include Directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time

syntax

```
<%@ include file="relativeURL" %>
```

No blank space is allowed between % and @ symbol.

Where

- include file – Keywords
- relative URL – name of the web oriented application

This directive has only one attribute name file. In this the name of the web application to be loaded in the JSP page is given. The web application file should not contain <body>, </body>tags. This is because these tags get confused with same tags in JSP page.

Example

```
<%@ page into =”JSP example”%>
<html>
      <body>
            <%@ include file="header.html" %>
              --------------
               --------------
      </body>
</html>
```

The file header.html will be included in the jsb file.

4. Taglib Directive

The JSP taglib directive is used to define a tag library that defines many tags. We use the Tag Library Descriptor file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

Syntax

```
<%@ taglib uri="uriofthetaglibrary" prefix="java class filename" %>
```

4.2.3 Scripting Elements

Scripting elements are used to embed jsp codes directly into HTML page. The following are the different scripting elements. They are

- Declarations
- Scriplets
- Expression

1. Declarations

Declarations are used to define variables or methods in the jsp file.

Syntax

```
<%! Variable or method declaration %>
```

where

<%! - Declaration Opening tag

%> - Declaration Closing tag

Declarations are used with java code in the jsp file. That is, it is used with JSP scriptlets or expressions.

Example

```
<%!
        String makeItLower(String data)
        {
              returndata.toLowerCase();
        }
 %>
```

2.Scriptlets

In JSP, java code can be written inside the jsp page using the scriptlet tag.

Syntax

```
<%  java source code %>
```

The scriplet contains any number of valid java language statements only. Any text, HTML tags or JSP elements must be outside the scriplet.

Example

```
<html>

<body>

<% System.out.print("welcome to jsp"); %>

</body>

</html>
```

3. Expressions

Expressions are used to dynamically calculate values in the JSP page.

Syntax

```
<%=  statement %>
```

Where

<% - expression opening tag

%> - expression closing tag

The value of the expression is calculated and is automatically converted to string. After conversion it is written to the out object. The expression must be a valid java expression and should not terminate with a semicolon.

If the expression has more than one part, it is evaluated from left to right.

Example

```
<html>

<body>

Current Time: <%= java.util.Calendar.getInstance().getTime() %>

</body>

</html>
```

4.2.4 Simple JSP page

The steps given below gives the procedure followed to create and run a JSP page

1. First Install java
2. Edit the jsp program using any editor(notepad)
3. Store the edited program using .jsp extension in the tomcat server directory..ie

```
C:\program\ApacheSoftwareFoundation\tomcat5.0\webapps\
Root\user directory\.jsp files
```

4. Open the internet explorer or mozila firebox web browser. In the address bar give http://localhost:8080/. All the folders in Root directory will be listed.
5. Select the user directory where the .jsp files are stored. All the .jsp files will be listed.
6. Select the needed .jsp file to execute. The selected file will be executed.

Example

```
<html>
<body>
        <%
          int a=5, b=10, c=15, avg;

          avg=(a+b+c)/z;
           out.println("Average:"+avg);
           %>
</body>
</html>
```

4.3 Implicit Objects

Implicit objects created by the web container. These objects are used to create JSP pages. The users need not to declare these objects, but they are provided by the container in the implementation class.

All the implicit objects are available only to scriplets and expressions only. The following are the implicit objects.

- ✓ Request Object
- ✓ Response Object
- ✓ Page context Object
- ✓ Session Object
- ✓ Application Object
- ✓ Out Object
- ✓ Page Object
- ✓ Config Object
- ✓ Exception Object

1.Request Object

The request object is an instance of a javax.servlet.http.HttpServletRequest object. Each time a client requests a page the JSP engine creates a new object to represent that request.

The request object provides methods to get HTTP header information including form data, cookies, HTTP methods etc.

Method	Description
getRequestURL()	This methods returns the full URL of the client which request the JSP page
getCharacterEncoding()	This method returns the character set in which the page is encoded.
getServerName()	This method returns the name of the computer on which the server is running.
getServerPort()	This method returns the port number used for communication with the server
getRemoteHost()	This method returns the name of the computer which requests the server
getQueryString()	This method returns the query portion of the URL after the question mark.
getMethod()	This method returns the method used for the request.
getRemoteUser()	This method returns the name of the logged user.

2.Response Object

The response object is an instance of a javax.servlet.http.HttpServletResponse object. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes etc.

syntax

```
public void_jspService(HttpServletRequest request),

HttpServletRepsonse response) throws

IOException,Servlet Exception
```

3. Page context Object

This object details about the page context for a particular page. This object has only scope and is instance of **javax.servlet.PageContext.**

syntax

```
pageContext=_jspFactory.getPageContext(this,request
,response, “errorpage.jsp”,true,8192,true)
```

4. Session object

A session is defined as a series of related interaction between a single client and the server. A session object is created on the server. It is an unique identifier and is called **session id**. This id is given to the client for communicating with the server. If the user is accessing a site having more than one JSP or servlet pages, the same session id is shared among all the page.

Method	**Description**
long getCreationTime	This method returns the time when the session was created
string getId()	This method returns the unique identifier assigned to the session.
long getLastAccessedTime()	This method returns the time when the user sent a last request associated with this session.
Void setMaxInactiveInterval(int interval)	This method is used to give the maximum length of time in seconds that the servlet engine keeps the session open if no user requests have been made.
void setMaxInactiveInterval()	This method returns the maximum time interval in seconds that the servlet engine keeps this session open between client requests.
boolean isNew(0	This method returns true if a new session was created, but client has not joined.

5. Application Object

JSP application contains more than one web pages. For each web page an application object is created to allocate memory during execution. This object is an instance of **Javax.servlet.htp.HttpSession**

6. Out Object

The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object and is used to send content in a response.

Method	Description
out.print(dataType dt)	This method is used to print a data type value.
out.println(dataType dt)	This method is used to Print a data type value then terminate the line with new line character.
out.flush()	This method is used to Flush the stream.

7. Page Object

This object is an actual reference to the instance of the page. It contain the details of the current page. It can be thought of as an object that represents the entire JSP page.

8. Config Object

The config object is an instantiation of **javax.servlet.ServletConfig** and is a direct wrapper around the ServletConfig object for the generated servlet. Config Implicit object is used for getting configuration information for a particular JSP page.

9. Exception Object

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

4.3.1 Scope

Application Scope

- A JSP object created using the ‘application’ scope can be accessed from any pages across the application.
- The JSP object is bound to the application object. Implicit object application has the ‘application’ scope.

Session Scope

- ‘session’ scope means, the JSP object is accessible from pages that belong to the same session from where it was created.
- The JSP object that is created using the session scope is bound to the session object. Implicit object session has the ‘session’ scope.

Request

- A JSP object created using the ‘request’ scope can be accessed from any pages that serves that request. More than one page can serve a single request.
- The JSP object will be bound to the request object. Implicit object request has the ‘request’ scope.

Summary

- ✓ JSP technology is used to create dynamic web applications.
- ✓ JSP pages are easier to maintain then a Servlet.
- ✓ JSP does not require additional files like, java class files, web.xml etc
- ✓ Any change in the JSP code is handled by Web Container (Application server like tomcat), and doesn't require re-compilation.
- ✓ Client side scripting is used to validate the data send by the client to the server.
- ✓ Server side scripting is used to create dynamic pages based a number of conditions when the users browser makes a request to the server.
- ✓ Server side scripting can’t be blocked by the user.
- ✓ Client initiates the request to the server for a resource.
- ✓ Server never initiates a request or any activity.
- ✓ Comments are explanations include in a JSP program for better understanding. These will not be executed.
- ✓ In Jsp two types of comments are allowed in the Jsp page. They are i) HTML comments ii) Hidden Comments
- ✓ HTML comments are explanation included in a html part of the JSP page. These comments will not be displayed on the page.
- ✓ The JSP engine ignores a hidden comment, and does not process any code within hidden comment tags
- ✓ Directives are prewritten files in JSP.
- ✓ The jsp directives are a message that tells the web container how to translate a JSP page into the corresponding servlet.
- ✓ The page directive defines attributes that apply to an entire JSP page.

- ✓ The import attribute is used to import class, interface or all the members of a package.
- ✓ The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used
- ✓ Info attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface
- ✓ The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.
- ✓ The language attribute specifies the scripting language used in the JSP page
- ✓ Servlet and JSP both are multithreaded.
- ✓ The errorPage attribute is used to define the error page
- ✓ The isErrorPage attribute is used to declare that the current page is the error page
- ✓ The autoflush attribute is used to clear the output buffer.
- ✓ The Session attribute is used to extend the session of the JSP page to the bean application used inside this page
- ✓ The include directive is used to include the contents of any resource it may be jsp file, html file or text file.
- ✓ The JSP taglib directive is used to define a tag library that defines many tags.
- ✓ Scripting elements are used to embed jsp codes directly into HTML page.
- ✓ Declarations are used to define variables or methods in the jsp file
- ✓ In JSP, java code can be written inside the jsp page using the scriptlet tag.
- ✓ Expressions are used to dynamically calculate values in the JSP page.
- ✓ Implicit objects created by the web container
- ✓ Implicit objects are used to create JSP pages.
- ✓ The request object passes the client request to the JSP page for processing.
- ✓ The response object sends back the response generated by the server back to the client
- ✓ The page context object contains details about the page context for a particular page.
- ✓ A session object is defined as a series of related interaction between a single client and server.
- ✓ A session object is created on the server.
- ✓ JSP application contains more than one pages.

- ✓ Page object is an actual reference to the instance of the page.
- ✓ Config Implicit object is used for getting configuration information for a particular JSP page.
- ✓ Exception Object is used to generate an appropriate response to the error condition.

Review Questions

PART – A

Define Client side and Server side scripting

Give the advantages of JSP

List out the types of JSP Servers.

Define: Comments

What are the two types of Comments?

Define: Directives

What are implicit objects?

List out the three types of implicit objects.

Define: Session Object

Define: Exception Object

PART – B

1. Give the difference between JSP and java script
2. What are the responsibilities of Client and server?
3. Write about HTML Comments.
4. What are the attributes of JSP page directives?
5. Define: scripting elements and list out different types of scripting elements.
6. Write about the methods of Session object.

PART – C

Explain the steps to installing JSP servers.

Explain the JSP architecture with a neat diagram

Discuss about the life cycle of a JSP page.

Explain the steps to create a JSP page.

List and explain the different types of directives

Discuss about different types of scripting elements

Explain the following implicit objects a)request b) Session c) application d)page

----

UNIT-5

JSP PROGRAMS & DATABASE ACCESS

Objectives :

At end the this unit, students can

- Define mysql
- Define jdbcodbc driver
- Create connection and statement in mysql
- Define executeUpdate() method
- Define executeQuery() method
- Write simple projects

5.0 INTRODUCTION

JSP technology is used to create dynamic web applications. JSP pages are easier to maintain than a Servlet. JSP pages are opposite of Servlets as a servlet adds HTML code inside Java code, while JSP adds Java code inside HTML using JSP tags. Everything a Servlet can do, a JSP page can also do it.

5.1 WRITING SIMPLE JSP PROGRAMS

5.1.1 Write a simple JSP program to convert entered text into uppercase

```
<html>

<body>

<%

String s=”web programming”;

String s1=s.toUpperCase();

Out.println(“converted uppercase text is :”+s1);

%>

</body>

</html>
```

Output:

```
Converted uppercase text is:  WEB PROGRAMMNG
```

5.1.2 Write a JSP program to find the maximum of three numbers

```
<html>
```

```
<body>
<%
int a=40,b=27,c=80;
if(a>b)
{
if(a>c)
out.println("BIG="+a);
else
out.println("BIG="+c);
}
else
{
if(b>C)
out.println("BIG="+b);
else
out.println("BIG="+c);
}
%>
</body>
</html>
```

Output:

```
        BIG=80
```

5.1.3 Write a program to add two numbers

```
<html>
<body>
<%
int a=20,b=30,c;
c=a+b;
out.println("the sum is:"+c);
%>
</body>
```

</html>

Output:

```
the sum is:50
```

5.2 MYSQL

MySQL is the most popular Open Source Relational SQL database management system. MySQL is one of the best RDBMS being used for developing web-based software applications.

The MySQL database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used in more than 6 million installations ranging from large corporations to specialized embedded applications on every continent in the world.

5.2.1 Creating MySQL Tables

The CREATE TABLE statement is used to create a table in MySQL.

The table creation command requires:

- Name of the table
- Names of fields
- Definitions for each field

Syntax:

```
Create table table_name (column1 datatype, column2 datatype, column3 datatype,…);
```

The following example creates a table called "student" that contains four columns: name, regno, dept and year:

Example

Create table student(name char(15), trgno number(5), dept char(25), year number(4));

5.2.2 Creating Records

The INSERT INTO statement is used to create new records in a table.

It is possible to write the INSERT INTO statement in two ways.

First method

The first way specifies both the column names and the values to be inserted. In this method, values for all columns are need not to be given.

Syntax:

```
INSERT INTO tablename (column1, column2, column3) VALUES (value1, value2, value3);
```

Example:

INSERT INTO student (name, regno, dept) VALUES ('raja', 1542, 'EEE');

Here, out of four columns, only 3 values are given.

Second method:

In the second method, column names are not mentioned, but values for all columns should be given.

Syntax:

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

Example:

INSERT INTO student VALUES ('raja', 1542, 'EEE', 2016);

Here, values for all columns are given.

5.2.3 Data Source

5.2.3.1<sql:setDataSource> Tag

The <sql:setDataSource> creates a datasource to connect to a database. A datasource creates a way to communicate with the database server. We use datasource to query, insert, update or delete information from the databases.

Syntax

```
<sql:setDataSource
  var="<string>"
  scope="<string>"
  dataSource="<string>"
  driver="<string>"
  url="<string>"
  user="<string>"
  password="<string>"/>
```

JSP <sql:setDataSource> Tag has following attributes:

1. **driver Attribute:** Specifies the driver to connect database. For each database there is a separate driver.

2. **url Attribute:** Specifies the location of the database.

3. **var Attribute:** Specifies the variable which holds the dataSource once it is created.

4. **user Attribute:** Specifies the user you already created to access database.

5. **password Attribute:** Specifies the password for the user you already assigned to access database.

5.2.4 JDBC-ODBC Bridge

JDBC drivers are used to open database connections and to interact with it by sending SQL or database commands then receiving results with Java. JDBC Bridge is used to access ODBC drivers installed on each client machine. ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database.

At the beginning, most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available. The following figure (Fig 5.1) explain the architecture of JDBC-ODBC Bridge

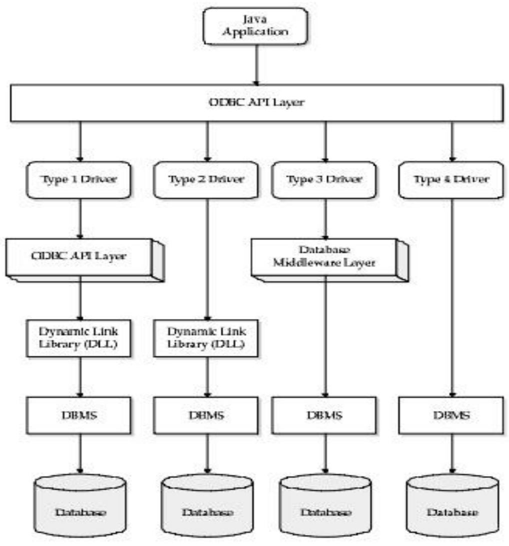


Fig 5.1

5.2.5 Creating connection

Database can be accessed only after creating connection object. Before creating connection a driver must be loaded and registered.

The general function used to create connection is

```
getConnection()
```

DriverManager provides three methods for creating connections:

- getConnection(String url)
- getConnection(String url, String userID, String password)
- getConnection(String url, Properties prop)

The driver manager maintains a list of registered drivers. When its getConnection() method is invoked, it checks each driver whether it will accept the specified URL. The driver manager do this by calling the driver's connect() method, which returns null if the driver cannot accept the URL or an active Connection object if it accept the URL.

5.2.6 Creating statement

The SQL language consists of statements that create, manipulate, and extract data from a relational database. JDBC provides an object-oriented representation of these SQL statements. This representation is called the java.sql.Statement interface. Statement objects send SQL commands to a database, which can be any of the following types:

- A data definition command such as CREATE TABLE or CREATE INDEX
- A data manipulation command such as INSERT or UPDATE
- A SELECT statement for performing a query

Data manipulation commands return a count of the number of rows modified, whereas a SELECT statement returns a set of rows known as a result set.

The Statement interface has two specialized sub interfaces:

- Prepared Statement - It uses precompiles SQL
- Callable Statement -It invokes stored procedures.

5.2.7 Statement

The basic interface for creating a statement is java.sql.Statement. It is obtained from the connection object with Connection.createStatement().

Example:

```
Connection con = null;

try {

con = DriverManager.getConnection(URL);

Statement stmt = con.createStatement();

...

stmt.close();

}

finally {

if (con != null)
con.close();

}
```

Once a statement is created, it can be used to execute commands. The following four methods are used to execute commands

- executeUpdate
- executeQuery
- execute
- executeBatch.

5.2.8 executeUpdate() method

executeUpdate() method can be used with the SQL INSERT, UPDATE, or DELETE statements, or with data definition statements such as CREATE TABLE.

Example

The program to count the number of rows updated is as follows.

```
import java.sql.*;

public class StatementUpdateExample {

public static void main(String... arg) {

Connection con = null;

Statement stmt = null;

try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver ");

// getting connection

con = DriverManager.getConnection ("jdbc:odbc:MyDataSource","user","pass");
System.out.println("Connection established successfully!");

stmt = con.createStatement();

//execute update query

int numberOfRowsUpdated = stmt.executeUpdate("UPDATE EMPLOYEE "

+ "set NAME='ravi' where ID=2 ");

System.out.println("numberOfRowsUpdated=" + numberOfRowsUpdated);

} catch (ClassNotFoundException e) {
```

```
e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

finally{

try {

if(stmt!=null) stmt.close(); //close Statement

if(con!=null) con.close(); // close connection

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}
```

After executed, the program prints “1 rows updated”.

5.2.9 executeQuery() method

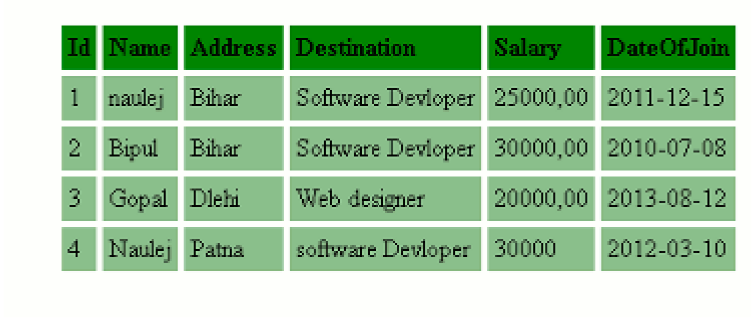
executeQuery() method is used to execute an SQL SELECT statement and to return a result set.

Example:

```
<%@page import="java.sql.DriverManager"%>

<%@page import="java.sql.ResultSet"%>

<%@page import="java.sql.Statement"%>

<%@page import="java.sql.Connection"%>

<%

String driverName = "com.mysql.jdbc.Driver";

String connectionUrl = "jdbc:mysql://localhost:3306/";

String dbName = "naulej";

String userId = "root";

String password = "root";
```

```
try {

Class.forName(driverName);

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

Connection connection = null;

Statement statement = null;

ResultSet resultSet = null;

%>

<h2 align="center"><font color="#FF00FF"><strong>Select query in
JSP</strong></font></h2>

<table align="center" cellpadding="4" cellspacing="4">

<tr>

</tr>

<tr bgcolor="#008000">

<td><b>Id</b></td>

<td><b>Name</b></td>

<td><b>Address</b></td>

<td><b>Destination </b></td>

<td><b>Salary</b></td>

<td><b>DateOfJoin</b></td>

</tr>

<%

try {

connection = DriverManager.getConnection(

connectionUrl + dbName, userId, password);

statement = connection.createStatement();

String sql = "SELECT * FROM employeedetails";

resultSet = statement.executeQuery(sql);
while (resultSet.next()) {
%>
<tr bgcolor="#8FBC8F">
<td><%=resultSet.getString("id")%></td>
<td><%=resultSet.getString("name")%></td>
<td><%=resultSet.getString("address")%></td>
<td><%=resultSet.getString("destination")%></td>
<td><%=resultSet.getString("salary")%></td>
<td><%=resultSet.getString("dateOfJoin")%></td>
</tr>
<%
}
} catch (Exception e) {
e.printStackTrace();
}
%>
</table>
```

Output



Id	Name	Address	Destination	Salary	DateOfJoin
1	naulej	Bihar	Software Devloper	25000,00	2011-12-15
2	Bipul	Bihar	Software Devloper	30000,00	2010-07-08
3	Gopal	Dlehi	Web designer	20000,00	2013-08-12
4	Naulej	Patna	software Devloper	30000	2012-03-10

5.2.10 Select Operation

Select operation is used to select record in the database. SELECT statement is used to do this operation

Example:

```
<%@ page import="java.io.* ,java.util.* ,java.sql.* "%>
<%@ page import="javax.servlet.http.* ,javax.servlet.* " %>
<%@ taglib uri="http://java.sun.com /jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com /jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>SELECT Operation</title>
</head>
<body>
<sql:setDataSource var="snapshot" driver="com .mysql.jdbc.Driver"
url="jdbc:mysql://localhost/TEST"
user="root" password="pass123"/>
<sql:query dataSource="$ {snapshot}" var="result">
SELECT * from Employees;
</sql:query>
<table border="1" width="100%">
<tr>
<th>Em p ID</th>
<th>First Nam e</th>
<th>Last Nam e</th>
<th>Age</th>
</tr>
<c:forEach var="row" item s="$ {result.rows}">
<tr>
<td><c:out value="$ {row.id}"/></td>
<td><c:out value="$ {row.first}"/></td>
<td><c:out value="$ {row.last}"/></td>
<td><c:out value="$ {row.age}"/></td>
</tr>
</c:forEach>
```

```
</table>

</body>

</html>
```

The Output is

Emp ID	First Name	Last Name	Age
100	Ram	Kumar	18
102	John	Peter	25
103	Abdullah	Khan	30

5.2.11 Insert Operation

Insert operation is used to insert a new record in the table. INSERT statement is used to do this operation

Example:

```
<%@ page import="java.io.* ,java.util.* ,java.sql.* "%>

<%@ page import="javax.servlet.http.* ,javax.servlet.* " %>

<%@ taglib uri="http://java.sun.com /jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com /jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JINSERT Operation</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<sql:update dataSource="$ {snapshot}" var="result">

INSERT INTO Employees VALUES (104, 2, 'Gowtham', 'Bhudha');

</sql:update>
```

```
<sql:query dataSource="$ {snapshot}" var="result">
SELECT * from Employees;
</sql:query>
<table border="1" width="100%">
<tr>
<th>Em p ID</th>
<th>First Nam e</th>
<th>Last Nam e</th>
<th>Age</th>
</tr>
<c:forEach var="row" items="$ {result.rows}">
<tr>
<td><c:out value="$ {row.id}"/></td>
<td><c:out value="$ {row.first}"/></td>
<td><c:out value="$ {row.last}"/></td>
<td><c:out value="$ {row.age}"/></td>
</tr>
</c:forEach>
</table>
</body>
</html>
```

The Output is:

Emp ID	First Name	Last Name	Age
100	Ram	Kumar	18
102	John	Peter	25
103	Abdullah	Khan	30
104	Gouwtham	Bhudha	5

5.2.12 Update Operation

Update operation is used to update the record in the table. UPDATE statement is used to do this operation

Example:

```
<%@ page import="java.io.* ,java.util.* ,java.sql.* "%>

<%@ page import="javax.servlet.http.* ,javax.servlet.* " %>

<%@ taglib uri="http://java.sun.com /jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com /jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>UPDATE Operation</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com .mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<c:set var="empId" value="102"/>

<sql:update dataSource="$ {snapshot}" var="count">

UPDATE Employees SET last = 'Sri'

<sql:param value="$ {em pId}" />

</sql:update>

<sql:query dataSource="$ {snapshot}" var="result">

SELECT * from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Em p ID</th>

<th>First Nam e</th>

<th>Last Nam e</th>
```

```
<th>Age</th>

</tr>

<c:forEach var="row" items="$ {result.rows}">

<tr>

<td><c:out value="$ {row.id}"/></td>

<td><c:out value="$ {row.first}"/></td>

<td><c:out value="$ {row.last}"/></td>

<td><c:out value="$ {row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</html>
```

The Output is

Emp ID	First Name	Last Name	Age
100	Ram	Kumar	18
102	John	Peter	25
103	Abdullah	Khan	30
104	Gouwtham	Sri	5

5.2.13 Delete Operation

Delete operation is used to delete the record in the table. DELETE statement is used to do this operation

Example

```
<%@ page import="java.io.* ,java.util.* ,java.sql.* "%>

<%@ page import="javax.servlet.http.* ,javax.servlet.* " %>

<%@ taglib uri="http://java.sun.com /jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com /jsp/jstl/sql" prefix="sql"%>
```

```
<html>

<head>

<title>DELETE Operation</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<c:set var="empId" value="103"/>

<sql:update dataSource="$ {snapshot}" var="count">

DELETE FROM Em ployees WHERE Id = ?

<sql:param value="$ {em pId}" />

</sql:update>

<sql:query dataSource="$ {snapshot}" var="result">

SELECT * from Em ployees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Em p ID</th>

<th>First Nam e</th>

<th>Last Nam e</th>

<th>Age</th>

</tr>

<c:forEach var="row" item s="$ {result.rows}">

<tr>

<td><c:out value="$ {row.id}"/></td>

<td><c:out value="$ {row.first}"/></td>

<td><c:out value="$ {row.last}"/></td>
```

```
<td><c:out value="$ {row.age}"/></td>

</tr>

</c:forEach>

</table>

</body>

</htm l>
```

The Output is

Emp ID	First Name	Last Name	Age
100	Ram	Kumar	18
102	John	Peter	25
103	Abdullah	Khan	30

5.3 DEVELOPING A MINI PROJECT USING HTML5, CSS, JSP TO MANIPULATE DATA IN MYSQL DATABASE.

Project Description:

This project " Product Stock Management System" is a web-based application. The main objective of this project is to manage a stock for a company or organization, and take care of maintaining the stock of the products. This project includes the modules for the following operation :

1. Adding new product to the stock
2. Updating / Deleting the product from stock
3. Getting report of the stock

Tables used :

The table named "product" is used here. It contains the following fields :

1. Product ID
2. Product Name
3. Quantity

Forms used :

S.No	Forms Name	Description
1.	Homepage	This form display main page with three links for data manipulation
2.	Insertform	This form receives input from the user and stored in the stock.

3.	Datamanipulateform	This form contains update and delete link for each record in the stock
4.	Updateform	This form is used to update the stored data in the stock
5.	Deleteform	This form is used to Delete particular record from the stock
6.	Productreportform	This form display all the products available in the stock

Programs to manipulate data :

The following jsp modules are used along with above forms for input and output operations :

S.No	JSP Modules	Operations
1.	Insert.jsp	This is used along with the insert form.
2.	Datamanipuate.jsp	This is used along with the datamanipulateform
3.	Update.jsp	This is used along with the updateform
4.	Delete.jsp	This is used along with the delete form
5.	Productreport.jsp	This is used along with the Productreportform

Software Required:

- ApacheTomcat 7 server
- MySql database server.
-

Starting MySql database server:

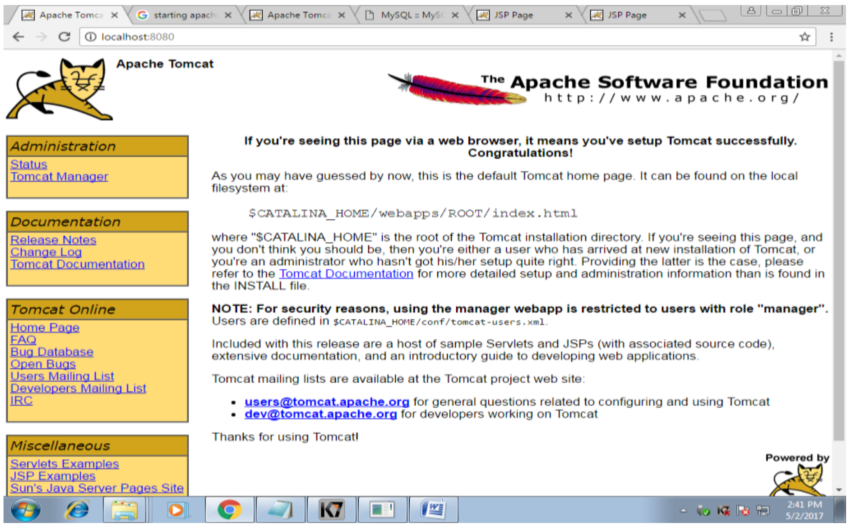
To start the mysqld server from the command line, enter the command prompt and enter this command:

C:\> "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld"

Starting Apache tomcat Server:

- Open any browser
- In the address bar type http://localhost:8080

The following page will be appear



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

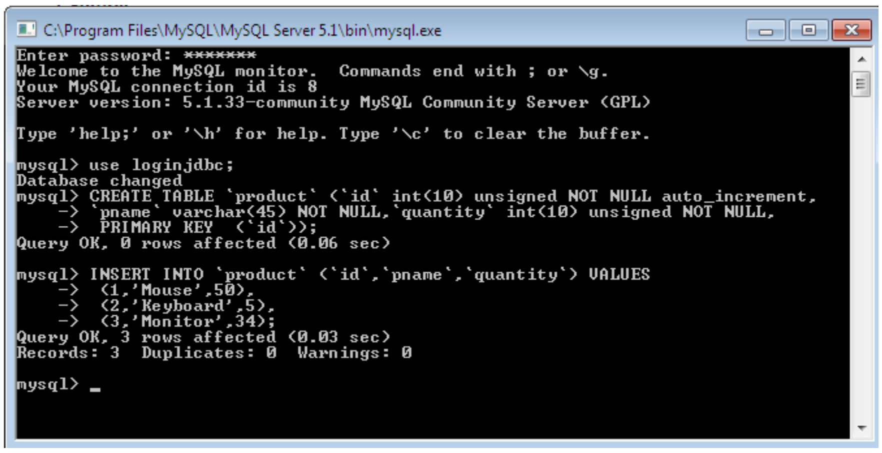
Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat
- **dev@tomcat.apache.org** for developers working on Tomcat

Thanks for using Tomcat!

fig5.2

Database Creation



```
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.1.33-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use loginjdbc;
Database changed
mysql> CREATE TABLE `product` (`id` int(10) unsigned NOT NULL auto_increment,
    -> `pname` varchar(45) NOT NULL,`quantity` int(10) unsigned NOT NULL,
    -> PRIMARY KEY  (`id`));
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO `product` (`id`,`pname`,`quantity`) VALUES
    -> (1,'Mouse',50),
    -> (2,'Keyboard',5),
    -> (3,'Monitor',34);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

fig 5.3

In Figure(fig5.3) mysql database and table is created. Database can be created using the following steps

- Open Mysql database
- Type “create database loginjdbc;” and press enter
- Now the database is created
- Type “Use loginjdbc:” and press enter

Table Creation

The table can be created using the following steps

- Open Mysql database
- Type the below query and press enter

```
CREATE TABLE `product` (`id` int(10) unsigned NOT NULL auto_increment,
`pname` varchar(45) NOT NULL,`quantity` int(10) unsigned NOT NULL,
 PRIMARY KEY  (`id`));
```

Inserting Records

For inserting record in a table,

- Open Mysql database
- Type the below query and press enter

```
INSERT INTO `product` (`id`,`pname`,`quantity`) VALUES
 (1,'Mouse',50),
 (2,'Keyboard',5),
 (3,'Monitor',34);
```

Steps to run the modules

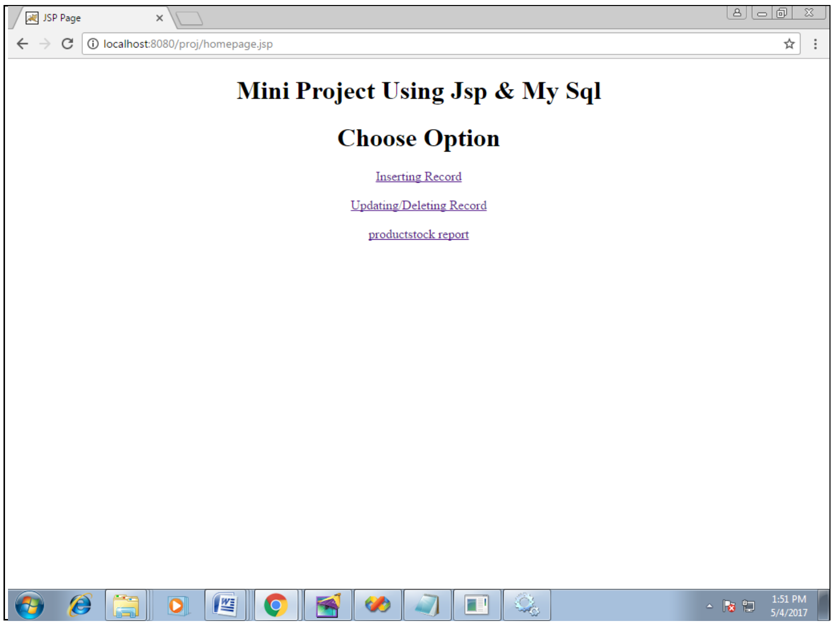
- Open any browser
- Type http://localhost:8080/homepage.jsp in the address bar.
- Press enter and the output will be displayed in the browser
- Now three links are displayed in the screen
- For insert a new record, click “Inserting Record” link in the output screen.
- For update/delete a record click “Updating/Deleting Record” link in the output screen.
- For view the stock report click “Product Stock Report” link in the screen.

Module-I

In this module, we have created a home page(index.jsp). It has three links to manipulate the data.

They are

- Inserting Record –It is used to insert a new record.
- Updating/Deleting record- It is used to modify and delete the data.
- Product stock report - It is used to view all the products in the database.



Mini Project Using Jsp & My Sql

Choose Option

Inserting Record

Updating/Deleting Record

productstock report

Fig 5.4

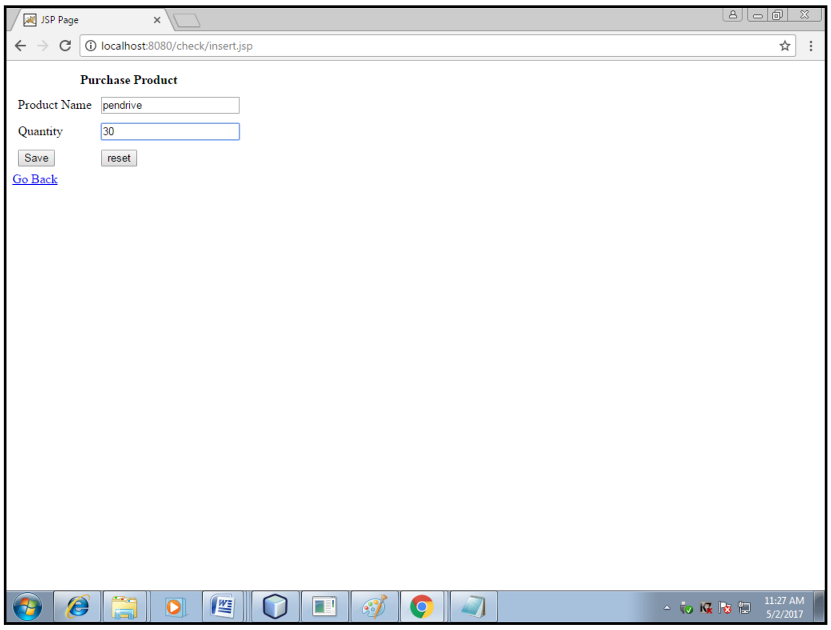
homepage.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<h1 align=center> Mini Project Using Jsp & My Sql</h1>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<h1 align=center>Choose Option</h1>
<P align=center>
<a  href="insert.jsp">Inserting Record</a><br><br>
<a href="datamanipulate.jsp">Updating/Deleting Record</a><br><br>
<a  href="productreport.jsp">productstock report</a><br><br>
</p>
</p>
</body>
</html>
```

Module II

Inserting Record:

In this module we have to insert a new record into the database. For inserting a record we have to establish a connection between tomcat server and mySql database by using JDBC.



localhost:8080/check/insert.jsp

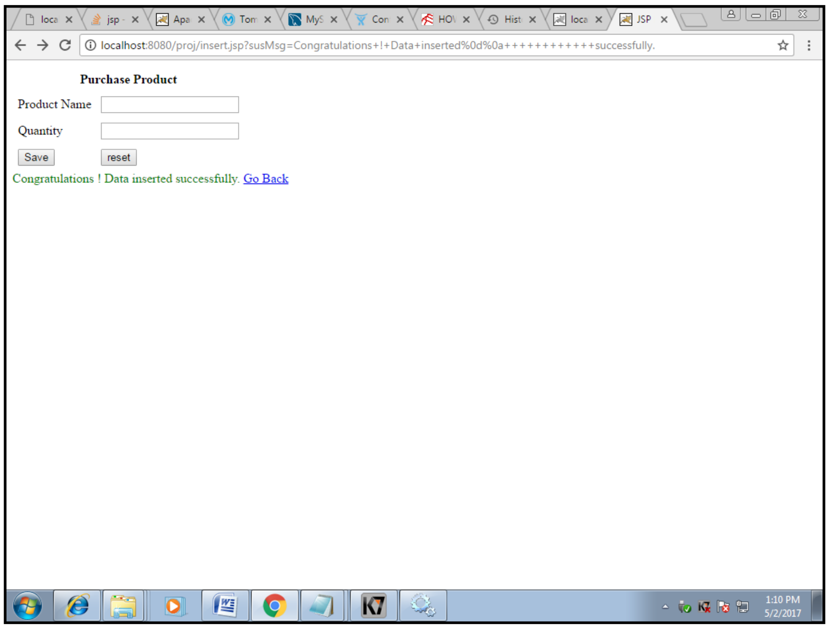
Purchase Product

Product Name: pendrive

Quantity: 30

Fig 5.5

- Click save button in the screen .Now the new record is saved in the database and display the result as below fig 5.6



localhost:8080/proj/insert.jsp?susMsg=Congratulations+!+Data+inserted%0d%0a+++++++++++successfully.

Purchase Product

Product Name

Quantity

Congratulations ! Data inserted successfully. Go Back

Fig 5.6

The following code (insertform.jsp) is used to get the values from the user and store data in the database

Insertform.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<form action="insert.jsp" method="post">
<table border="0" cellspacing="2" cellpadding="5">
<thead>
<tr>
<th colspan="2">Purchase Product</th>
</tr>
</thead>
<tbody>
<tr>
<td><label>Product Name</label></td>
<td><input type="text" name="pname"/></td>
</tr>
<tr>
<td><label>Quantity</label></td>
<td><input type="text" name="qty"/></td>
</tr>
<tr>
<td><input type="submit" value="Save" /></td>
<td><input type="reset" value="reset"/></td>
</tr>
</tbody>
</table>
</form>
<font color="red"><c:if test="${not empty param.errMsg}">
<c:out value="${param.errMsg}" />
<a href="index.jsp">Go Back</a>
</c:if></font>
<font color="green"><c:if test="${not empty param.susMsg}">
<c:out value="${param.susMsg}" />
<a href="homepage.jsp">Go Back</a>
</c:if></font>
</body>
</html>
```

. The following code **insert.jsp** is used to establish connection and stores the values into the database

Algorithm

- Create datasourse using <sql:setDataSource> tag before establishing database connection.

- Use JDBC drivers for establishing connections.
- Insert query is used to insert data into the database.
- Get Product name and Quantity as input from the users.

insert.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>JINSERT Operation</title>
</head>
<body>
<c:if test="${ empty param.pname or empty param.qty}">
<c:redirect url="insertform.jsp" >
<c:param name="errMsg" value="Please Enter Product and Quantity" />
</c:redirect>
</c:if>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/loginjdbc"
user="root"  password="pass123"/>
<sql:update dataSource="${dbsource}" var="result">
INSERT INTO product(pname, quantity) VALUES (?,?);
<sql:param value="${param.pname}" />
<sql:param value="${param.qty}" />
</sql:update>
<c:if test="${result>=1}">
<font size="5" color='green'> Congratulations ! Data inserted
successfully.</font>
<c:redirect url="insertform.jsp" >
<c:param name="susMsg" value="Congratulations ! Data inserted
successfully." />
</c:redirect>
</c:if>
</body>
</html>
```

Module III

Updating/Deleting Records:

After clicking the updating/deleting record link in the home page. All records are displayed with two links named “update” and “delete” for each row.

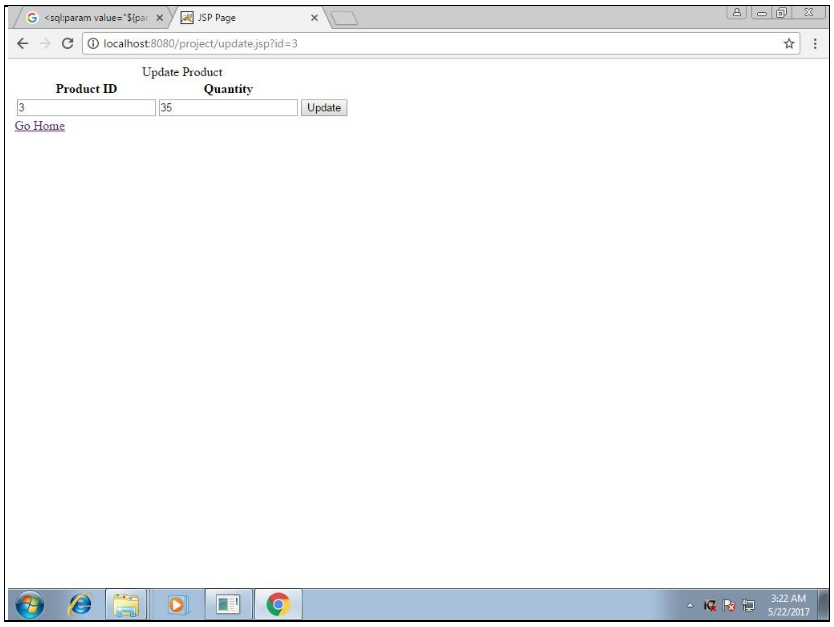
datamanipulate.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>SELECT Operation</title>
<script>
function confirmGo(m,u) {
if ( confirm(m) ) {
window.location = u;
}
}
</script>
</head>
<body>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/loginjdbc"
user="root"  password="pass123"/>
<sql:query dataSource="${dbsource}" var="result">
SELECT * from product;
</sql:query>
<center>
<form>
<table border="1" width="40%">
<caption>Product List</caption>
<tr>
<th>Product ID</th>
<th>Product Name</th>
<th>Quantity</th>
<th colspan="2">Action</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
<td><c:out value="${row.id}"/></td>
<td><c:out value="${row.pname}"/></td>
<td><c:out value="${row.quantity}"/></td>
<td><a href="update.jsp?id=<c:out value="${row.id}"/>">Update</a></td>
<td><a href="javascript:confirmGo('Sure to delete this record?','deletedb.jsp?id=<c:out
value="${row.id}"/>')">Delete</a></td>
</tr>
</c:forEach>
</table>
</form>
<a href="homepage.jsp">Go Home</a>
</center>
```

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

Update records

Updating records is used to modify the values that are already stored in the database.



localhost:8080/project/update.jsp?id=3

Update Product

Product ID	Quantity	
3	35	Update

Go Home

Fig 5.7

After Clicking the update button from the above figure (fig 5.7) the data are updated in the database.

Updateform.jsp

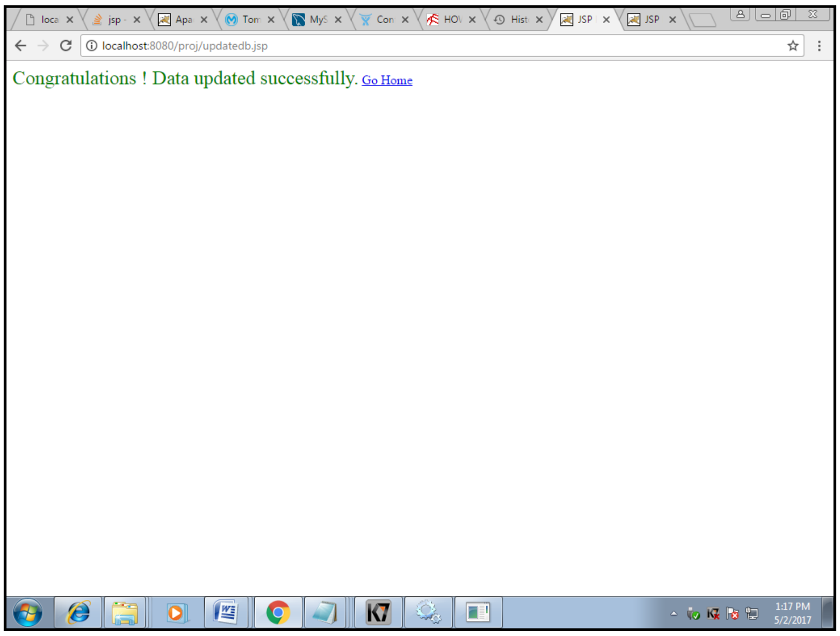
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/loginjdbc"
user="root"  password="pass123"/>
<sql:query dataSource="${dbsource}" var="result">
SELECT * from product where id=?;
<sql:param value="${param.id}" />
</sql:query>
```

```
<form action="updatedb.jsp" method="post">
<table border="0" width="40%">
<caption>Update Product</caption>
<tr>
<th>Product ID</th>
<th>Quantity</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
<td><input type="text" value="${param.id}" name="id"/>
<input type="hidden" value="${row.pname}" name="pname"/></td>
<td><input type="text" value="${row.quantity}" name="qty"/></td>
<td><input type="submit" value="Update"/></td>
</tr>
</c:forEach>
</table>
<a href="index.jsp">Go Home</a>
</form>
</body>
</html>
```

Update.jsp

Algorithm

- Create datasourse using <sql:setDataSource> tag before establishing database connection.
- Use JDBC drivers for establishing connections.
- Update query is used to update data in the database.
- Product ID and Quantity fields are used in the form to perform update operation.


localhost:8080/proj/updatedb.jsp

Congratulations ! Data updated successfully. Go Home

Fig 5.8

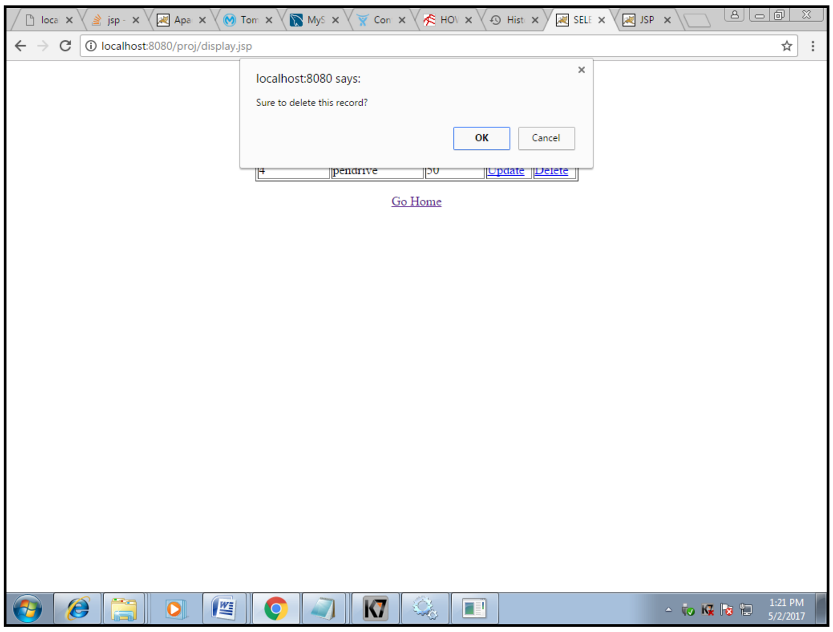
After Updating records in the database the output will be as fig 5.8 and the complete code to update the database is as follows

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/loginjdbc"
user="root"  password="pass123"/>
<sql:update dataSource="${dbsource}" var="count">
UPDATE product SET pname = ?, quantity=?
WHERE id='${param.id}'
<sql:param value="${param.pname}" />
<sql:param value="${param.qty}" />
</sql:update>
<c:if test="${count>=1}">
<font size="5" color='green'> Congratulations ! Data updated
successfully.</font>
<a href="homepage.jsp">Go Home</a>
</c:if>
</body>
```

</html>

Deleting records

The SQL DELETE Query is used to delete the existing records from a table.



localhost:8080/proj/display.jsp

localhost:8080 says:

Sure to delete this record?

4	pendrive	50	Update	Delete

Go Home

Fig 5.9

When the below code delete.jsp is executed the above output (fig 5.9) is displayed

Algorithm

- Create datasourse using <sql:setDataSource> tag before establishing database connection.
- Use JDBC drivers for establishing connections.
- Delete query is used to delete data in the database.

Delete.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
```

```
<body>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/loginjdbc"
user="root"  password="pass123"/>
<sql:update dataSource="${dbsource}" var="count">
DELETE FROM product
WHERE id='${param.id}'
</sql:update>
<c:if test="${count>=1}">
<font size="5" color='green'> The file was deleted successfully.</font>
<a href="homepage.jsp">Go Home</a>
</c:if>
</body>
</html>
```

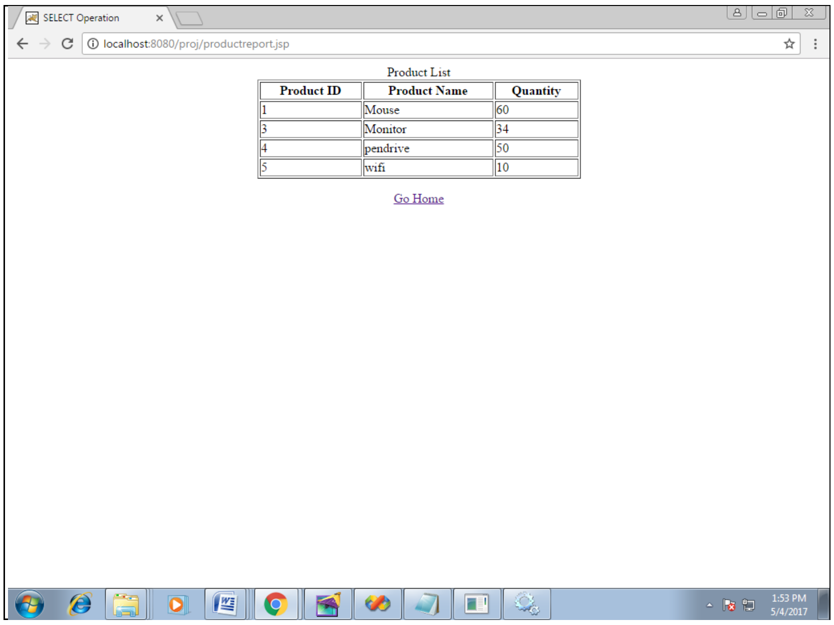
Module IV

Viewing Records:

In this module "productstockreport.jsp" file contains coding for display all the records.

Algorithm

- Create datasourse using <sql:setDataSource> tag before establishing database connection.
- Use JDBC drivers for establishing connections.
- Select query is used to display all the data in the database.
- Product ID.Product Name and Quantity are displayed in table format.



SELECT Operation

localhost:8080/proj/productreport.jsp

Product List

Product ID	Product Name	Quantity
1	Mouse	60
3	Monitor	34
4	pendrive	50
5	wifi	10

Go Home

1:53 PM
5/4/2017

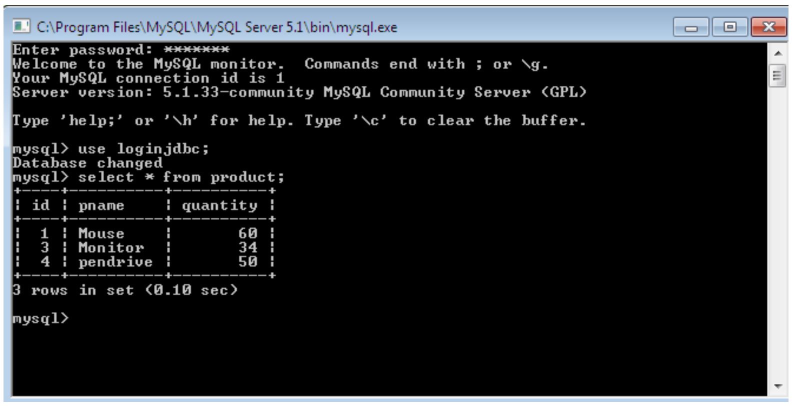
Fig 5.10

Productstockreport.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>SELECT Operation</title>
<script>
function confirmGo(m,u) {
if ( confirm(m) ) {
window.location = u;
}
}
</script>
</head>
<body>
<sql:setDataSource var="dbsource" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/loginjdbc"
user="root"  password="pass123"/>
<sql:query dataSource="${dbsource}" var="result">
SELECT * from product;
</sql:query>
<center>
<form>
<table border="1" width="40%">
<caption>Product List</caption>
<tr>
<th>Product ID</th>
<th>Product Name</th>
<th>Quantity</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
<td><c:out value="${row.id}"/></td>
<td><c:out value="${row.pname}"/></td>
<td><c:out value="${row.quantity}"/></td>
</c:forEach>
</table>
</form>
<a href="index.jsp">Go Home</a>
</center>
</body>
</html>
```

Steps to check the records in mysql after run this project

- Open Mysql database
- Type "Use loginjdbc:" and press enter
- Type the query "Select * from Product;" and press enter.
- The complete database report is displayed as fig 5.11



```
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.33-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use loginjdbc;
Database changed
mysql> select * from product;
+----+----------+----------+
| id | pname    | quantity |
+----+----------+----------+
|  1 | Mouse    |       60 |
|  3 | Monitor  |       34 |
|  4 | pendrive |       50 |
+----+----------+----------+
3 rows in set (0.10 sec)

mysql>
```

Fig 5.11

Summary

- ✓ JSP technology is used to create dynamic web applications.
- ✓ MySQL is one of the best RDBMS being used for developing web-based software applications
- ✓ The CREATE TABLE statement is used to create a table in MySQL.
- ✓ The <sql:setDataSource> creates a datasource object to connect to a database.
- ✓ url Attribute Specifies the location of the database.
- ✓ JDBC drivers are used to open database connections and to interact with it by sending SQL or database commands then receiving results with Java.
- ✓ Database can be access only after creating connection. Before creating connection, a driver must be loaded and registered.
- ✓ JDBC Bridge is used to access ODBC drivers installed on each client machine.
- ✓ The driver manager maintains a list of registered drivers.
- ✓ The SQL language consists of statements that create, manipulate, and extract data from a relational database.
- ✓ Data manipulation commands return a count of the number of rows modified, whereas a SELECT statement returns a set of rows known as a result set.
- ✓ The basic interface for creating a statement is java.sql.Statement.
- ✓ executeUpdate() method can be used with the SQL INSERT, UPDATE, or DELETE statements, or with data definition statements such as CREATE TABLE.
- ✓ executeQuery() method is used to execute an SQL SELECT statement and to return a result set.
- ✓ Select operation is used to select record in the database. SELECT statement is used to do this operation

- ✓ Update operation is used to update the record in the table. UPDATE statement is used to do this operation
- ✓ Delete operation is used to delete the record in the table. DELETE statement is used to do this operation

Review Questions

Part A

1. Define : MySql.
2. What is the use of url attribures?
3. Write the general syntax to create a table.
4. Define : Statement.
5. Write the use of update operation?
6. What is the use of select operation?
7. Define CSS.
8. How to delete a record in the database?

Part B

How to create records in mysql?.

Write the jsp code to convert entered text into uppercase.

Briefly explain about connection.

Write about select operation in mysql.

Write about var and driver attributes.

Write short notes on creating statements.

What are the uses of JSP?

Part C

1. Write a jsp program to find maximum of three numbers
2. Explain about <sql:setDataSource> in detail.
3. Discuss about JdbcOdbcDriver with neat diagram.
4. Explain in detail about executeUpdate().
5. With an example, explain about insert operation.

6. Write about executeQuery() with example.
7. Write a jsp program to add two numbers.
8. With example explain about update operation.
9. Explain about delete operation with example.
10. Write a jsp code to manipulate data in mysql database using html and css.
