➢ Introduction to Internet:

The Internet is the global system of interconnected computer networks that use the Internet protocol suite to link devices worldwide. The purpose of the internet is to communicate between computers that are interconnected with each other. Internet is accessible to every user all over the world.

It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.

The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony and file sharing.

Browser is a tool used to access the internet using WWW (World Wide Web) and HTTP (Hyper Text Transfer Protocol). In the browser, if the user types the domain name such as www.tn.gov.in, the browser calls a protocol name DNS (Domain Name Server). DNS is used to get the IP address of the domain names.

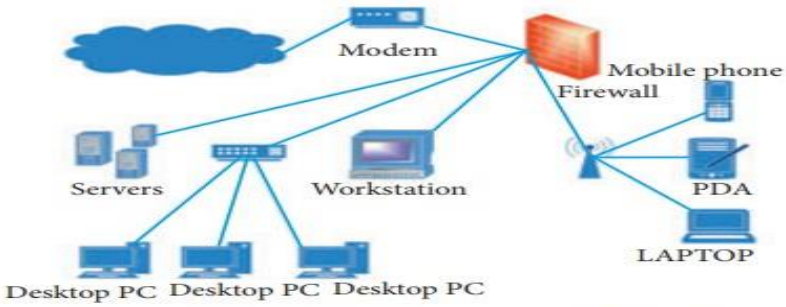


Figure 15.1 Block Diagram of Internet

Evolution of Internet:

Internet evolved in 1969 and evolved many changes in several technologies and Infrastructural levels.

• Internet was started by ARPANET (Advanced Research Project Agency Network), developed by United States. Department of Defence for communication among different government bodies, initially with four nodes.

• In 1972, the four nodes has been developed and it grown to 23 nodes located in different countries making it Internet.

• Invented TCP/IP protocols, DNS, WWW, browsers scripting languages.

• Internet is used as a medium to publish and access the information

• In 1985, The NSFNET was composed of multiple regional networks and peer networks

• In 1986, the NSFNET created a three-tiered network architecture.

• In 1988, updated the links to make it faster

• In 1990, Merit, IBM, and MCI started a new organization known as Advanced Network and Services (ANS).

• By 1991, data traffic had increased tremendously, which necessitated upgrading the NSFNET's backbone network service to T3 (45 Mbps) links.

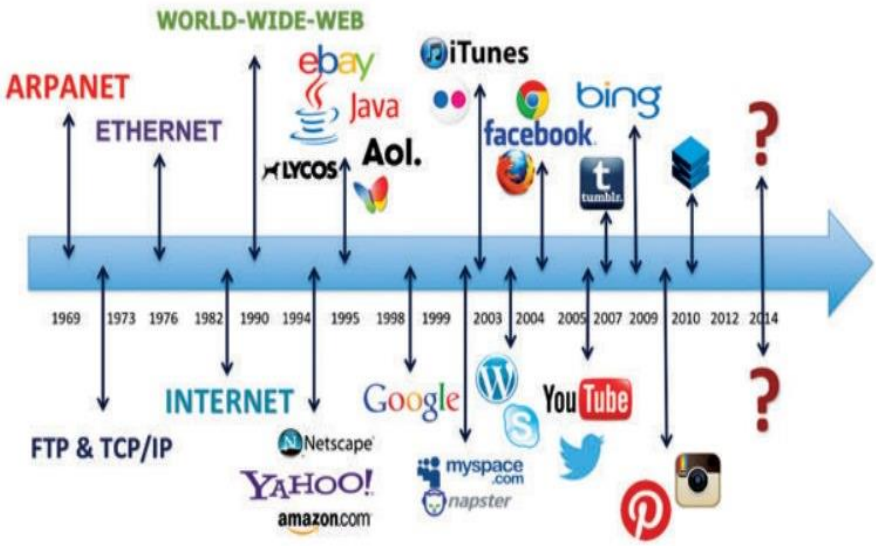


Figure 15.2 Internet History Timeline

Internet Evalution:

Internet covers almost every aspect of life. Internet allows the users to communicate with the people sitting at remote locations. There are various applications available on the web that uses Internet as a medium for communication. One can find various social networking sites such: Facebook, Twitter, Yahoo, Google+, Flickr, and Orkut. One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health and Science, Social Studies, Geographical Information, Information Technology and Products can be surfed with help of a search engine.

Apart from communication and source of information, internet also serves as a medium for entertainment. Internet also allows the users to use as many services as like E-mail, Internet Banking, Online Shopping, Online Ticket Booking, Online Bill Payment and Data Sharing. Internet provides concept of electronic commerce that allows the business deals to be conducted on electronic systems.

Hardware and Software Requirements for Internet connection:

The following are the methods of connecting a computer to the Internet using software and hardware peripherals.

Three

- Connecting a computer using Wireless Broadband
- Connecting a computer using an Ethernet Cable
- Connecting a Computer Using Dial-Up Community

Hardware Requirement:

- To connect the Internet, any one of the following is mandatory.
- Modem is used to connect Internet through Telephone connection.
- NIC- Network Interface Card(wired/ wireless) facility is the most important hardware required to connect Internet. For example, the Laptop can be connected Internet through the wired/wireless.
- Dongle is used to connect the Internet using cellular network
- Wi-Fi router or Hotspot is used to connect the Internet using wireless network
- Electronic device which supports cellular network
- Internet Connectivity such as Dial-up connection, ISDN, DSL, Cable TV, wired and wireless (Cellular) Network.

Software Requirement

- The operating system should support TCP (Transfer Control Protocol) / IP (Internet Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), HTTP (Hyper Text Transfer Protocol) and HTTPS (Hyper Text Transfer Protocol Secured) protocols.
- Browsers and other Internet clients access to the web applications such as Outlook, Gmail, Whatsapp, Facebook, Twitter and etc.

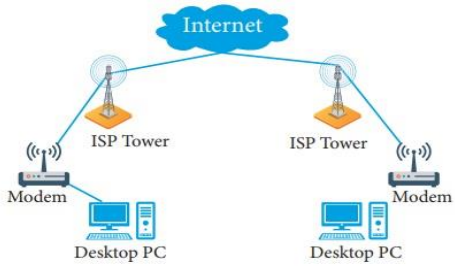


Figure 15.3 Dial-up Connection

Connection Types:

The following methods are able to connect internet.

Dial-up Connection :

A dial-up connection is established when two or more data communication devices use a Public Switched Telephone Network (PSTN) to connect to an Internet Service Provider (ISP) from computers. Many remote locations depend on Internet dial-up connections because broadband and cable are rare in remote areas with low population. Internet Service Providers often provide dial-up connections, a feasible alternative for budget-conscious subscribers.

Figure 15.4a Integrated Services Digital Network

ISDN

ISDN is the acronym of Integrated Services Digital Network. It establishes the connection using the phone lines (PSTN) which carry digital signals instead of analog signals. It is a set of communication standards for simultaneous digital transmission of data, voice, video, and other services over the traditional circuits of the public switched telephone network. There are two techniques to deliver ISDN services such as Basic Rate Interface (BRI) and Primary Rate Interface (PRI).

The following diagram shows accessing internet using ISDN connection:

DSL:

Digital Subscriber Line (DSL) is a high-speed Internet service for homes and businesses that competes with cable and other forms of broadband Internet. DSL provides high-speed networking over ordinary Telephone lines using broadband modem technology. The technology behind DSL enables Internet and telephone service to work over the same phone line without requiring customers to disconnect either their Voice or Internet connections.

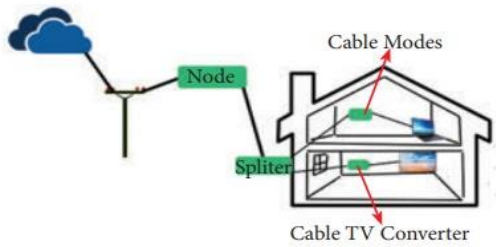


Figure 15.4b Cable TV Connection

Cable TV Internet Connection (setup box):

The cable TV network can be used for connecting a computer or a local network to the Internet, competing directly with DSL (Digital Subscriber Line) technology.

This type of network is classified as HFC (Hybrid Fiber-Coaxial), as it uses both fiber optics and coaxial cables. The connection between the cable TV company to the distribution points (Optical nodes) is made using fiber optics, with distances up to 25 miles (40 km). Each optical node is typically serves between 500 and 2,000 clients (customers).

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

Satellite Internet Connection:

Satellite Internet access is Internet access provided through satellite communication for domestic and enterprise usage. The facility of modern consumer grade satellite Internet service is typically provided to individual users through geostationary satellites. It provides fairly high data speeds, along with latest satellites using Ka-band to attain downstream data speeds up to 50 Mbps internet speed.

Wireless Internet Connection:

It is a technology for wireless local area networking with devices based on the IEEE 802.11 standards. Devices that can use Wi-Fi technology include personal computers, video-game consoles, phones and tablets, digital cameras, smart TVs, digital audio players and modern printers. Wi-Fi compatible devices can connect to the Internet via a WLAN and a wireless access point. Such an access point (or hotspot) has a range of about 20 meters (66 feet) indoors and a greater range of outdoors. Hotspot coverage can be as small as a single room with walls that block radio waves, or as large as many square kilometres achieved by using multiple overlapping access points.

Services Available on the Internet:

- Data Transfer
- Internet banking
- E-commerce
- E-Learning
- E-Governance
- Browsing and Chating
- E-Mail

➢ Browser Architecture-IE

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 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 is a lot of web browser 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 browsers must exhibit are listed below:

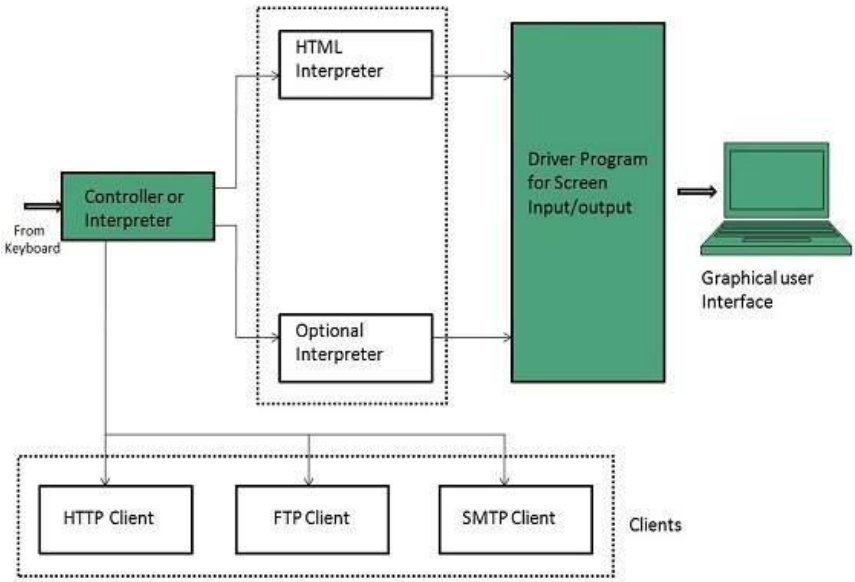
- Controller/Dispatcher
- Interpreter
- Client Programs

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 executes 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 that 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:

Navigation

A web page may contain hyperlinks. When we click on these links other web page is opened. These hyperlinks can be in form of text or image. When we take the mouse over an hyperlink, pointer change its shape to hand.

Key Points

- In case, you have accessed many web pages and willing to see the previous webpage then just click back button.
- You can open a new web page in the same tab, or different tab or in a new window.

Saving Webpage

You can save web page to use in future. In order to save a webpage, follow the steps given below:

- Click File > Save As. Save Webpage dialog box appears.
- Choose the location where you want to save your webpage from save in: list box. Then choose the folder where you want to save the webpage.
- Specify the file name in the File name box.
- Select the type from Save as type list box.
  - Webpage, complete
  - Web Archive
  - Webpage HTML only
  - Text File
- From the encoding list box, choose the character set which will be used with your webpage. By default, Western European is selected.
- Click save button and the webpage is saved.

Saving Web Elements

Web elements are the pictures, links etc. In order to save these elements follow the steps given below:

- Right click on the webpage element you want to save. Menu options will appear. These options may vary depending on the element you want to save.

Evolution of the Web Browsers

Tim Berner Lee was the founder of the Web Browser as a concept developed in 1990. The initial name of his browser was World Wide Web but now it has become Nexus. Erwise was the first graphical user interface web browser. Robert Cailliau was the man behind this development.

Marc Andreessen developed the browser further in 1993 by releasing one of the most popular browsers of all time – Mosaic. The WWW system became more user-friendly and accessible after this. Andresen lead company Netscape reached its boom in 1994 with 90% of the users using it.

Internet Explorer by Microsoft came out in 1995 giving major competition to Mosaic. The upper advantage of windows in internet explorer led to increase in its market share to 95% by 2002. Opera came out in 1996 but did not manage to get a stronghold in the market by only having a 2% share by 2012.

Netscape's Mozilla foundation led to the introduction of the Firefox browser in 1998 that grew to 27% of the market share in 2011. Apple Safari was late in the market by coming out in 2003 but yet it is the dominant browser for Ios users around the globe.

Chrome which accounts for the majority of the users in the world came out in 2008 but its usage only increased with every passing year. By 2014, 45% of the internet users were using google chrome as their go-to web browser.

How the World Wide Web Works?

The clients server format enables the basic functioning of the web. The users request information on the web pages and the server works on transferring it to them. Software to serve these pages to the users is a web server. And the user's computer here becomes the client and the browser on this allows document retrieval.

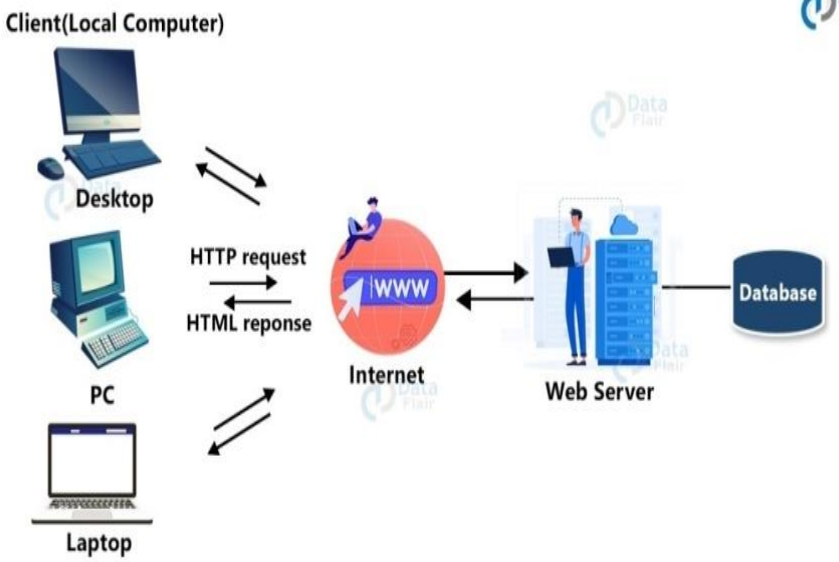
All websites are on this server as guests and every time there is a request, the server pays the hosting price. The second users type an URL on the search, www begins to work. The three key elements here are Hypertext Markup Language (HTML), Hypertext Transfer Protocol (HTTP) and Web browsers.

How does a web browser work?

Like mentioned in the introduction, browser allows sending and receiving of information from all parts of the web. The Hypertext Transfer Protocol allows data transfer while governing all pieces

of content. The user interface is what enables graphical representation of the data and allows users to explore it.

The backend of the browsers are the one handling requests and becomes the carrier for information transfer and interaction. While the clean front end allows users to scroll and browse. The backend supports it completely with handling all the functions smoothly. This is how the web browsers function.



Components of a Web Browser

1. User Interface

It is an environment allowing users to use certain features like search bar, refresh button, menu, bookmarks, etc.

2. Browser Engine

The bridge connects the interface and the engine. It monitors the rendition engine while manipulating the inputs coming from multiple user interfaces.

3. Networking

The protocol provides an URL and manages all sorts of safety, privacy and communication. In addition, the store network traffic gets saved in retrieved documents.

4. Data Storage

The cookies store information as the data store is an uniform layer that the browsers use. Storage processes like IndexedDB, WebSQL, localStorage, etc works well on browsers.

5. JavaScript Interpreter

It allows conversion of JavaScript code in a document and the executes it. Then the engine shows the translation on the screen to the users.

Architecture of Web Browser

The architecture of a web browser has certain components. They are Controller/Dispatcher, Interpreter, and Client Programs. The control unit in a CPU is a controller that takes in the input, interprets it, and then instructs the device to work in a certain way.

The interpreter receives this info from the controller to get ahead in the given task step by step. Lastly, the client program has sets of protocols to complete a particular service. Some common names are – HTTP, SMTP, FTP, etc.

Websites

As mentioned above, the location on a web browser using a URL opens up a website. They are of two types –

1. Static Websites – These are flat or stationary websites made by the client that only they can change. The users can read and watch them but cannot interact with the content in any way. They use HTML language to create as the information is fixed.

2. Dynamic Websites – These websites keep updating and the users will find new information every time. The ajax technology allows users to change information as they want to make it very dynamic.

Web Pages

The web pages are the places where users find all sorts of information. The hyperlinks to many other pages allowing users to switch and explore. And a collection of these pages make up a

website. Web pages are also of two types like websites – Static and Dynamic. The brief for them is the same as above.

Under dynamic web pages, they can be either Server-side dynamic web pages or Client-side dynamic web pages. The server-side dynamic page has server-side scripting while the other one has client-side scripting using the Document Object Model. The scripting has become quite common nowadays and some of them are –

Client-Side Web Scripting

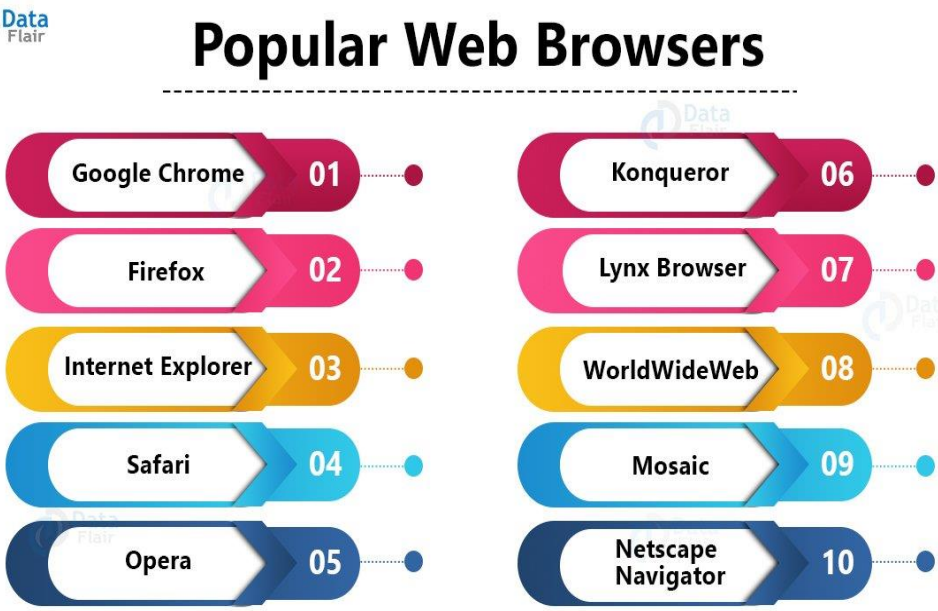
1. JavaScript – prototype-based scripting that inherits its name from java and is saved as a .js extension.
2. ActionScript – an object-oriented language used to target adobe flash player for website development.
3. Dart – source to source compiler and is the open-source language by Google.
4. VBScript – open source web language by Microsoft that has static typing class-based object programming.

Server-side Web Scripting

1. Active Server Pages – supports Component Object Model that uses functions of DLL.
2. ActiveVFP – uses native Visual Foxpro language and database for website creation
3. ASP.net – develops websites, web applications, and web services
4. Java – Java code with byte code creates the website
5. Python – object-oriented and functional programming for website creation
6. WebDNA – uses an embedded database system

Features and Functions of Web Browser

- Giving users access to all sorts of information is the primary function of a web browser that includes retrieval, display, and navigation.
- The new browsers are very minimal with an easy user interface with support of HTML opening all file formats and protocols in one place.
- There are internet suites to support email, Usenet news, and Internet Relay Chat and they are often not the same as web browsers.
- Users can open several information resources at the same time in the same window or different windows.
- They also block the redirect to prevent users from reaching unwanted windows and websites without their consent.
- There are options for users to bookmark the web pages so it's easier and quicker to access in the future. These bookmarks are the user's "favorites" and most of the browsers have this as an inbuilt feature.
- The user interfaces is mostly similar in all browsers with a refresh or reload button, stop button, an address bar, a search bar, a status bar, a viewport, and other incremental find features on the web page.
- There are a few devices with photos of LG Smart TV on-screen keyboard with a magic motion remote.
- There is an option to delete web history, cookies, and cache ensuring security as well as privacy of the user.
- Before there was only HTML support but the new browsers have HTML and XHTML making the process smoother.
- There is a browser extension as well for extending the web browser's functionality.
- The web browsers have a user interface, layout engine, scripting, interpreter, user interface backend, networking, and data.



1. Google Chrome

Google Chrome has the most number of users in the world. It came out very late but its effective results and a large pool of information attracted users to it making google the most preferred search engine. This works well on most of the operating systems and devices becoming easily accessible as well. 89.0.4389.105 is the latest version of this browser.

2. Firefox

One of the widely used web browsers, it supports all sorts of operating systems becoming very flexible for the users. And because it is open-source, a large community of developers uses it regularly. There are many plugins/add-ons allowing customization. It is a Mozilla foundation product and the latest version is Firefox

3. Internet Explorer

A windows operating system browser, internet explorer comes preinstalled in all Microsoft computers. The design is a way to improve the user interface so they can open multiple web pages at the same time within the operating system The latest versions are Internet Explorer 7 with 8 in beta.

4. Safari

Being an Apple product, safari works only on Ios devices like iPhone, MacBook, iPad, etc. This browser comes pre-installed on these devices as other browsers cannot function well on Apple devices. The latest version of this browser is macOS 11 Big Sur launched in 2021.

5. Opera

Opera is a commonly used mobile browser with magnification and gesture features to handle internet-related tasks. It has inbuilt malware and phasing security with strong encryption to secure user's and client's personal data and information. Opera works well on Windows, OS X, and Linux.

6. Konqueror

An open-source browser that supports all basic file management on local UNIX to advanced remote and local network file browsing. It is not the most preferred option by the users due to lack of user-friendliness and results

7. Lynx Browser

Again not a very famous name, it is a World Wide Web browser for Umic and VMS users. They use it to run cursor-addressable, terminals or emulators.

8. WorldWideWeb

The first web browser came out in 1990 but now its allied Nexus. This is to avoid the misconception between www as a concept and nexus as a browser. Not used very commonly due to less interactive interface and no bookmark option.

9. Mosaic

The second web browser that came out in 1993 with an improved interface and was by the National Center for Supercomputing Applications. Marc Andersson being the team leader made it very popular.

10. Netscape Navigator

This browser came out in 1994 and became the market leader for good 10 years. Netscape kept improving by launching better versions with license schemes and free usage.

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

➢ Chrome-Search Engine Introduction

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:

1. Web Crawler
2. Database
3. 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.

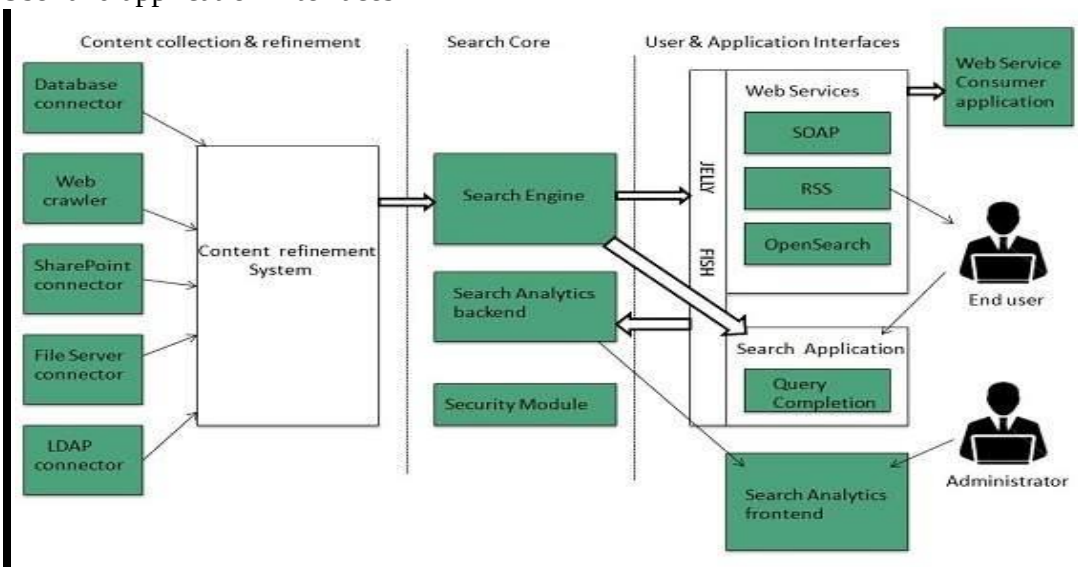
These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

- 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

Text acquisition

It identifies and stores documents for indexing.

Text Transformation

It transforms document into index terms or features.

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

User interaction
It supporst creation and refinement of user query and displays the results.
Ranking
It uses query and indexes to create ranked list of documents.

➢ Introduction to HTML

What is HTML
HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Let's see a simple example of HTML.

```
1.  <!DOCTYPE>
2.  <html>
3.  <head>
4.  <title>Web page title</title>
5.  </head>
6.  <body>
7.  <h1>Write Your First Heading</h1>
8.  <p>Write Your First Paragraph.</p>
9.  </body>
10. </html>
```

Description of HTML Example

<! DOCTYPE>: It defines the document type or it instructs the browser about the version of HTML.

<html >:This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata (information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body> : Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p> : Text between <p> tag describes the paragraph of the webpage.

HTML5 : HTML5 is the newest version of HyperText Markup language. The first draft of this version was announced in January 2008. There are two major organizations one is W3C (World Wide Web Consortium), and another one is WHATWG(Web Hypertext Application Technology Working Group) which are involved in the development of HTML 5 version, and still, it is under development.

Features of HTML

1) It is a very **easy and simple language**. It can be easily understood and modified.

2) It is very easy to make an **effective presentation** with HTML because it has a lot of formatting tags.
3) It is a **markup language**, so it provides a flexible way to design web pages along with the text.
4) It facilitates programmers to add a **link** on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.
5) It is **platform-independent** because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.
6) It facilitates the programmer to add **Graphics, Videos, and Sound** to the web pages which makes it more attractive and interactive.
7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

➢ HTML5

HTML5 provides details of all 40+ HTML tags including audio, video, header, footer, data, datalist, article etc. This HTML tutorial is designed for beginners and professionals.
HTML5 is a next version of HTML. Here, you will get some brand new features which will make HTML much easier. These new introducing features make your website layout clearer to both website designers and users. There are some elements like <header>, <footer>, <nav> and <article> that define the layout of a website.

Why use HTML5

It is enriched with advance features which makes it easy and interactive for designer/developer and users.
It allows you to play a video and audio file.
It allows you to draw on a canvas.
It facilitate you to design better forms and build web applications that work offline.
It provides you advance features for that you would normally have to write JavaScript to do.
The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C recommendation.
HTML 5 Example
Let's see a simple example of HTML5.

```
1. <!DOCTYPE>
2. <html>
3. <body>
4. <h1>Write Your First Heading</h1>
5. <p>Write Your First Paragraph.</p>
6. </body>
7. </html>
```

HTML5 User Manual

HTML Coding Convention
You should follow some convention while using HTML:
Use a Consistent CSS
A user must use consistent style while writing HTML. It makes the code simpler and more understandable for people.
Your code must be small, clean and well-formed.
Use Correct Document Type
You should declare document type at the starting of your code..2M630TML Tutorial
For example:
<!DOCTYPE html>
You can also use <!DOCTYPE html> to maintain your lower case practice.
Use Lower Case Element Names
HTML5 facilitates you to use upper case and lower case letters in element name. But it is good practice to use only lower case. Because:

- Mixing lower case and upper case letters in elements is not a good idea.
- Lowercase looks neat and clean.
- Lower case is easy to write.

- Developers mainly use lower case letters.

Example:

Bad practice:

```
<SECTION>
<p>This is javatpoint.com</p>
</SECTION>
```

Very Bad practice:

```
<Section>
<p>This is a javatpoint.com</p>
</SECTION>
```

Good practice:

```
<section>
<p>This is javatpoint.com.</p>
</section>
```

Close all HTML Elements

In HTML5, it is not required to close all HTML tags. But, it is recommended to close tags.

Bad practice:

```
<section>
<p>This is javatpoint.com
</section>
```

Good practice:

```
<section>
<p>This is javatpoint.com</p>
</section>
```

Close empty HTML Elements

In HTML5, it is not mandatory to close empty HTML tags. You can close it or let it be unclosed.

Good practice:

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

Don't Omit <html> and <body>

HTML5 facilitates you to omit and tag. You can exclude both tags and the program will work well enough.

Example:

```
<!DOCTYPE html>
<head>
<title>Page Title</title>
</head>
<h1>This is javatpoint.com</h1>
<p>Welcome to javatpoint.com</p>
```

Syntax:

```
<!DOCTYPE html>
<html lang="en-US">
```

Example:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

HTML 5 Tags

There is a list of newly included tags in HTML 5. These HTML 5 tags (elements) provide a better document structure. This list shows all HTML 5 tags in alphabetical order with description.

List of HTML 5 Tags

Tag	Description
<article>	This element is used to define an independent piece of content document, that may be a blog, a magazine or a newspaper article.
<aside>	It specifies that article is slightly related to the rest of the whole page.
<audio>	It is used to play audio file in HTML.
<bdi>	The bdi stands for bi-directional isolation. It isolates a part of text that is formatted in other direction from the outside text document.
<canvas>	It is used to draw canvas.
<data>	It provides machine readable version of its data.
<datalist>	It provides auto complete feature for text field.
<details>	It specifies the additional information or controls required by user.
<dialog>	It defines a window or a dialog box.
<figcaption>	It is used to define a caption for a <figure> element.
<figure>	It defines a self-contained content like photos, diagrams etc.
<footer>	It defines a footer for a section.
<header>	It defines a header for a section.
<main>	It defines the main content of a document.
<mark>	It specifies the marked or highlighted content.
<menuitem>	It defines a command that the user can invoke from a popup menu.
<meter>	It is used to measure the scalar value within a given range.
<nav>	It is used to define the navigation link in the document.
<progress>	It specifies the progress of the task.
<rp>	It defines what to show in browser that don't support ruby annotation.
<rt>	It defines an explanation/pronunciation of characters.
<ruby>	It defines ruby annotation along with <rp> and <rt>.
<section>	It defines a section in the document.
<summary>	It specifies a visible heading for <detailed> element.
<svg>	It is used to display shapes.
<time>	It is used to define a date/time.
<video>	It is used to play video file in HTML.
<wbr>	It defines a possible line break.

➢ HTML Forms-Controls

An **HTML form** is *a* section of a document which contains controls such as text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing such as name, email address, password, phone number, etc. .

Why use HTML Form

HTML forms are required if you want to collect some data from of the site visitor.

Tag	Description
<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

For example: If a user want to purchase some items on internet, he/she must fill the form such as shipping address and credit/debit card details so that item can be sent to the given address.

HTML Form Syntax

```
1. <form action="server url" method="get|post">
2.  //input controls e.g. textfield, textarea, radiobutton, button
3. </form>
```

HTML Form Tags

Let's see the list of HTML 5 form tags.

HTML 5 Form Tags

Let's see the list of HTML 5 form tags.

Tag	Description
<datalist>	It specifies a list of pre-defined options for input control.
<keygen>	It defines a key-pair generator field for forms.
<output>	It defines the result of a calculation.

HTML <form> element

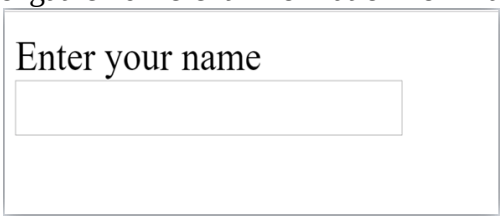
The HTML <form> element provide a document section to take input from user. It provides various interactive controls for submitting information to web server such as text field, text area, password field, etc.

Syntax:

```
1. <form>
2. //Form elements </form>
```

HTML <input> element

The HTML <input> element is fundamental form element. It is used to create form fields, to take input from user. We can apply different input filed to gather different information form user. Following is the example to show the simple text input.



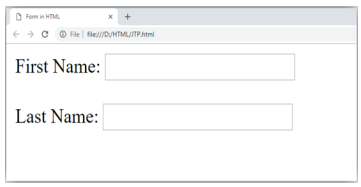
Example:

```
1. <body>
2.  <form>
3.  Enter your name <br>
4.  <input type="text" name="username"> </form>
5. </body>
```

Output:

HTML TextField Control

The type="text" attribute of input tag creates textfield control also known as single line textfield control. The name attribute is optional, but it is required for the server side component such as JSP, ASP, PHP etc.




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

Output:

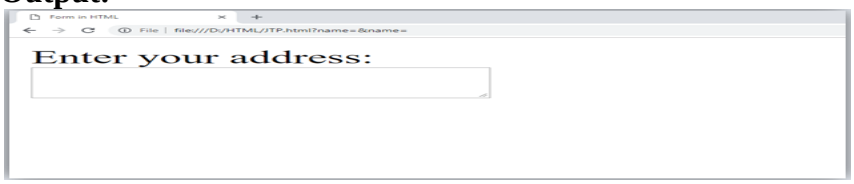
HTML <textarea> tag in form

The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> can be specify either using "rows" or "cols" attribute or by CSS.

Example:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.  <title>Form in HTML</title>
5. </head>
6. <body>
7.  <form>
8.  Enter your address:<br>
9.  <textarea rows="2" cols="20"></textarea>
10.  </form>
11. </body>
12. </html>
```

Output:



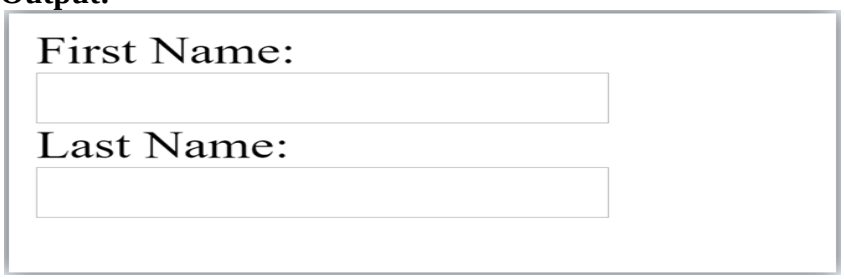
Label Tag in Form

It is considered better to have label in form. As it makes the code parser/browser/user friendly.

If you click on the label tag, it will focus on the text control. To do so, you need to have for attribute in label tag that must be same as id attribute of input tag.

```
1.  <form>
2.      <label for="firstname">First Name: </label> <br/>
3.          <input type="text" id="firstname" name="firstname"/> <br/>
4.     <label for="lastname">Last Name: </label>
5.          <input type="text" id="lastname" name="lastname"/> <br/>
6.   </form>
```

Output:



HTML Password Field Control

The password is not visible to the user in password field control.

```
1.  <form>
2.      <label for="password">Password: </label>
3.          <input type="password" id="password" name="password"/> <br/>
4.  </form>
```

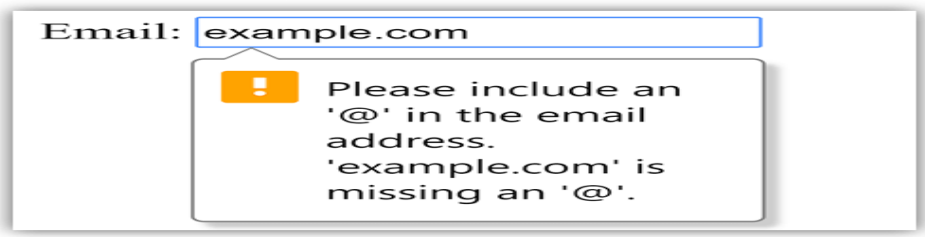
Output:

HTML 5 Email Field Control

The email field in new in HTML 5. It validates the text for correct email address. You must use @ and . in this field.

```
1.  <form>
2.      <label for="email">Email: </label>
3.          <input type="email" id="email" name="email"/> <br/>
4.  </form>
```

It will display in browser like below:



Radio Button Control

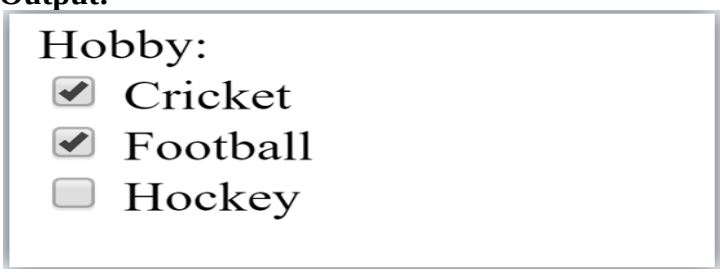
The radio button is used to select one option from multiple options. It is used for selection of gender, quiz questions etc.
If you use one name for all the radio buttons, only one radio button can be selected at a time.
Using radio buttons for multiple options, you can only choose a single option at a time.

```
<form>
        <label for="gender">Gender: </label>
         <input type="radio" id="gender" name="gender" value="male"/>Male
         <input type="radio" id="gender" name="gender" value="female"/>Female <br/>
</form>
```

Checkbox Control
The checkbox control is used to check multiple options from given checkboxes.

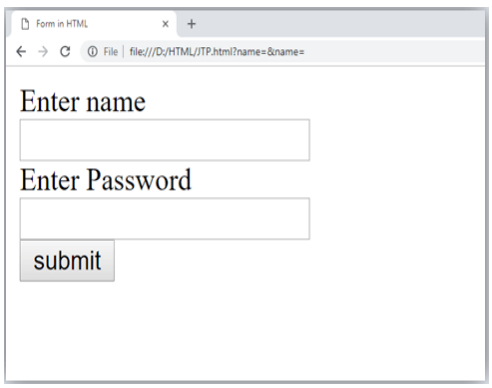
```
<form>
Hobby:<br>
        <input type="checkbox" id="cricket" name="cricket" value="cricket"/>
          <label for="cricket">Cricket</label> <br>
        <input type="checkbox" id="football" name="football" value="football"/>
          <label for="football">Football</label> <br>
        <input type="checkbox" id="hockey" name="hockey" value="hockey"/>
          <label for="hockey">Hockey</label>
</form>
```

Output:



Submit button control
HTML **<input type="submit">** are used to add a submit button on web page. When user clicks on submit button, then form get submit to the server.
Syntax:
<input type="submit" value="submit">
The type = submit , specifying that it is a submit button
The value attribute can be anything which we write on button on web page.
The name attribute can be omit here.
Example:

```
<form>
 <label for="name">Enter name</label><br>
 <input type="text" id="name" name="name"><br>
 <label for="pass">Enter Password</label><br>
 <input type="Password" id="pass" name="pass"><br
>
 <input type="submit" value="submit">
</form>
```



Output:
HTML <fieldset> element:
The <fieldset> element in HTML is used to group the related information of a form. This element is used with <legend> element which provide caption for the grouped elements.
Example:

```
<form>
```

```
    <fieldset>
     <legend>User Information:</legend>
    <label for="name">Enter name</label><br>
  <input type="text" id="name" name="name"><br>
  <label for="pass">Enter Password</label><br>
  <input type="Password" id="pass" name="pass"><br>
  <input type="submit" value="submit">
  </fieldset>
  lt;/form>
```

Output:

HTML Form Example

Following is the example for a simple form of registration.

```
<!DOCTYPE html>
<html>
<head>
<title>Form in HTML</title>
</head>
<body>
<h2>Registration form</h2>
<form>
<fieldset>
<legend>User personal information</legend>
<label>Enter your full name</label><br>
<input type="text" name="name"><br>
<label>Enter your email</label><br>
<input type="email" name="email"><br>
<label>Enter your password</label><br>
<input type="password" name="pass"><br>
<label>confirm your password</label><br>
<input type="password" name="pass"><br>
<br><label>Enter your gender</label><br>
<input type="radio" id="gender" name="gender" value="male"/>Male <br>
<input type="radio" id="gender" name="gender" value="female"/>Female <br/>
<input type="radio" id="gender" name="gender" value="others"/>others <br/>
<br>Enter your Address:<br>
<textarea></textarea><br>
<input type="submit" value="sign-up">
</fieldset>
</form>
</body>
</html>
```

Output
:

HTML Form Example
Let's see a simple example of creating HTML form.

```
1.  <form action="#">
2.  <table>
3.  <tr>
4.      <td class="tdLabel"><label for="register_name" class="label">Enter name:</label></td>

5.      <td><input type="text" name="name" value="" id="register_name" style="width:160px"/
></td>
6.  </tr>
7.  <tr>
8.      <td class="tdLabel"><label for="register_password" class="label">Enter password:</labe
l></td>
9.      <td><input type="password" name="password" id="register_password" style="width:160
px"/></td>
10. </tr>
11. <tr>
12.     <td class="tdLabel"><label for="register_email" class="label">Enter Email:</label></td
>
13.     <td
14. ><input type="email" name="email" value="" id="register_email" style="width:160px"/></t
d>
15. </tr>
16. <tr>
17.     <td class="tdLabel"><label for="register_gender" class="label">Enter Gender:</label></
td>
18.     <td>
19. <input type="radio" name="gender" id="register_gendermale" value="male"/>
20. <label for="register_gendermale">male</label>
21. <input type="radio" name="gender" id="register_genderfemale" value="female"/>
22. <label for="register_genderfemale">female</label>
23.     </td>
24. </tr>
25. <tr>
26.     <td class="tdLabel"><label for="register_country" class="label">Select Country:</label>
</td>
27.     <td><select name="country" id="register_country" style="width:160px">
28.     <option value="india">india</option>
```

```
29.     <option value="pakistan">pakistan</option>
30.     <option value="africa">africa</option>
31.     <option value="china">china</option>
32.     <option value="other">other</option>
33.  </select>
34.  </td>
35.  </tr>
36.  <tr>
37.     <td colspan="2"><div align="right"><input type="submit" id="register_0" value="registe
r"/>
38.  </div></td>
39.  </tr>
40.  </table>
41.  </form>
```

➢ HTML Audio Tag

HTML audio tag is used to define sounds such as music and other audio clips. Currently there are three supported file format for HTML 5 audio tag.

1. mp3
2. wav
3. ogg

HTML5 supports <video> and <audio> controls. The Flash, Silverlight and similar technologies are used to play the multimedia items.

This table defines that which web browser supports which audio file format.

Browser	mp3	wav	ogg
Internet Explorer	yes	no	no
Google Chrome	yes	yes	yes
Mozilla Firefox	yes*	yes	yes
Opera	no	yes	yes
Apple Safari	yes	yes	no

HTML Audio Tag Example

Let's see the code to play mp3 file using HTML audio tag.

```
1.  <audio controls>
2.    <source src="koyal.mp3" type="audio/mpeg">
3.  Your browser does not support the html audio tag.
4.  </audio>
5.        <audio controls>
6.          <source src="koyal.mp3" type="audio/mpeg">
7.        Your browser does not support the html audio tag.
8.        </audio>
```

9. Output:

koyal.mp3

Let's see the example to play ogg file using HTML audio tag.

```
1.  <audio controls>
2.    <source src="koyal.ogg" type="audio/ogg">
3.  Your browser does not support the html audio tag.
4.  </audio>
```

Supporting Browsers

Element	Chrome	IE	Firefox	Opera	Safari
<audio>	Yes	Yes	Yes	Yes	Yes

Attributes of HTML Audio Tag

There is given a list of HTML audio tag.

Attribute	Description
controls	It defines the audio controls which is displayed with play/pause buttons.
autoplay	It specifies that the audio will start playing as soon as it is ready.
loop	It specifies that the audio file will start over again, every time when completed.
muted	It is used to mute the audio output.
preload	It specifies the author view to upload audio file when the page loads.
src	It specifies the source URL of the audio file.

HTML Audio Tag Attribute Example

Here we are going to use controls, autoplay, loop and src attributes of HTML audio tag.

```
1. <audio controls autoplay loop>
2.   <source src="koyal.mp3" type="audio/mpeg"></audio>
```

MIME Types for HTML Audio format

The available MIME type HTML audio tag is given below.

Audio Format	MIME Type
mp3	audio/mpeg
ogg	audio/ogg
wav	audio/wav

➢ HTML Video Tag

HTML 5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.

Currently, there are three video formats supported for HTML video tag:

1. mp4
2. webM
3. ogg

Let's see the table that defines which web browser supports video file format.

Browser	mp4	webM	ogg
Internet Explorer	yes	no	no
Google Chrome	yes	yes	yes
Mozilla Firefox	yes	yes	yes
Opera	no	yes	yes
Apple Safari	yes	no	no

HTML Video Tag Example

Let's see the code to play mp4 file using HTML video tag.

```
1. <video controls>
2.   <source src="movie.mp4" type="video/mp4">
3.   Your browser does not support the html video tag.
4. </video>
```

Let's see the example to play ogg file using HTML video tag.

```
1. <video controls>
2.   <source src="movie.ogg" type="video/ogg">
3.   Your browser does not support the html video tag.
4. </video>
```

Supporting Browsers

Element	Chrome	IE	Firefox	Opera	Safari
<video>	Yes	Yes	Yes	Yes	Yes

Attributes of HTML Video Tag

Let's see the list of HTML 5 video tag attributes.

Attribute	Description
controls	It defines the video controls which is displayed with play/pause buttons.
height	It is used to set the height of the video player.
width	It is used to set the width of the video player.
poster	It specifies the image which is displayed on the screen when the video is not played.
autoplay	It specifies that the video will start playing as soon as it is ready.
loop	It specifies that the video file will start over again, every time when completed.
muted	It is used to mute the video output.
preload	It specifies the author view to upload video file when the page loads.
src	It specifies the source URL of the video file.

HTML Video Tag Attribute Example

Let's see the example of video tag in HTML where are using height, width, autoplay, controls and loop attributes.

```
<video width="320" height="240" controls autoplay loop>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the html video tag.
</video>
```

MIME Types for HTML Video format

The available MIME type HTML video tag is given below.

Video Format	MIME Type
mp4	video/mp4
ogg	video/ogg
webM	video/webM

```
<!DOCTYPE>
<html>
<head>
<style>
h1{
color:white;
background-color:red;
padding:5px;
}
        p{
        color:blue;
        }
        </style>
        </head>
        <body>
        <h1>Write Your First CSS Example</h1>
        <p>This is Paragraph.</p>
        </body>
        </html>
```

➢ CSS STYLING

Introduction:

- **Content and Style**
- **HTML Structure**
- **CSS Attachment to Pages**
- **CSS Main Facilities**
- **Browser Support**
- **CSS 1 Specification**

Content and Style:

Two aspects of any document are **content** and **style.**

The **content** gives the **information** to be presented

And the **style** defines how the information is presented. Most publishers have a House Style that is a consistent w presenting information.

HTML's main role is to **define content**. What is needed is the ability for publishers to have control of style. In the context of the Web, the publisher can be the organisation that owns the Web site but it can also be the person viewing the information. It is only by having a clear separation between content and style that this can be achieved. So in this Primer we shall use HTML in most of the examples as a means of defining the **content** and CSS to define the **style**.

CSS is a **World Wide Web Consortium (W3C**) Recommendation. That means that all the W3C Members(which includes all the main browser manufacturers) have agreed to support it in their products. That does not necessarily mean immediately as each has its own production schedule for enhancements to its products but long term all should support all of the features. CSS first became a W3CRecommendation in 1996 (called CSS 1) and there was a significant update in May 1998 .**HTML** allows a Web page to be laid out and there is some guidance to browsers as to how to represent each element. Thus, **h1** should be larger than **h2** and **em** should be emphasised by some method but
that is as far as it goes.

The aim of Cascading Style Sheets (CSS) is to give the page developer much more control on how apage should be displayed by all browsers.

A **style sheet** is a set of **rules** that controls the formatting of HTML elements on one or more Web pages. Thus, the appearance of a Web page can be changed by changing the style sheet associated with it. There is no need to make detailed changes within the Web page to change how it looks. advantages of using style sheets :

1. **accessibility**, different styling can be provided for different users dependent on their requirements.
2. Separating style and content is **good design** and will normally produce a better and more consistent web site. As one style sheet can be used for a whole web site, it normally means that the overall **size** of the web site is smaller and the downloads required for each page can be decreased by up to 40%.

An Allwoing browser to make overall decisions on styling often means that the rendering time by the browser is also shorter.

3. A Web page may have its own style sheet that refines the information in the common style sheet. Readers may define their own style sheet indicating their preferences. Thus style sheets **cascade** and decisions need to be made as to which style sheet is in control when there is a conflict.

A style sheetruleconsists of **two parts**. A **selector** that defines which HTML elements are controlled by the rule and a **declaration** that says what the required effect is.

Thus a simple rule is:

```
h1 { color: blue }
```

This says that all **h1** elements in a page should be displayed in blue. The selector is **h1**. **color** is the **property** that

is to be changed, **blue** is the **value** that the property is changed to and **color: blue** is thedeclaration.

HTML Structure

HTML elements in the body of the document fall into two main classes:

1. Block-level
2. Inline (sometimes called character-level)

Figure 1.1: Block level and Inline Elements

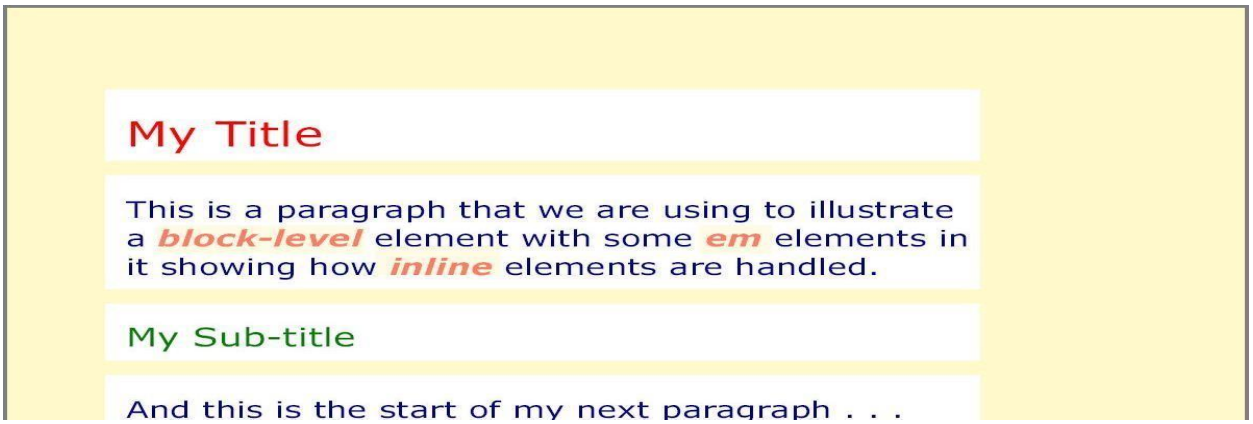


Figure 1.1: Block level and Inline Elements

Lists and tables are special types of elements that have their own unique styling.

When a **block-level element** is inserted into a document, it terminates the previous element and effectively start a new line. Some examples are **<p>**and **<h1>**. Inline elements do not terminate the previous element and form pa of the previous element; **em** is an example.

The **body** of an HTML page contains text marked-up using block-level elements (**h1, p, ul, div** etc). Each block level element has an area which contains the content. For a **p** element, this is the area that contains the whol paragraph while for **h1** it is often an area consisting of a single line of text. CSS provides facilities for controllin where that area is placed relative to other areas.

The **inline** elements often inherit the style of the block-level element that they are part of but can be given specia styling appropriate to the inline element.

For example:

```
<h1>My Title</h1>
<p>This is a paragraph that I am using to illustratea <em>block-level</em> element
with some <em>em</em> elements in it showing how <em>inline</em> elements are handled.</p>
<h2>My Sub-title</h2>
<p>And this is the start of my next paragraph . . .</p>
```

Figure 1.1 shows the areas covered by the block level elements and the sub-areas covered by the inline elements. The inline elements have been given their own background colour, colour, font-weight (bold) and font style (italic) while inheriting the text size and font from the enclosing paragraph.

CSS Attachment to Pages:

There are **four ways** of linking a style sheet to an HTML page:

1. By defining a link from the HTML page to the style sheet (normally stored in a **xxx.css** file). Thi allows you to have a single style sheet that changes the appearance of many Web pages.
2. By specifying an HTML **style** element in the **head** of the page. This allows you to define th appearance of a single page.
3. By adding inline styles to specific elements in the HTML file by the **style** attribute. This allows yo to change a single element or set of elements. This should only be used when absolutely necessary as it negate some of the advantages of having style sheets. It effectively over-rides

the overall style for the page. Also, it is likely that it will be removed from CSS 3.

4. By importing a style sheet stored externally into the current style. The **style** element for the page ca

consist of a set of imports plus some rules specific to the page.

Initially, while you are experimenting, the simplest method is to use the HTML **style** element to adda style shee to an HTML page. The **style** element, consisting of a set of rules, is placed in the document **head**. When you hav decided on your house style, it is likely that you will move to having the style sheet external to an individual pag and either linked in or imported.

Main Facilities

The facilities provided by style sheets are much as you would expect:

1. Better control of fonts including colour
2. Better control of block-level layout (indents, margins, alignment etc)
3. Better control of inline layout, particularly with regard to diagrams and related text

Browser Support

To achieve these effects, the Web browser has to know what to do with style sheets. Really old browsers will not have this capability but the recent offerings from Microsoft, Opera and Netscape have good support now and are committed to full support in the future. HTML Editors are beginning to have Style Sheet additions and separate Style Sheet Editors are appearing. Visit the W3C CSS Web Page [1]for all the latest information.

Given the way CSS has been implemented, it is possible to design your web pages so that they still present the HTML information even when CSS support is not there. So there is little excuse for not adding style to your web pages now even if you expect your pages are to be viewed by really old browsers.

CSS Specification

This Primer will not tell you everything about CSS. If you get to the position where you really need to know precisely what happens in some subtle situation, you need to consult the formal specification that can be found on the World-Wide Web Consortium's web site [2] [3].

To make it as easy as possible to relate the specification to the Primer, the order of presentation here is similar to the order in the Specification.

2. CSS Rules:

Introduction

Syntax for CSS Properties

Introduction

Let us assume initially that we wish to define a style sheet by adding a **style** element to a page. Thepage will look something like:

```
<html>
<style type="text/css">
h1 {font: 12pt/14pt "Palatino"; font-weight: bold; color: red} h2
{font: 10pt/12pt "Palatino"; font-weight: bold; color: blue}p
{font: 10pt/12pt "Times"; color: black}
</style>
<body>
...:
</body>
</html>
```

The **set of rules** are contained between **<style>**and **</style>**. Ignore the **type** attribute, for now. The reason for this will be come clearer later. For now let us concentrate on the rules. As stated in the introduction, each rule consists **of two parts**:
a **selector** that defines what HTML elements are controlled by the rule and a **declaration** that says what the required effect is. The selectors above define three rules that apply to all **h1** elements, all **h2** elements and all **p** elements respectively. We shall show how we can be both more selective (control a subset of **h1** elements) or control a set of elements (a block of different HTML elements) later. Initially let us look at the declarations. Each declaration consists of the name of a **propert**y and the value to be assigned to it. So the simplest style rule is:

```
selector { property: value }
```

To set several properties in a single rule requires the individual declarations to be separated bysemicolons:

```
selector { property1: value1; property2: value2; property3: value3 }
```

Syntax for CSS Properties:

The general format for each sub-section is for
the title to give a descriptive name followed immediately by the syntax definition (in BNF style) for theproperty declaration.
For example:

```
font-weight: <value>
```

This says that the name of the property is **font-weight** and **<value>**is assigned to it. In the CSS specification, the possible values of **<value>**are defined. In this Primer, sometimes only a sample of the possible values are given to give an idea of what is possible rather than exhaustively list them all.
In BNF, the notation could be written:
<font-weight> ::= font-weight: <value>
This style is used if, for example, the definition of <value> needs to be given separately. Some of the syntax definitions can be quite complicated. The notation used is asfollows:**Notation Meaning:**

Notation	Meaning
<Abc>	A value of type Abc. Some of the more common types are discussed later.
abc	A keyword that must appear exactly as written.
X Y Z	X must occur then Y then Z in that order.
X | Y	Xor Y must occur.
X || Y || Z	X, Y or Z or some combination in any order.
[Abc]	The square brackets are used to group items together.
ST*	ST is repeated zero or more times. ST can be a bracketed set of items.
ST+	ST is repeated one or more times.
ST?	ST is optional. It can either appear once or not at all.
ST {A,B}	ST must occur at least A times and at most B times.

3. Length, Percentage, Color and URLs:

- 3.1 Introduction
- 3.2 Length Values
- 3.3 Percentage values
- 3.4 Color Values
- 3.5 URLs
- 3.6 CSS Specification

Introduction

Four values (<length>, <percentage> , <color>, <url>) are used by many of the rules. They areexplained in this s

Length Values

<length> ::= [+ | -]? <number><unit>

A length value is formed by an optional + or -, followed by a number, followed by a two-letter abbreviation that indicates the unit (this can be left out if the value is 0). There are no spaces in the middle of a length value. Both relative and absolute length units are supported in CSS. Relative units give a length relative to another length property. The **relative length units** are:

Unit	Meaning
em	Historically width of letter **M**, but in CSS it equals the font height (if font is 12pt then 1em=12pt)
ex	x-height: the height of the letter **x** which varies between fonts (a 12pt Times Roman 'ex' will be bigger than a 12pt Baskerville 'ex')
px	pixels: in CSS the pixel is equal to one pixel on a normal computer display. So if the output is to go to a 1200dpi laser printer, the browser will take 1 pixel to mean about 12 laserprinter pixels.

The absolute length units are:

Notation	Meaning
in	inches
cm	centimetres
mm	millimetres
pt	Printing industry point; originally there was 72pt to a French inch which was larger than an English inch! Postscript and CSS define 1pt=1/72in
pc	picas; 1pc=12pt

Percentage Values:

<percentage> ::= [+ | -]? <number> %

A percentage value is formed by an optional + or -, followed by a number, followed by %. There are no spaces in a percentage value.

Percentage values are relative to other values, as defined for each property. Often the percentage value is relative t element's font size but it might be the width of the page.

Color Values:

<color> ::= <keyword> | # <hex><hex><hex> | # <hex><hex><hex><hex><hex><hex>

<hex> |

A color value is a keyword or a numerical RGB specification. The original set of keyword colours was aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white and yellow. These have been enhanced as a result of the requirements from other specifications. The set of keywords with the corresponding RGB values given in Appendix C are likely to be supported by a browser.

RGB colours can be defined in one of four ways:

1. #rrggbb(e.g., #00ff00)

2. #rgb (e.g., #0f0)

3. rgb(x,x,x) where x is an integer between 0 and 255

4. rgb(y%,y%,y%) where y is a number between 0.0 and 100.0

The following examples all specify the same colour:

pre { color: red }h1 { color: #foo }

p { color: #ff0000 }

h2 { color: rgb(255,0,0) }

h3 { color: rgb(100%, 0%, 0%) }

Values outside the range are clipped. So, for example, integer values have a maximum of 255 andeverything abov that will be changed to 255. Similarly 120% is interpreted as 100%.

URLs:

```
<url> ::= url( <whitespace>? [ ' | " ]? <characters in url> [ ' | " ]? <whitespace> )
```

A URL value is of the form:

```
url(xyz)
```

where xyz is the URL. The URL may be quoted with either single (') or double (") quotes and may have whitespace before the initial quote or after the final quote.

Parentheses, commas, spaces, single quotes, or double quotes in the URL must be escaped with a backslash.

Partial URLs are interpreted relative to the style sheet source, not to the HTML source.Some examples are:

```
body { background: url("circle.gif") }
```

4. Font Properties:

- 4.1 Font Family (Sets typeface)
- 4.2 Font Style (Italicises text)
- 4.3 Font Variant (Mostly upper case)
- 4.4 Font Weight (Sets thickness of type)
- 4.5 Font Size (Sets size of text)
- 4.6 Font (Shorthand)

Font Family (set typeface)

font-family: [<family-name> | <generic-family>] [, [<family-name> | <generic-family>]]*

<family-name> ::= serif | sans-serif | cursive | fantasy | monospace

The font to use is set by the font-family property. Instead of defining a single font, a sequence of fonts should be listed. The browser will use the first if it is available but try the second and so on. The value can either be a specific font name or a generic font family.

Ex:p { font-family: "New Century Schoolbook", Times, serif }

Any font name containing spaces (multiple words) must be quoted, with either single or double quotes. That i why the first font name is quoted but Times is not.

The first two requests are for specific fonts **New Century Schoolbook** and **Times**. As both are serif fonts, th generic font family listed as a backup is the **serif** font family. In this case, Times will be used if New Centur Schoolbook is not available, and a serif font if Times is not available.

Some example fonts are:

serif

Times New Roman, Bodini, Garamond

sans-serif

Trebuchet, Arial, Verdana, Futura, Gill Sans, Helvetica

cursive

Poetica, Zapf-Chancery, Roundhand, Script

fantasy

Critter, Cottonwood

monospace
Courier, Courier New, Prestige, Everson Mono
See also the **font** property which lets the user define a set of font properties in a single command..

Font Style (italicise text):

```
font-style: normal | italic | oblique
```

The font-style property has one of three values: **normal**, **italic** or **oblique** (slanted). A sample stylesheet might be

```
h3 { font-style: italic } p
{ font-style: normal }
```

Italic and oblique are similar. ***Italic*** is normally a separate font while ***oblique*** often bends the normalfont.

Font Variant (size of upper case):

```
font-variant: normal | small-caps
```

Normally, **upper case characters** are much larger than the lower-case characters. **Small-caps are** often used when all the letters of the word are in capitals. In this case the upper case characters are onlyslightly larger than the lowercase ones. An example is:

```
p { font-variant: small-caps }
```

For example, the text **Elephant** with font-variant set to **small-caps** would appear as ELEPHANT.

Font Weight (set thickness of type):

```
font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |
```

The font-weight property sets the weight of the font to values like **normal** and **bold**. It also has values **bolder** and **lighter** which are relative to any inherited font -weight. Block-level elements are displayed with default values inherited from the **body** or **div** element that surrounds it. It is also possible to define absolute font-weights. Values range from 100 to 900 with **normal** being 400 and bold being **700** approximately. If that much control is not possible, some of the weights may be grouped together (for example, 100, 200 and 300 may all look the same) and if the specified weight is not available, an alternative value will be chosen. Some examples are:

```
h1 { font-weight: 800 }
h2 { font-weight: bold }
h3 { font-weight: 500 }
```

Font Size (set size of text):

```
font-size: <absolute-size> | <relative-size> | <length> | <percentage>

<absolute-size> ::= xx-small | x-small | small | medium | large | x-large | xx-large
```

The **font-size property sets** the size of the displayed characters. The absolute-size values are **small**, **medium** and **large** with additional possibilities like **x-small** (extra small) and **xx-small**. Relative sizes are **smaller** and **larger**. They are relative to any inherited value for the property. The length value defines the precise height in units like **12pt** or **1in**. Percentage values are relative to the inherited value.

Some examples are:

```
h1 { font-size: large }
h2 { font-size: 12pt }
h3 { font-size: 5cm }
p { font-size: 10pt } li
{ font-size: 150% }
```

The guidance is that the medium font should be around 10pt so, in this case **li** and **em** might be similarin size.

Font (Shorthand):

```
font: [ <font-style> || <font-variant> ||
```

The **font property may** be used as a shorthand for the various font properties, as well as the line-height. For example:

```
p { font: italic bold 12pt/14pt Times, serif }
```

specifies paragraphs with a bold and italic Times or serif font with a size of 12 points and a line height of14 point Note: The order of attributes is significant: the font weight and style must be specified before the others.

5. Color and Background Properties:

- 5.1 Color
- 5.2 Background Color
- 5.3 Background Image
- 5.4 Background Repeat
- 5.5 Background Attachment
- 5.6 Background Position
- 5.7 Background (Sets background images or color)

Color

```
color: <color>
```

The color property sets the foreground colour of a text element. For example:

```
h1 {color: blue }

h2 {color: rgb(255,0,0) }
```

Background Color:

```
background-color: <color> | transparent
```

The **background-color property** sets the background colour of an element. For example:

```
body { background-color: white }
h1 { background-color: #000080 }
```

The value **transparent** indicates that whatever is behind the element can be seen. For example if the background to the **body** element was specified as being red, a block-level element with **backgroundcolor** set to **transparent** would appear with a red background.

Background Image:

```
background-image: <url> | none
```

The **background-image** property sets the background image of an element. For example:

```
body { background-image: url("/images/monkey.gif") }

p { background-image: url("http://www.cclrc.com/pretty.png") }
```

When a background image is defined, a compatible background colour should also be defined for thoseusers who do not load the image or to cover the case when it is unavailable. If parts of the image are transparent, the background colour will fill these parts.
For example:

```
<ul style="background-image:url('wall.png')">

<li>An item</li>

<li>Another</li>
```

Figure 5.1 shows what might be the result depending on the font-size and other properties specified for the list.

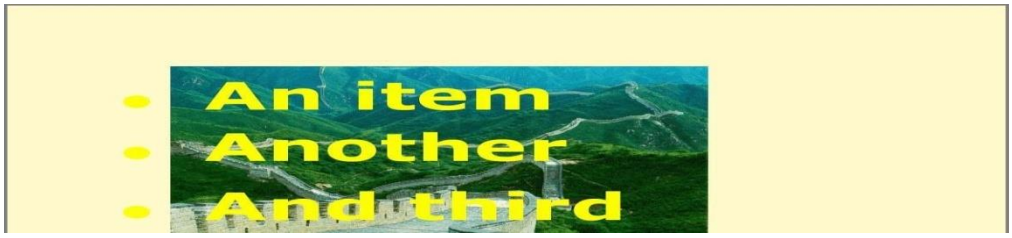


Figure 5.1: List with background-imageBackground Repeat:

```
background-repeat: repeat | repeat-x | repeat-y | no-repeat
```

If the background image does not fill the whole element area, this description specifies how it is repeated. The **repeat-x** value will repeat the image horizontally while the **repeat-y** value will repeat the image vertically. Setting **background-image** to **repeat** will repeat the image in both x and y directions. For example:

```
body { background-image: url(/images/monkey.gif) }
body { background-repeat: repeat-x }
```

In the above example, the monkey will only be tiled horizontallya as shown in Figure 5.2.

Figure 5.2: Monkey tiled horizontally

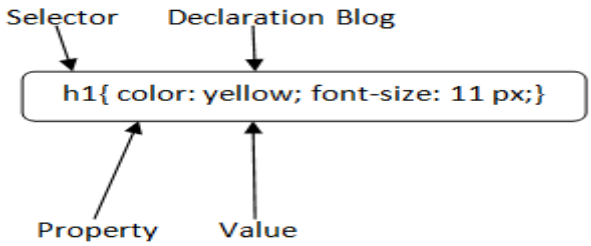
Background

Attachment: background-

attachment: scroll | fixed

CSS Syntax

A CSS rule set contains a selector and a declaration block.



Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

1. Selector{Property1: value1; Property2: value2; .....;}

CSS Selector

CSS selectors are used *to select the content you want to style*. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector
5. CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <style>
5. p{
6.     text-align: center;
7.     color: blue;
8. }
9. </style>
10. </head>
```

```
11. <body>
12. <p>This style will be applied on every paragraph.</p>
13. <p id="para1">Me too!</p>
14. <p>And me!</p>
15. </body>
16. </html>
```

Output:

This style will be applied on every paragraph.
Me too!
And me!

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.
It is written with the hash character (#), followed by the id of the element.
Let?s take an example with the id "para1".

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  #para1 {
6.      text-align: center;
7.      color: blue;
8.  }
9.  </style>
10. </head>
11. <body>
12. <p id="para1">Hello Javatpoint.com</p>
13. <p>This paragraph will not be affected.</p>
14. </body>
15. </html>
```

Output:

Hello Javatpoint.com
This paragraph will not be affected.

3) CSS Class Selector

The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.
Let's take an example with a class "center".

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  .center {
6.      text-align: center;
7.      color: blue;
8.  }
9.  </style>
```

```
10. </head>
11. <body>
12. <h1 class="center">This heading is blue and center-aligned.</h1>
13. <p class="center">This paragraph is blue and center-aligned.</p>
14. </body>
15. </html>
```

Output:

This heading is blue and center-aligned.

This paragraph is blue and center-aligned.

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

Let's see an example.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  p.center {
6.      text-align: center;
7.      color: blue;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h1 class="center">This heading is not affected</h1>
13. <p class="center">This paragraph is blue and center-aligned.</p>
14. </body>
15. </html>
```

Output:

This heading is not affected

This paragraph is blue and center-aligned.

4) CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  * {
6.    color: green;
7.    font-size: 20px;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h2>This is heading</h2>
13. <p>This style will be applied on every paragraph.</p>
```

```
14. <p id="para1">Me too!</p>
15. <p>And me!</p>
16. </body>
17. </html>
```

Output:

This is heading

This style will be applied on every paragraph.

Me too!

And me!

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Let's see the CSS code without group selector.

```
1.  h1 {
2.      text-align: center;
3.      color: blue;
4.  }
5.  h2 {
6.      text-align: center;
7.      color: blue;
8.  }
9.  p {
10.     text-align: center;
11.     color: blue;
12. }
```

As you can see, you need to define CSS properties for all the elements. It can be grouped in following ways:

```
1.  h1,h2,p {
2.      text-align: center;
3.      color: blue;
4.  }
```

Let's see the full example of CSS group selector.

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <style>
5.  h1, h2, p {
6.      text-align: center;
7.      color: blue;
8.  }
9.  </style>
10. </head>
11. <body>
12. <h1>Hello Javatpoint.com</h1>
13. <h2>Hello Javatpoint.com (In smaller font)</h2>
14. <p>This is a paragraph.</p>
```

15. </body>
16. </html>

Output:

Hello Javatpoint.com

Hello Javatpoint.com (In smaller font)

This is a paragraph.

How to add CSS

CSS is added to HTML pages to format the document according to information in the style sheet. There are three ways to insert CSS in HTML documents.

1. Inline CSS
2. Internal CSS
3. External CSS

1) Inline CSS

Inline CSS is used to apply CSS on a single line or element.

For example:

```
<p style="color:blue">Hello CSS</p>
```

For more visit here: Inline CSS

2) Internal CSS

Internal CSS is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.

For example:

```
<style>
p{color: blue}
</style>
```

For more visit here: Internal CSS

3) External CSS

External CSS is used to apply CSS on multiple pages or all pages. Here, we write all the CSS code in a css file. Its extension must be .css for example style.css.

For example:

```
p{color:blue}
```

You need to link this style.css file to your html pages like this:

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

Java script :-

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

Advantages of Java Script :-

- **Form validation:-** Checking the form errors Is called form validation
- **Popup ads (small window):-** the ads that will open automatically depending upon the user action
- **Dynamic pages:-** user can modify the content of the webpage at runtime.
    - (or)
    - A webpage which consist updated information

- **Less server interaction** – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** – They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Control Structures

The control structures within Javascript allow the program flow to change within a unit of code or function. These statements can determine whether or not given statements are executed, as well as repeated execution of a block of code.

Contents

1 if
2 while
3 do... while
4 for
5 switch

1. If...Else Statements

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

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

(i) If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
 {
        code to be executed if condition is true
 }
```

Note that if is written in lowercase letters. Using uppercase letters (**IF**) will generate a JavaScript error!

(ii) If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
 {
        code to be executed if condition is true
 }
else
 {
        code to be executed if condition is not true
 }
```

(iii) If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

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

2. Loops

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- Often when you write code, you want the same block of code to run over and over again in a row.

  Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

  In JavaScript, there are three different kind of loops:

(i) While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

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

Syntax

```
intialization
while (condition)
{
code to be executed
increment or decrement
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<script type="text/javascript">
    var i=0;
    while (i<=5)
    {
        document.write("The number is " + i);
        document.write("<br />");
        i++;
    }
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

(ii). do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
intialization
do
{
    code to be executed
    increment or decrement
}
while (condition);
```

Example

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

```
<script type="text/javascript">
    var i=0;
    do
    {
    document.write("The number is " + i);
    document.write("<br />");
    i++;
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

```
        }
while (i<=5);
</script>
```

(ii) The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (intialization;condition;increment or decrement)
{
    code to be executed
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example

```
<script type="text/javascript">
    var i=0;
    for (i=0;i<=5;i++)
    {
        document.write("The number is " + i);
        document.write("<br />");
    }
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

1. **Switch Statement**

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

Syntax

```
switch(variablename)
{
    case 1:
        execute code block 1
        break;
    case 2:
        execute code block 2
        break;
    default:
        code to be executed if variable value  is different

        from case 1 and 2
}
```

Break and Continue Statements

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
 if (i==5)
 break;
 document.write("The number is " + i);
 document.write("<br />");

}
 document.write("Hello");
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
Hello
```

The continue Statement

The continue statement will break the current condition and continue with the next value.

Example

```
<script type="text/javascript">
        var i=0
        for (i=0;i<=10;i++)
        {
                if (i==3)
                {
                        continue;
                }
                document.write("The number is " + i);
                document.write("<br />");
         }
</script>
```

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10
```

JavaScript Functions

- To keep the browser from executing a script when the page loads, you can put your script into a function.
- A function contains code that will be executed by an event or by a call to the function.
- You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).
- Whenever we call a function immediately it jumps to implementation part of the specififed function.
- After execution of the specified function code the next statement of the javascript code can be executed.

1. Functions (Without parameters)

- In this situation we call a method without passing any parameters.
- **Rules:** Call a function
  - Write implementation part for the function
  - When we call a function immediately it jumps to the implementation part of the function
  - After execution of the function. The next statement of the called function will be executed.

Syntax:

functionname(); //Function Calling

function functionname()
{
    some code; // Implementation part
}

When we call function immediately it jumps to implementation part of the function.

```
<script type="text/javascript">
muni();
document.write("hello");
function muni()
{
  document.write("Function without parameter");
}
</script>
```

Output:

Function without parameters
hello

2. Functions (With parameters)

- In this situation we call a method with parameters.
- **Rules:** Call a function
  - Write implementation part for the function
  - Declare variables to hold the values of called function.
  - When we call a function immediately it jumps to the implementation part of the function
  - After execution of the function. The next statement of the called function will be executed.

functionname(parameter1,parameter2,…..); //Function Calling

function functionname(variablename1,variablename2,…) // Implementation part
{
    some code;
}

```
<script type="text/javascript">
muni(4);
```

```
document.write("<br/>"+"hello");
function muni(a)
{
  document.write("result is"+a*a);
}
</script>
```

output

result is16
hello

3. The return Statement

- The return statement is used to specify the value that is returned from the function.
- So, functions that are going to return a value must use the return statement.
- The example below returns the product of two numbers (a and b):

Example

```
 <script type="text/javascript">
var x=muni(4,3);
document.write("Result is"+x);
function muni(a,b)
{
        return a*b;
}
</script>
```

output

result is 12

Global Functions

- JavaScript provides seven functions that are available globally in a JavaScript.
- The JavaScript global functions can be used with all the built-in JavaScript objects.

1. **eval()** Function

The **eval()** function evaluates or executes an argument.

If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements.

```
var x = 10;
var y = 20;
var a = eval("x * y") + "<br>";
var b = eval("2 + 2") + "<br>";
var c = eval("x + 17") + "<br>";

var res = a + b + c;
```

The result of *res* will be:
200
4
27

2. **isFinite()** Function

The **isFinite()** function determines whether a number is a finite, legal number.

This function returns false if the value is +infinity, -infinity, or NaN (Not-a-Number), otherwise it returns true.

```
var a = isFinite(123) + "<br>";
var b = isFinite(-1.23) + "<br>";
var c = isFinite(5-2) + "<br>";
var d = isFinite(0) + "<br>";
var e = isFinite("Hello") + "<br>";
var f = isFinite("2005/12/12") + "<br>";

var res = a + b + c + d + e + f;
```

The result of *res* will be:
true
true
true
true
false
false

3. isNaN() Function

The isNaN() function determines whether a value is an illegal number (Not-a-Number).

This function returns true if the value is NaN, and false if not.

```
var a = isNaN(123) + "<br>";
var b = isNaN(-1.23) + "<br>";
var c = isNaN(5-2) + "<br>";
var d = isNaN(0) + "<br>";
var e = isNaN("Hello") + "<br>";
var f = isNaN("2005/12/12") + "<br>";

var res = a + b + c + d + e + f;
```

The result of *res* will be:
false
false
false
false
true
true

4.**unescape()** Function

The unescape() function decodes an encoded string.

Encode and decode a string:

```
var str = "m? Visit W3Schools!";
var str_esc = escape(str);
document.write(str_esc + "<br>")
document.write(unescape(str_esc))
```

The output of the code above will be:
Need%20tips%3F%20Visit%20W3Schools%21
Need tips? Visit W3Schools!

Math Object

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical properties and methods.

Syntax for using properties/methods of Math:

Math.property;

Math.method();

Math object properties.

JavaScript provides eight mathematical constants that can be accessed from the Math object. [Note: By convention, the names of these constants are written in all uppercase letters.]

Constant	Description	Value
`Math.E`	Euler's constant.	Approximately 2.718.
`Math.LN2`	Natural logarithm of 2.	Approximately 0.693.
`Math.LN10`	Natural logarithm of 10.	Approximately 2.302.
`Math.LOG2E`	Base 2 logarithm of Euler's constant.	Approximately 1.442.
`Math.LOG10E`	Base 10 logarithm of Euler's constant.	Approximately 0.434.
`Math.PI`	π—the ratio of a circle's circumference to its diameter.	Approximately 3.141592653589793.
`Math.SQRT1_2`	Square root of 0.5.	Approximately 0.707.
`Math.SQRT2`	Square root of 2.0.	Approximately 1.414.

Math object Methods.

JavaScript provides mathematical methods that can be accessed from the Math object.

Method	Description	Example

`abs( x )`	absolute value of `x`	`abs( 7.2 )` is `7.2`
		`abs( 0.0 )` is `0.0`
		`abs( -5.6 )` is `5.6`

`ceil( x )`	rounds `x` to the smallest integer not less than `x`	`ceil( 9.2 )` is `10.0` `ceil( -9.8 )` is `-9.0`
`cos( x )`	trigonometric cosine of `x` (`x` in radians)	`cos( 0.0 )` is `1.0`
`exp( x )`	exponential method e^x	`exp( 1.0 )` is `2.71828` `exp( 2.0 )` is `7.38906`
`floor( x )`	rounds `x` to the largest integer not greater than `x`	`floor( 9.2 )` is `9.0` `floor( -9.8 )` is `-10.0`
`log( x )`	natural logarithm of `x` (base *e*)	`log( 2.718282 )` is `1.0` `log( 7.389056 )` is `2.0`

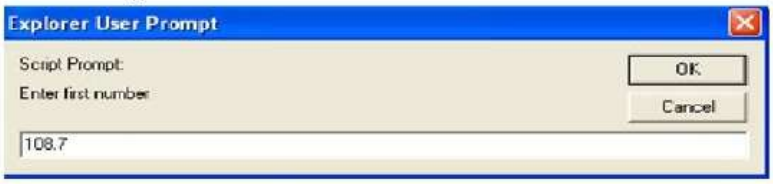
`max( x, y )`	larger value of `x` and `y`	`max( 2.3, 12.7 )` is `12.7` `max( -2.3, -12.7 )` is `-2.3`
`min( x, y )`	smaller value of `x` and `y`	`min( 2.3, 12.7 )` is `2.3` `min( -2.3, -12.7 )` is `-12.7`
`pow( x, y )`	`x` raised to power `y` (x^y)	`pow( 2.0, 7.0 )` is `128.0` `pow( 9.0, .5 )` is `3.0`
`round( x )`	rounds `x` to the closest integer	`round( 9.75 )` is `10` `round( 9.25 )` is `9`
`sin( x )`	trigonometric sine of `x` (`x` in radians)	`sin( 0.0 )` is `0.0`
`sqrt( x )`	square root of `x`	`sqrt( 900.0 )` is `30.0` `sqrt( 9.0 )` is `3.0`
`tan( x )`	trigonometric tangent of `x` (`x` in radians)	`tan( 0.0 )` is `0.0`

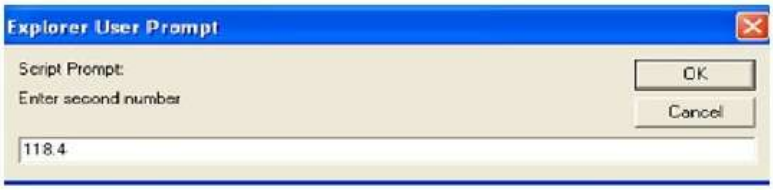
Math object Example

The script in our next example (Fig. 10.3) uses a programmer-defined function called **maximum to determine and return the largest of three floating-point values.**

```
<script type="text/javascript">
        var x,y,z,r;
        x=parseFloat(prompt("Enter first number","0"));
        y=parseFloat(prompt("Enter second number","0"));
        z=parseFloat(prompt("Enter third number","0"));
        r=maximum(x,y,z);
        document.write( "First number: " + x +"<br />Second number: " + y +
                        "<br />Third number: " + z +"<br />Maximum is: " + r );

        function maximum(x,y,z)
        {
          return Math.max(x, Math.max(y, z));
        }
</script>
```





First number: 108.7
Second number: 118.4
Third number: 118.9
Maximum is: 118.9

String Object

- A string is a series of characters treated as a single unit. A string may include letters, digits and various special characters, such as **+, -, *, /, $ and others.**

Methods of the String Object

- The **String object provides many methods (behaviors) for selecting characters from a string,**
- combining strings (called concatenation), obtaining substrings of a string, searching for substrings within a string, tokenizing a string and converting strings to all uppercase or lowercase letters.

1. charAt() Method Returns the character at the specified index (position)

Syntax: string.charAt(index)

Example: Return the first character of a string:

```
var str = "HELLO WORLD";
var res = str.charAt(0)
```

The result of res will be:

H

2. charCodeAt() Method Returns the Unicode of the character at the specified index

Syntax: string.charCodeAt(index)

Example: Return the Unicode of the first character in a string (the Unicode value for "H"):

```
var str = "HELLO WORLD";
var n = str.charCodeAt(0);
```

The result of n will be:

72

3. concat() Method Joins two or more strings, and returns a new joined strings

Syntax: string.concat(string1, string2, ..., stringX)

Example: Join two strings:

```
var str1 = "Hello ";
var str2 = "world!";
var res = str1.concat(str2);
```

The result of res will be:

Hello world!

4. indexOf() Method The indexOf() method returns the position of the first occurrence of a specified value in a string.

Syntax: string.indexOf(searchvalue, start)

Parameter	Description
Searchvalue	Required. The string to search for
Start	Optional. Default 0. At which position to start the search

Example1.: Find the first occurrence of the letter "e" in a string:

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("e");
```

The result of n will be:

1

5. lastIndexOf() Method The lastIndexOf() method returns the position of the last occurrence of a specified value in a string.

Syntax:string.lastIndexOf(searchvalue,start)

Parameter	Description
searchvalue	Required. The string to search for
start	Optional. The position where to start the search (searching backwards).

Example1: Search a string for the last occurrence of "planet":

```
var str = "Hello planet earth, you are a great p  lanet.";
var n = str.lastIndexOf("planet");
```

The result of n will be:

36

6. match() Method Searches a string for a match against a regular expression, and returns the matches

Syntax: string.match(regexp)

Example1:: Search a string for "ain":

```
var str = "The rain in SPAIN stays mainly in the plain";
var res = str.match(/ain/g);     where g means (global search)
```

The result of res will be an array with the values:

ain,ain,ain

- In the above example we have uppercase letters AIN that is not displayed on the screen.

Note: To search one (or) more uppercase letters use /searchtext/gi

7. replace() Method The replace() method searches a string for a specified value, or a regular expressio and returns a new string where the specified values are replaced.

Syntax: string.replace(searchvalue,newvalue)

Parameter	Description
searchvalue	Required. The value, or regular expression, that will be replaced by the new value
newvalue	Required. The value to replace the searchvalue with

Example1: Return a string where "Microsoft" is replaced with "SKIIMS":

```
var str = "Visit Microsoft!";
var res = str.replace("Microsoft", "SKIIMS");
```

The result of res will be:

Visit SKIIMS!

8. search() Method Searches a string for a specified value, or regular expression, and returns the **position** of the match

Syntax: string.search(searchvalue)

Parameter	Description
searchvalue	Required. A regular expression. A string will automatically be converted to a regular expression.

Example: Search for "W3Schools":

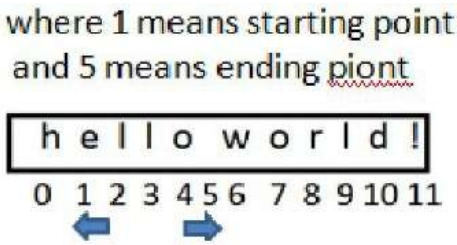
```
var str = "Visit W3Schools!";
var n = str.search("W3Schools");
```

The result of n will be:

9. slice() method The slice() method extracts parts of a string and returns the extracted parts in a new string.

Syntax: string.slice(start,end)

Example: Extract parts of a string:




```
var str = "Hello world!";

var res = str.slice(1,5);
```

The result of res will be:

ello

10. split() Method The split() method is used to split a string into an array of substrings, and returns the new array.

Syntax: string.split(separator,limit)

Parameter	Description
separator	Optional. Specifies the character, or the regular expression, to use for splitting the string. If omitted, the entire string will be returned (an array with only one item)
limit	Optional. An integer that specifies the number of splits, items after the split limit will not be included in the array

Example1: Separate each charater by **comma ,** including white-space:

```
var str = "How are you doing today?";
var res = str.split("");
```

{to get comma for each letter mention "" }

The result of res will be an array with the values:

H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?

11. substr() Method This method extracts parts of a string, beginning at the character at the specified position, and returns the specified number of characters.

Syntax: string.substr(start,length)

(i) start Required. The position where to start the extraction. First character is at index 0

(ii) length Optional. The number of characters to extract. If omitted, it extracts the rest of the string

Example1: Extract parts of a string:

```
var str = "Hello world!";
var res = str.substr(1, 4)
```

The result of res will be:

ello

12. toLowerCase() Method

Example: Convert the string to lowercase letters:

```
var str = "Hello World!";
var res = str.toLowerCase();
```

The result of res will be:

hello world!

13. toUpperCase() Method

Example: Convert the string to uppercase letters:

```
var str = "Hello World!";
var res = str.toUpperCase();
```

The result of res will be:

HELLO WORLD!

14. trim() Method : The trim() method removes whitespace from both sides of a string.

Note: Some browsers does not support trim() method.

Syntax: string.trim()

Example: Remove whitespace from both sides of a string:

```
var str = "      Hello World!      ";
alert(str.trim());
```

The alert box will display:

Hello World!

Date Object

The Date object is used to work with dates and times.

(i) Create a Date Object

The Date object is used to work with dates and times.

Date objects are created with the Date() constructor.

Some examples of initiating a date:

```
var today = new Date()
var d1 = new Date("October 13, 1975 11:13:00")
var d2 = new Date(79,5,24)
var d3 = new Date(79,5,24,11,33,0)
```

Date getMethods (How to display date and time)

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Date Set Methods

Set methods are used for setting a part of a date.

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

```
<script type="text/javascript">
var d=new Date();
d.setDate(18);
document.write("day is"+d.getDate()+"<br/>");
d.setMonth(4);
document.write("month is"+d.getMonth()+"<br/>");
d.setFullYear(2018);
document.write("year is"+d.getFullYear()+"<br/>");
d.setHours(11);
document.write("Hour is"+d.getHours()+"<br/>");
d.setMinutes(55);
document.write("Minutes is"+d.getMinutes()+"<br/>");
d.setSeconds(10);
document.write("Seconds is"+d.getSeconds()+"<br/>");
</script>
```

day is18
month is4
year is2018
Hour is11
Minutes is55
Seconds is10

Array Object

The Array object is used to store multiple values in a single variable.

What is an Array?

- An array is a special variable, which can hold more than one value, at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1="Saab";
var car2="Volvo";
var car3="BMW";
```

Create an Array

An array can be defined in three ways.

The following code creates an Array object called myCars:

Syntax 1:

```
var arrayname=new Array();
```

Store data into Array

We can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

```
arrayname[0]=value or string;
arrayname [1]= value or string;
arrayname [2]= value or string;
```

Display data from Array

We can display the array elements by using arrayname with index position.

Syntax: document.write(arrayname[indexposition]);

Examples:

Array Creation, Storing and Displaying Example

```
<script type="text/javascript">
var a=new Array();
a[0]=10;
a[1]=20;
a[2]=30;
a[3]=40;
a[4]=50;
document.write(a[0] + "<br/>");
document.write(a[1] + "<br/>");
document.write(a[2] + "<br/>");
document.write(a[3] + "<br/>");
document.write(a[4] + "<br/>");
</script>
```

10
20
30
40
50

document object

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

window.document
or
document

Methods of document object

We can access and change the contents of document by its methods.
The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.

document.getElementById():

The **document.getElementById()** method returns the element of specified id.

In the previous page, we have used**document.form1.name.value** to get the value of the input value. Instead of this, we can use document.getElementById() method to get value of the input text. But we need to define id for the input field.

Syntax: document.getElementBy("id name").value

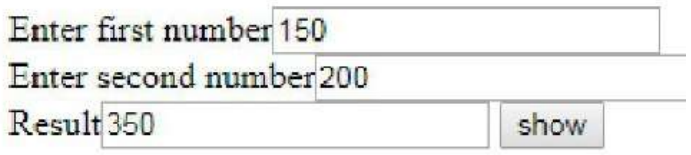
```
Enter name<input type="text" id="i" />
<input type="button" value="show" onclick="muni();" />

<script type="text/javascript">
 function muni()
 {
  var r=document.getElementById("i").value;
   alert("Your name is"+r);
  }
</script>
```

Addition of two numbers using getElementById() method

```
Enter first number<input type="text" id="f" /> <br/>
Enter second number<input type="text" id="s" /> <br/>
Result<input type="text" id="r" />
<input type="button" value="show" onclick="muni();" />

<script type="text/javascript">
 function muni()
 {
  var num1=parseInt(document.getElementById("f").value);
  var num2=parseInt(document.getElementById("s").value);
  var res=num1+num2;
  document.getElementById("r").value=res;
  }
</script>
```



document.getElementsByName()

The **document.getElementsByName()** method returns all the element of specified name.

The syntax of the getElementsByName() method is given below:

Syntax: document.getElementsByName("name")

```
<script type="text/javascript">
function totalelements()
```

```
{
var allgenders=document.getElementsByName("gender");
alert("Total Genders:"+allgenders.length);
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">

<input type="button" onclick="totalelements()" value="Total Genders">
</form>
```

document.getElementsByTagName()

The **document.getElementsByTagName()** method returns all the element of specified tag name.

The syntax of the getElementsByTagName() method is given below:

document.getElementsByTagName("name")

In this example, we going to count total number of paragraphs used in the document. To do this, we have called the document.getElementsByTagName("p") method that **returns the total paragraphs.**

```
<script type="text/javascript">
function countpara()
{
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</
p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```

document object properties

(i) innerText

The **innerText** property can be used to write the dynamic text on the html document.

This property does not support the html tag elements by assigning the values.

It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Example

```
<div id="chan">Hello </div>
<input type="button" value="click me" onclick="change();"/>
<script type="text/javascript">
function change()
{
```

AJAX

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies ike JavaScript, DOM, XML, HTML, CSS etc.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable lata from the client side is routed to the server side. It makes your application interactive and faster.

Ajax Applications (Where it is used)

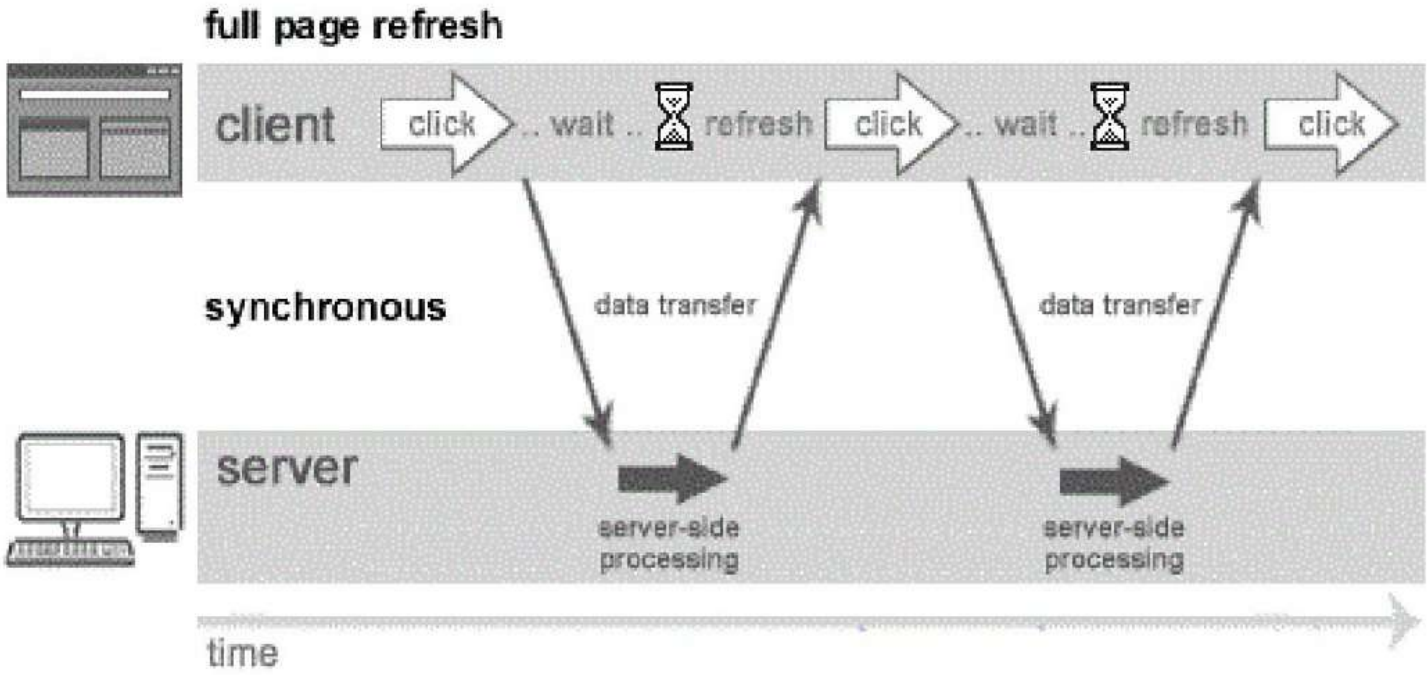
There are too many web applications running on the web that are using ajax technology like **gmail**, **acebook**,**twitter**, **google map**, **youtube** etc.

Synchronous vs Asynchronous

Before understanding AJAX, let's understand classic web application model and ajax web application nodel first.

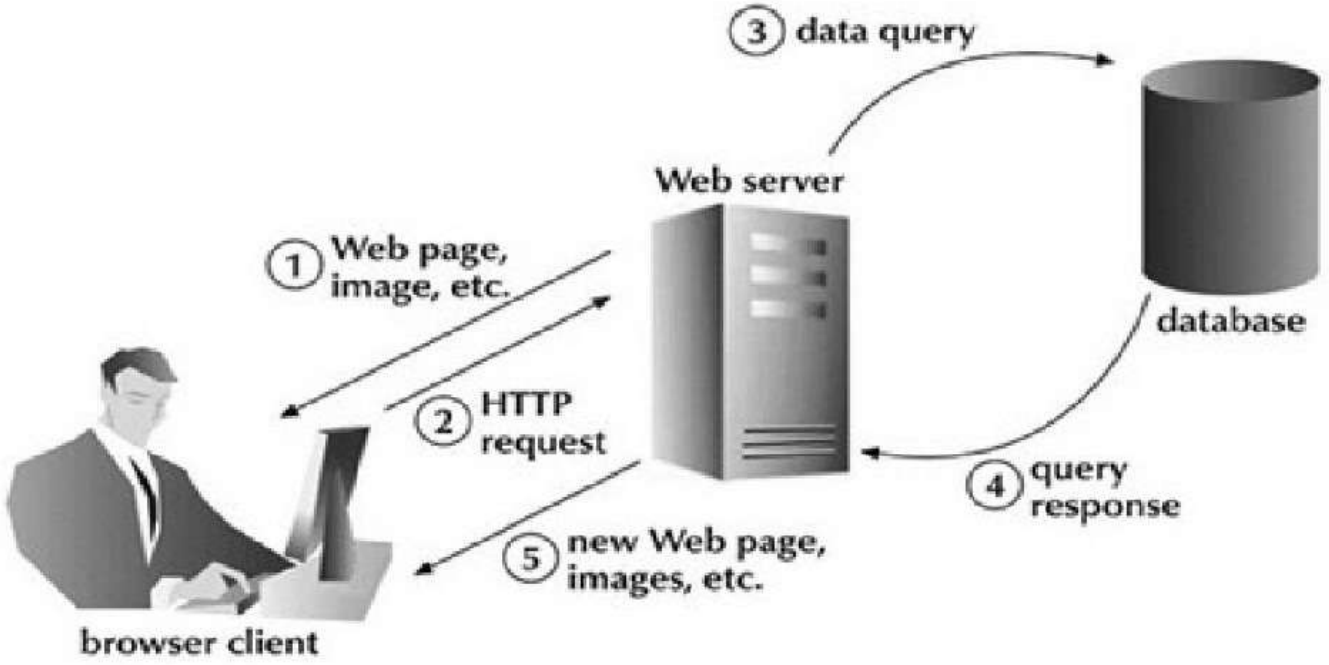
Synchronous (Classic Web-Application Model)

A synchronous request blocks the client until operation completes i.e. browser is not unresponsive. In such ase, javascript engine of the browser is blocked.



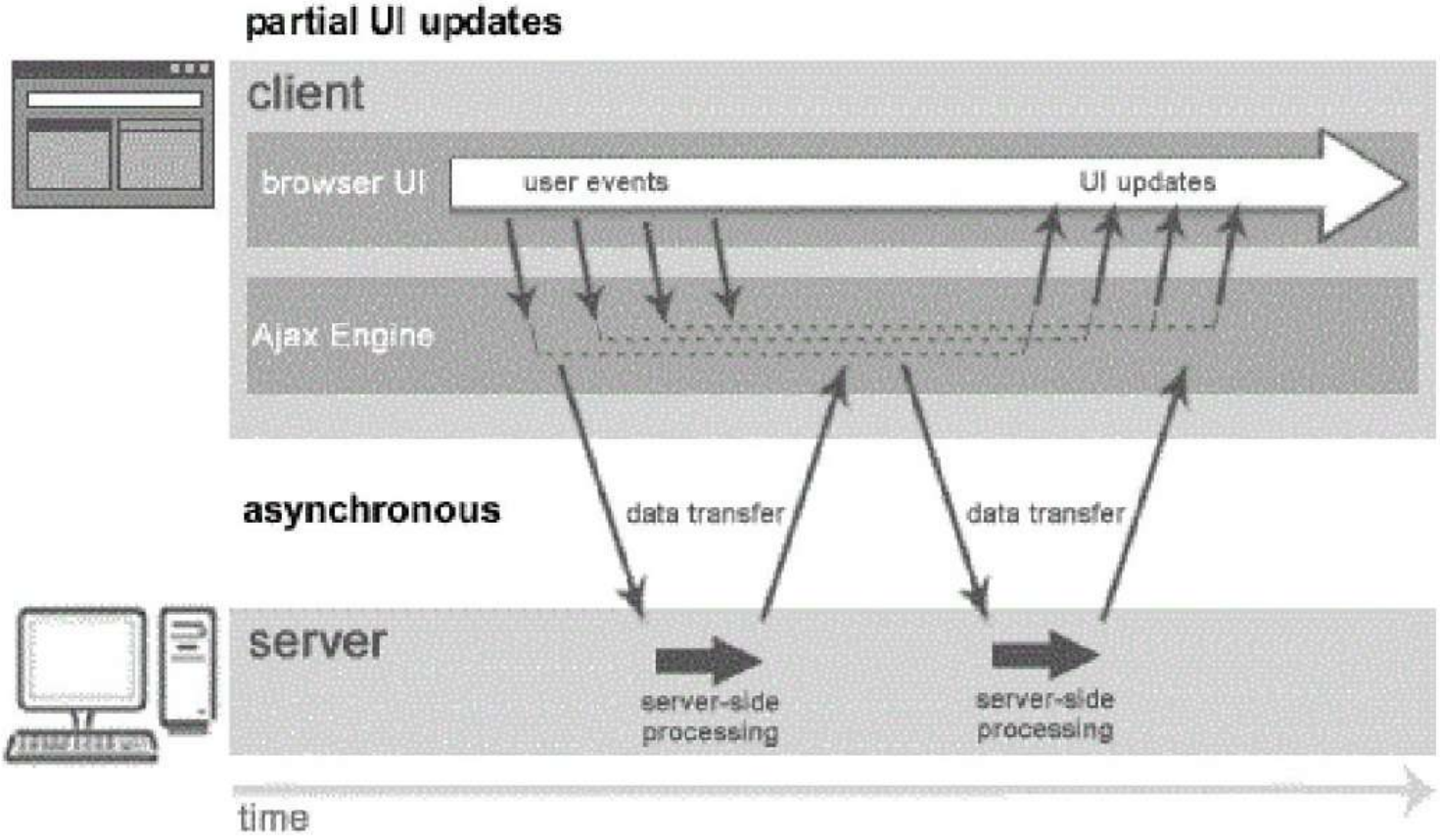
n the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.



Asynchronous (AJAX Web-Application Model)

An asynchronous request doesn't block the client i.e. browser is responsive. At that time, user can perform another operations also. In such case, javascript engine of the browser is not blocked.



In the above image, full page is not refreshed at request time and user gets response from the ajax engine.

Let's try to understand asynchronous communication by the image given below.

Note: every blocking operation is not synchronous and every unblocking operation is not asynchronous.

AJAX Technologies

Ajax is not a technology but group of inter-related technologies. AJAX technologies includes:

- HTML/XHTML and CSS
- DOM
- XML or JSON
- XMLHttpRequest
- JavaScript

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation.

DOM

It is used for dynamic display and interaction with data.

XML or JSON

For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

XMLHttpRequest

For asynchronous communication between client and server.

JavaScript

It is used to bring above technologies together.

Independently, it is used mainly for client-side validation.

Ajax Objects

1. XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

***var variablename* = new XMLHttpRequest();**

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data

status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

How to Send a Request to Server

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajaxfirst.txt", true);

xhttp.send();
```

Server Response

The onreadystatechange Property

The **readyState** property holds the status of the XMLHttpRequest.

The **onreadystatechange** property defines a function to be executed when the readyState changes.

The **status** property and the **statusText** property holds the status of the XMLHttpRequest object.

The onreadystatechange function is called every time the readyState changes.

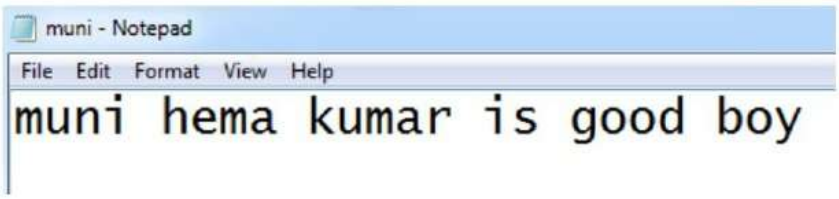
When readyState is 4 and status is 200, the response is ready:

```
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
```

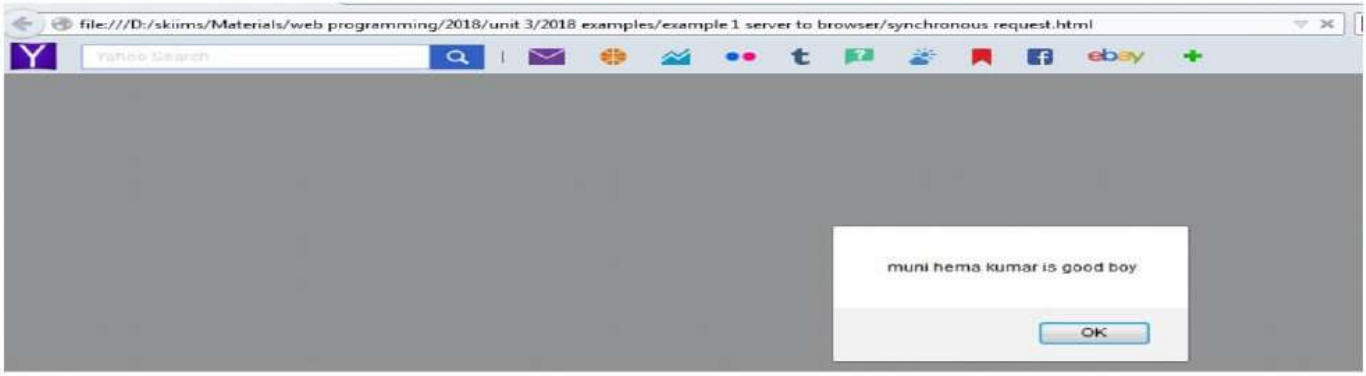
Synchronous Request(Mode):

This is not traditional mode we cannot get updated information without refreshing.

We must wait for response to do other java script code.
Step1: create xmlhttp request object
Step 2: call open method
Step 3: send request
Step 4: display responseText
synchronous.html

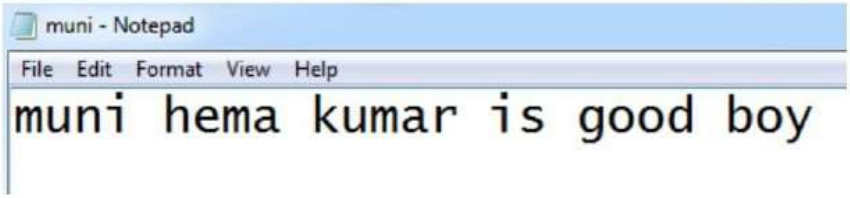


```
<script type="text/javascript">
var req=new XMLHttpRequest();
req.open("GET","muni.txt",false);
req.send();
alert(req.responseText);
</script>
```



Asynchronous Request(Mode):

This is traditional mode we can get updated information without refreshing.
In the same time we can do other java script codes.
Steps
Create XMLHttpRequest object
Call open method set the boolean value as true
onreadystatechange=function()
{
check ready state
if true display response text;
};
Call send method;



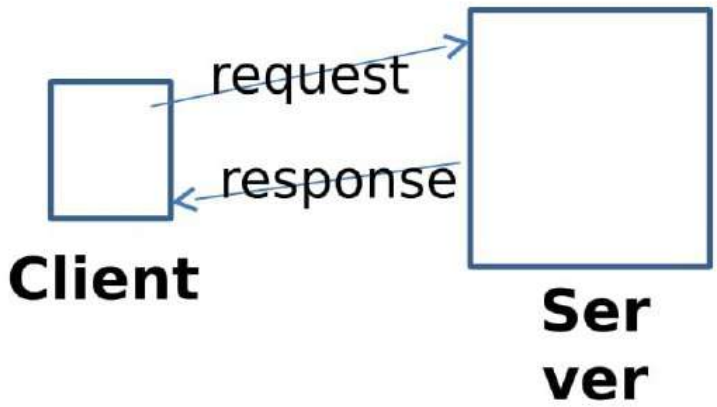
asynchronous.html

```
<script type="text/javascript">
function muni()
{
var req=new XMLHttpRequest();
req.open("GET","muni.txt",true);
req.onreadystatechange=function()
{
 if(req.readyState==4)
 {
  document.getElementById("ch").innerHTML=req.responseText;
 }
};
req.send();
```

```
}
</script>
<div id="ch">Hi students</div>
<button onclick="muni();">click to change</button>
```

2. JSON object:

- Java Script Object Notation
- It is a format to transfer data from server to client or client to server.
- The technology is used in Server the technology may be php, .net. Java it's not matter.
- What matter is From the server what you returned to client side would be data.
- You have design at client side but don't have data.
- The data coming from server.
- When client send request to server you will get response in some format from the server.
- I don't want html (it returns design page), I want data.



JSON Methods

1. JSON.parse():

A common use of JSON is to exchange data to/from a web server.

When receiving data from a web server, the data is always a string.

Parse the data with JSON.parse(), and the data becomes a JavaScript object.

To deserialize a json structure into java script object

Syntax:JSON.parse(xmlhttprequestobject.responseText);

Example:

parsejson.html

```
<script type="text/javascript">
function muni()
{
var req=new XMLHttpRequest();
req.open("GET","munijson.txt",true);
req.onreadystatechange=function()
{
 if(req.readyState==4)
 {
  var rss=JSON.parse(req.responseText);
  document.getElementById("ch").innerHTML=rss.name;
 }
};
req.send();
}
</script>
<div id="ch">Hi students</div>
<button onclick="muni();">click to change</button>
```

muni.txt

```
{ "name":"muni", "age":31, "city":"Tirupati" }
```

AJAX LIBRARIES

1. Dojo Toolkit 1.12

A JavaScript toolkit that saves you time and scales with your development process. Provides everything you need to build a Web app. Language utilities, UI components, and more, all in one place, designed to work together perfectly.

```
<script src="//ajax.googleapis.com/ajax/libs/dojo/1.12.1/dojo/dojo.js"></script>
```

someObect.someMethod();

Sending Data to the Server Using GET

```
<script type="dojo/method" data-dojo-event="onClick">
 dojo.xhrPost({
    url: 'HelloWorldResponsePOST.php',
    load: helloCallback,
    error: helloError,
    form: 'myForm'
 });
</script>
```

Reading Data from the Server

```
<script type="dojo/method" data-dojo-event="onClick">
 dojo.xhrGet({
    url: 'response.txt',
    load: helloCallback,
    error: helloError
 });
</script>
```

2. Prototype:

Prototype is a JavaScript library aims to ease the development of dynamic web applications. Prototype was developed by Sam Stephenson.

Prototype enables you to deal with Ajax calls in a very easy and fun way that is also safe (cross-browser).

```
<script type = "text/javascript" src = "/javascript/prototype.js"></script>
new Ajax.Request(url[, options]);
```

How to send data

```
function ajax_test()
{
        var url = "test_ajax_functions.php";
        var data = "something=nothing";
        var target = 'output';
        function ajax_response(resp)
        {
                alert(resp.responseText);
        }

        var myAjax = new Ajax.Updater(
                target,
                url,
```

```
            {method: 'post', parameters: data, onComplete: ajax_response}
        );
}
```

How to receive data

```
function SubmitRequest()
{
        new Ajax.Request('url', {
          method: 'get',
          onSuccess: successFunc,
          onFailure:  failureFunc
        });
      }
```

3. MooTools

MooTools is a collection of JavaScript utilities designed for the intermediate to advanced JavaScript developer. It allows you to write powerful and flexible code with its elegant, well documented, and coherent APIs.

MooTools code is extensively documented and easy to read, enabling you to extend the functionality to match your requirements.

```
var myRequest = new Request([options]);
```

Receiving data

```
 var myRequest = new Request({
            url: 'truncate.php',
            method: 'post',
            onRequest: function(){
            },
            onSuccess: function(responseText){
                alert("done!"+ responseText);
            },
            onFailure: function(){
                alert("failed");
            }
        });
```

Sending data

```
var myRequest = new Request({
      url: 'truncate.php',
      method: 'post',
      onRequest: function() {
      },
      onSuccess: function(responseText) {
        alert("done! " + responseText);
      },
      onFailure: function() {
        alert("failed");
      }
   });
```

4. Spry

The **Spry** is an open source Ajax library developed by Adobe Systems which is used in the construction of Rich Internet applications.[1] Unlike other pure JavaScript frameworks such as the Dojo Toolkit and Prototype, Spry is geared towards web designers, not web developers.

The Spry broadly consists of

- Spry Effects - animation effects like blind, fade, grow, highlight, shake, slide and squish.
- Spry Data - data binding to HTML markup using minimal code or proprietary markup. Spry uses Google's Xpath JavaScript library to convert XML into JavaScript objects. It can handle XML, HTML and JSON data.
- Spry Widgets - framework for development of widgets, and included widgets such as the accordion.

Usage

- Ajax development within an IDE such as Eclipse (software).
- Ajax generation from server code using ColdFusion. Ruby on Rails offers similar functionality.
- Ajax application generation from Adobe Flex code. OpenLaszlo will offer similar functionality with their "Legals" release (version 4).

5. YUI Library

The Yahoo! User Interface Library (**YUI**) is a discontinued open-source JavaScript library for building richly interactive web applications using techniques such as Ajax, DHTML, and DOM scripting. YUI includes several core CSS resources. Development on YUI began in 2005 and Yahoo! properties such as My Yahoo! and the Yahoo! front page began using YUI in the summer of that year. YUI was released for public use in February 2006.[1] It was actively developed by a core team of Yahoo! engineers.

Features

Core: The YUI Core is a light (31KB minified) set of tools for event management and DOM manipulation.
YUI Global Object: The YUI Global Object contains language utilities, a script loader, and other baseline infrastructure for YUI.

Dom Collection: Helps with common DOM scripting tasks, including element positioning and CSS style management.

Event Utility: Provides developers with easy and safe access to browser events (such as mouse clicks and key presses). It also provides the Custom Event object for publishing and subscribing to custom events.

Server:

- You want to print a file
- You will raise a request
- And for that request to some sort of software which can handle your request and print the document as per your demand.
- Piece software here print server.
- A server is piece of software that resides on a particular network and takes your request and do something according to that request.

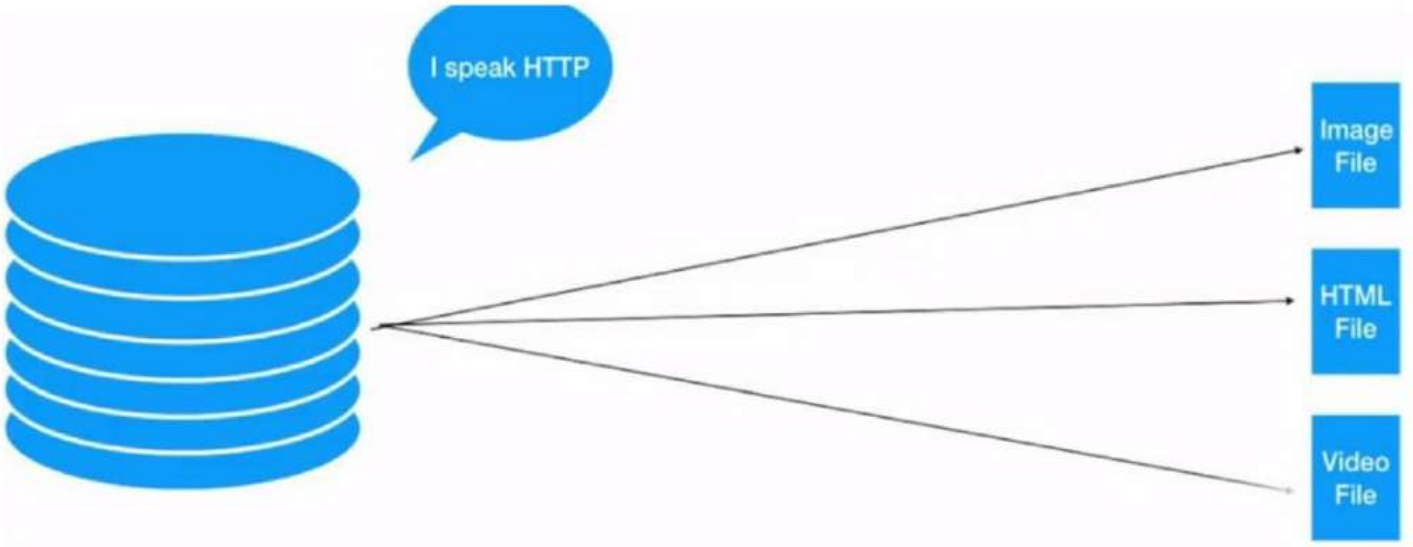
Web Server:

A **Web server** can be either a computer program or a computer running a program that is responsible for accepting HTTP requests from clients, serving back HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects on it.

Is all about dealing with the http content of content which can be handled over the http protocol.

A Web server is a system that delivers content or services to end users over the Internet. A Web server consists of a physical server, server operating system (OS) and software used to facilitate HTTP communication.

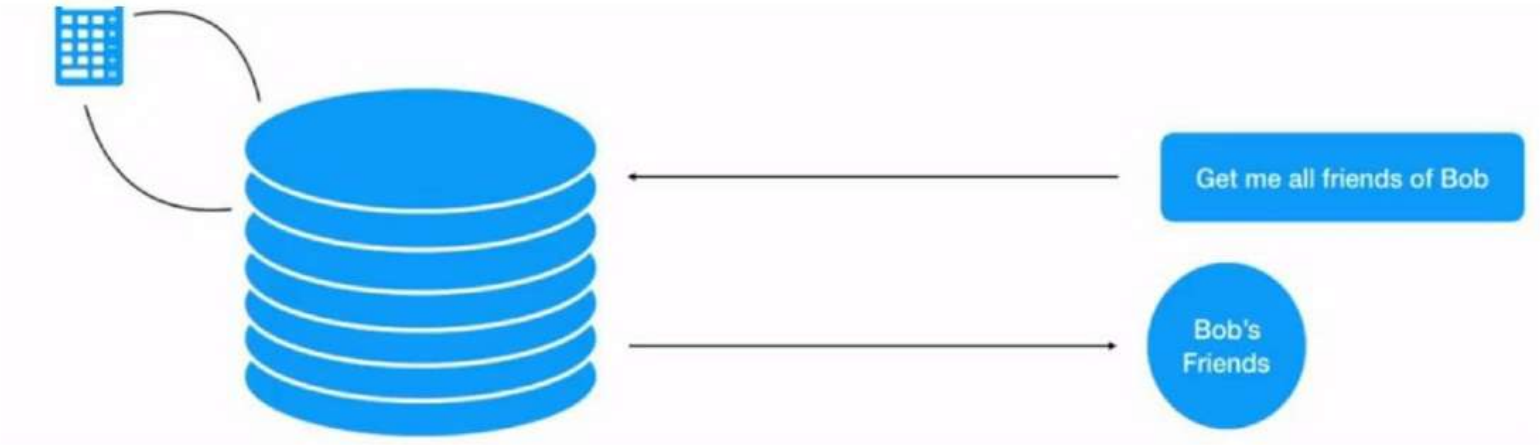
A Web server is a computer system that hosts websites. It runs Web server software, such as Apache or Microsoft IIS, which provides access to hosted webpages over the Internet.



Application Server:

An application server is the kind of software engine that will deliver various applications to another device. It is the kind of computer found in an office or university network that allows everyone in the network to run software off of the same machine.

Is all about executing the programming logic and produce the output according to application logic.

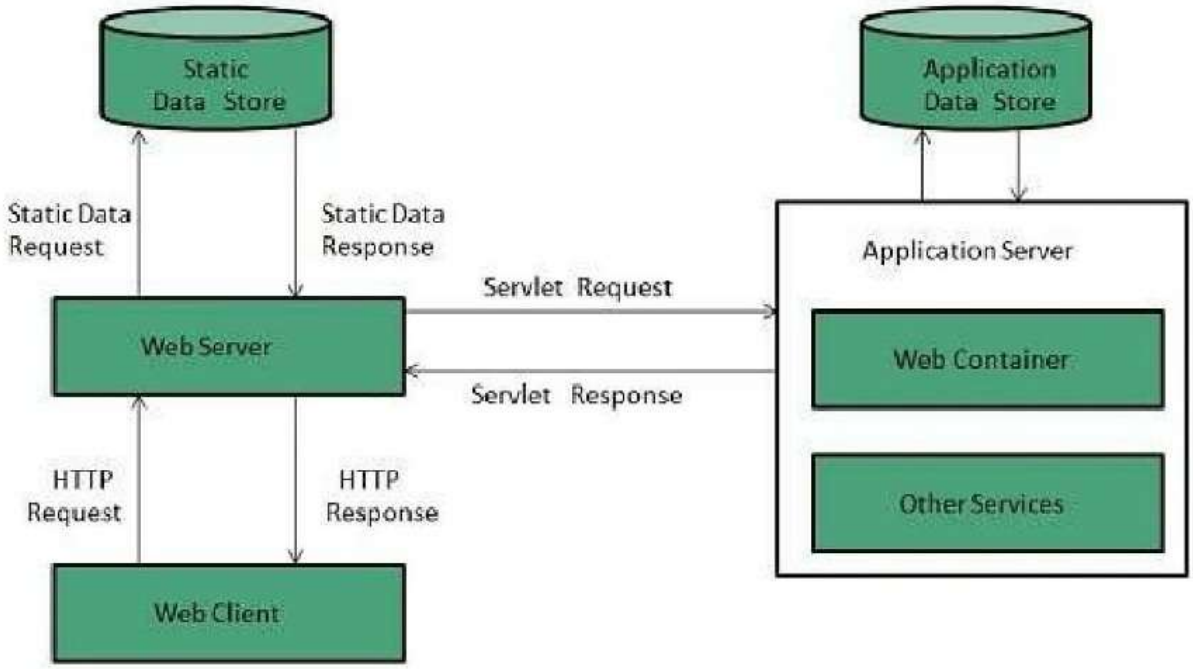


Web Server	Application Server
I have web application to run the web application we require web server.	**To run enterprise application we require application server.**
Web server provides environment to run web applications.	Application server provides environment to run enterprise applications.
Web server can provide support only for web related technologies	Application server can provide support any technologies

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



- 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.

Server Architecture

Web Server Architecture follows the following two approaches:

1. Concurrent Approach
2. Single-Process-Event-Driven Approach.

Concurrent Approach

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.

Multi-processing

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

Multi-threaded

Unlike Multi-process, it creates multiple single-threaded process.

Hybrid

It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

IIS(Internet Information Services/Server)

Stands for "Internet Information Services." IIS is a web server software package designed for Windows Server. It is used for hosting websites and other content on the Web.

Features:

- Microsoft's Internet Information Services provides a graphical user interface (GUI) for managing websites and the associated users.
- It provides a visual means of creating, configuring, and publishing sites on the web.
- The IIS Manager tool allows web administrators to modify website options, such as default pages, error pages, logging settings, security settings, and performance optimizations.
- IIS can serve both standard HTML webpages and dynamic webpages, such as ASP.NET applications and PHP pages.
- When a visitor accesses a page on a static website, IIS simply sends the HTML and associated images to the user's browser.
- When a page on a dynamic website is accessed, IIS runs any applications and processes any scripts contained in the page, then sends the resulting data to the user's browser.
- While IIS includes all the features necessary to host a website, it also supports extensions (or "modules") that add extra functionality to the server. For example, the WinCache Extension enables PHP scripts to run faster by caching PHP processes.
- The URL Rewrite module allows webmasters to publish pages with friendly URLs that are easier for visitors to type and remember.
- A streaming extension can be installed to provide streaming media to website visitors.
- IIS is a popular option for commercial websites, since it offers many advanced features and is supported by Microsoft.

- However, it also requires requires a commercial license and the pricing increases depending on the number of users. Therefore, Apache HTTP Server, which is open source and free for unlimited users, remains the most popular web server software.

Installing IIS

To install IIS:

1. In Windows, access the Control Panel and click **Add or Remove Programs**.
2. In the Add or Remove Programs window, click **Add/Remove Windows Components**.
3. Select the **Internet Information Services (IIS)** check box, click **Next**, then click **Finish**.
4. To learn how to use IIS, you can view the documentation at http://localhost/iishelp/iis/misc/default.asp.

Configuring IIS

To configure IIS:

1. Right-click the **My Computer** icon on your server computer desktop, and then click **Manage**.
2. In the **Computer Management** dialog box, open the **Services and Applications** node.
3. Click **Internet Information Services**, and then click **Web Sites**.
4. Right-click the **Default Web Site** node to start it, if it is not started already.
5. If a secure Internet connection is required, set up Secure Sockets Layer (SSL).

Apache

Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules.

Apache is the most popular web server (after which comes Microsoft's IIS) available. The reasons behind its popularity, to name a few, are:

1. It is free to download and install.

2. It is open source: the source code is visible to anyone and everyone, which basically enables anyone (who can rise up to the challenge) to adjust the code, optimize it, and fix errors and security holes. People can add new features and write new modules.

3. It suits all needs: Apache can be used for small websites of one or two pages, or huge websites of hundreds and thousands of pages, serving millions of regular visitors each month. It can serve both static and dynamic content.

How Apache Works

Apache's main role is all about communication over networks, and it uses the TCP/IP protocol (Transmission Control Protocol/Internet Protocol which allows devices with IP addresses within the same network to communicate with one another).

The TCP/IP protocol is a set of rules that define how clients make requests and how servers respond, and determine how data is transmitted, delivered, received, and acknowledged.

The Apache server is set up to run through configuration files, in which directives are added to control its behavior. In its idle state, Apache listens to the IP addresses identified in its config file (HTTPd.conf). Whenever it receives a request, it analyzes the headers, applies the rules specified for it in the Config file, and takes action.

But one server can host many websites, not just one - though, to the outside world, they seem separate from one another. To achieve this, every one of those websites has to be assigned a different name, even if those all map eventually to the same machine. This is accomplished by using what is known as virtual hosts.

Since IP addresses are difficult to remember, we, as visitors to specific sites, usually type in their respective domain names into the URL address box on our browsers. The browser then connects to a DNS server, which translates the domain names to their IP addresses. The browser then takes the returned IP address and connects to it. The browser also sends a *Host* header with the request so that, if the server is hosting multiple sites, it will know which one to serve back.

For example, typing in www.google.com into your browser's address field might send the following request to the server at that IP address:

```
1    GET / HTTP/1.1

2    Host: www.google.com
```

The first line contains several pieces of information. First, there is the method (in this case it's a GET), the URI, which specifies which page to be retrieved or which program to be run (in this case it's the root directory denoted by the /), and finally there is the HTTP version (which in this case is HTTP 1.1).

HTTP is a request / response stateless protocol.

HTTP is a request / response stateless protocol. It's a set of rules that govern communication between a client and the server. The client (usually but not necessarily a web browser) makes a request, the server sends back a response, and communication stops. The server doesn't look forward for more communication as is the case with other protocols that stay at a waiting state after the request is over.

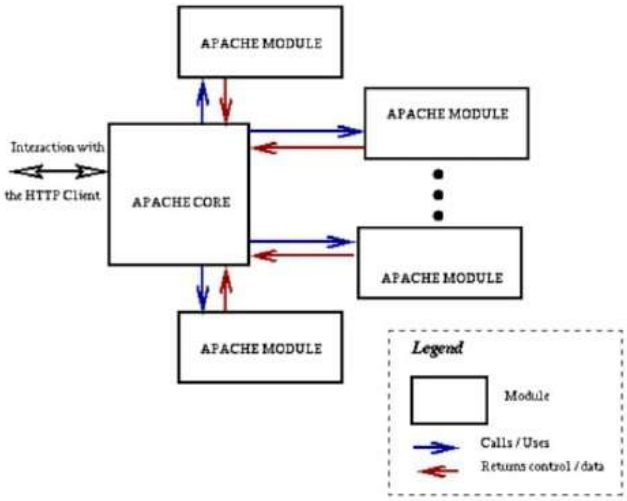
If the request is successful, the server returns a 200 status code (which means that the page is found), response headers, along with the requested data. The response header of an Apache server might look something like the following:

```
01  HTTP/1.1 200 OK
02  Date: Sun, 10 Jun 2012 19:19:21 GMT
03  Server: Apache
04  Expires: Wed, 11 Jan 1984 05:00:00 GMT
05  Cache-Control: no-cache, must-revalidate, max-age=0
06  Pragma: no-cache
07  Last-Modified: Sun, 10 Jun 2012 19:19:21 GMT
08  Vary: Accept-Encoding,User-Agent
09  Content-Type: text/html; charset=UTF-8
10  Content-Length: 7560
```

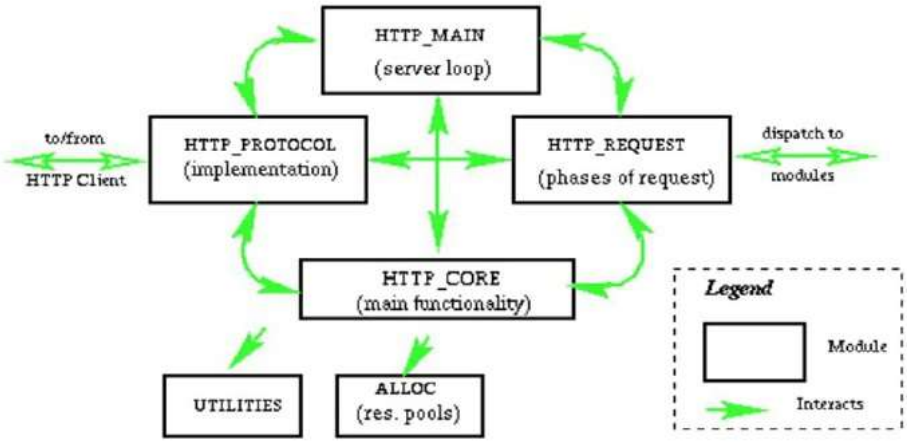
The first line in the response header is the status line. It contains the HTTP version and the status code. The date follows next, and then some information about the host server and the retrieved data. The Content-Type header lets the client know the type of data retrieved so it knows how to handle it. Content-Length lets the client know the size of the response body. If the request didn't go throw, the client would get an error code and message, such as the following response header in case of a page not found error:

```
1                    HTTP/1.1 404 Not Found
```

Apache Architecture



Apache Core: Which Handles the Basic functionality of the Server. Such as allocating requests and maintaining and pooling all the connections.



- The core components of make up the Apache core are as follows:
    - http_protocol.c: This is the component that handles all of the routines that communicate directly with the client by using the HTTP protocol. This is the component that knows how to also handle the socket connections through which the client connects to the server. All data transfer is done through this component.
    - http_main.c: this component is responsible for the startup of the server and contains the main server loop that waits for and accepts connections. It is also in charge of managing timeouts.
    - http_request.c: This component handles the flow of request processing, passing control to the modules as needed in the right order. It is also in charge of error handling.
    - http_core.c: the component implementing the most basic functionality, it just bairly serves documents.
    - alloc.c: the component that takes care of allocating resource pools, and keeping track of them.
    - http_config.c : this component provides functions for other utilities, including reading configuration files and managing the information gathered from those files (), as well as support for virtual hosts. An important function of http_config is that it forms the list of modules that will be called to service during different phases of the requests that are going on within the server.

How to deploy a webapplication in Tomcat Server(How to host a website)

Tomcat's manager application

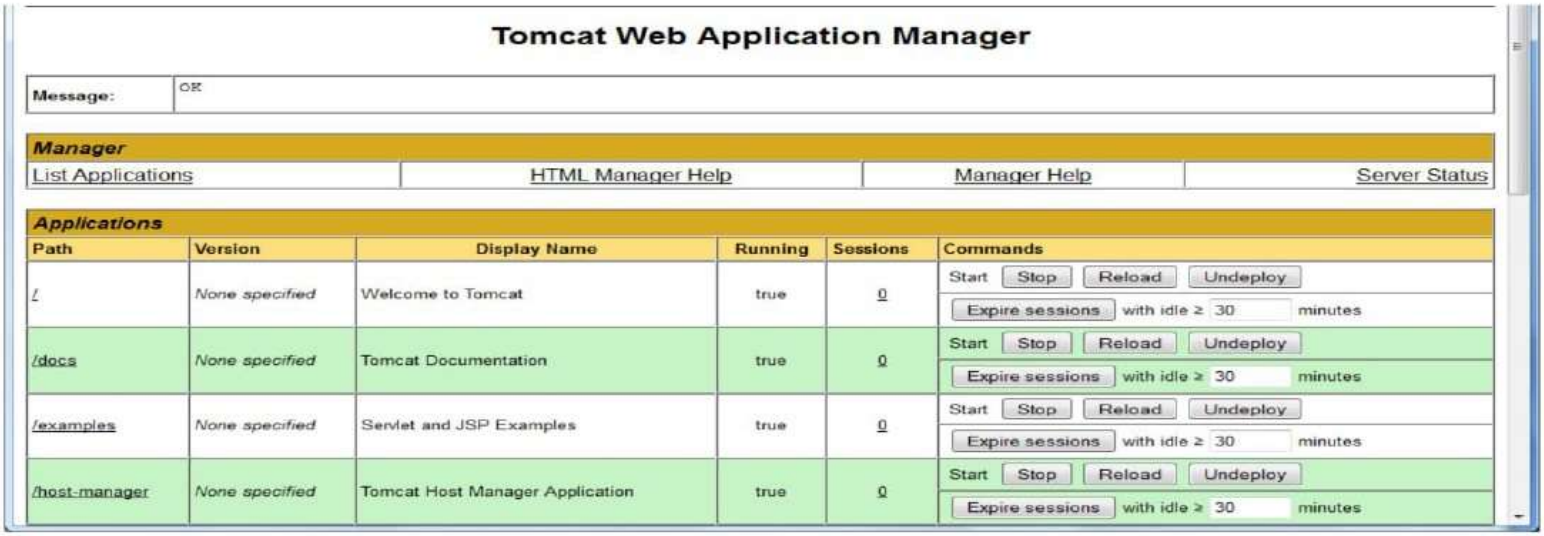
We can deploy the web application remotely via a web interface provided by Tomcat's manager application. You must have user name and password to access this application. The manager application is installed by default, but not always. So be sure that it is installed with your version of Tomcat.

Using the manager application, you can:

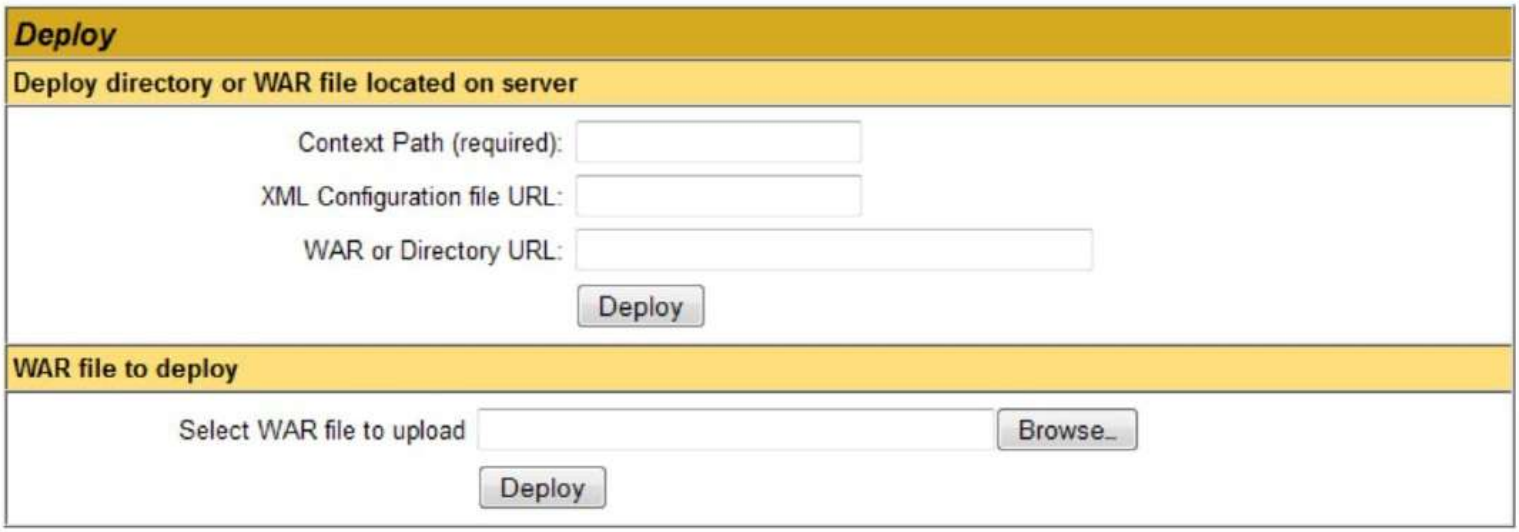
- View a list of applications deployed on the server and their status.
- Start, stop and restart an individual application.
- Deploy a new web application either by uploading a WAR file or supplying a directory on the server.
- Undeploy an individual application.

By default, the manager application is deployed under context */manager*, so to access it, type the following URL into your web browser's address bar (the port number may vary, depending on your server's configuration):
http://localhost:8080/manager

After supplying correct user name and password, you get into the following screen:



The list of deployed applications is shown at the top, scroll down a little bit to see the deployment section:



As we can see, there are two ways for deploying a web application using the manager:

- Deploy directory or WAR file located on server.
- WAR file to deploy.

PHP

PHP is a server side scripting language used to develop Static websites or Dynamic websites or Web applications.

PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

PHP scripts can only be interpreted on a server.

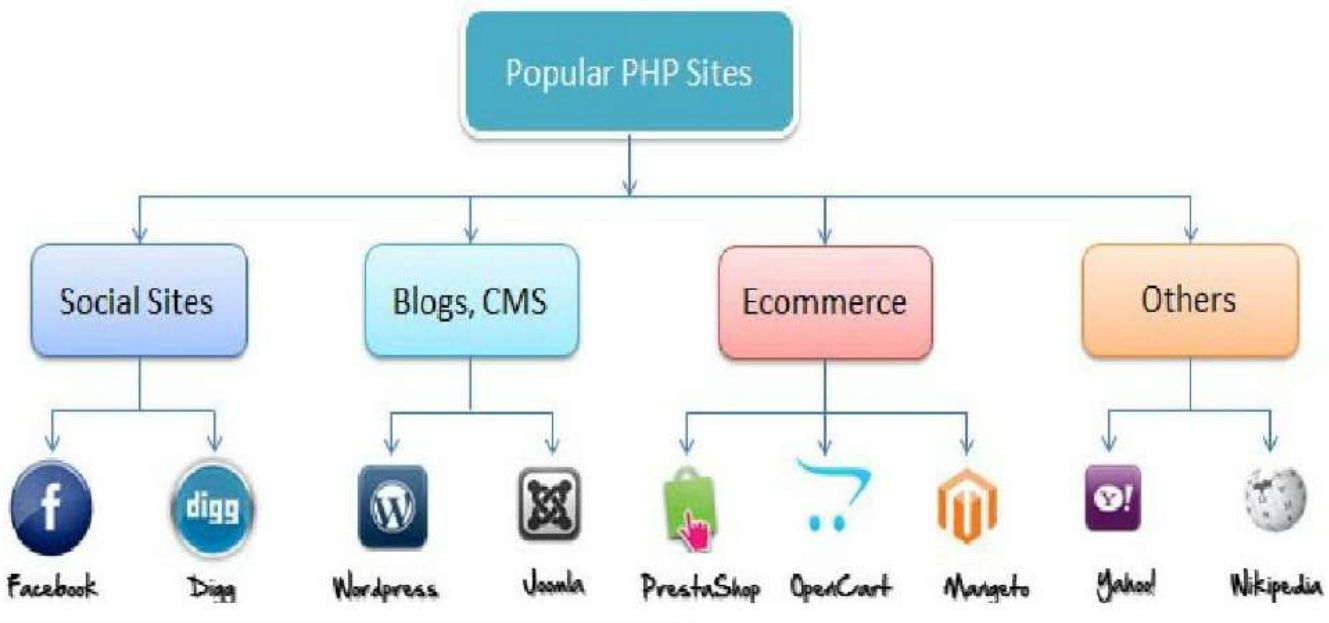
The client computers accessing the PHP scripts require a web browser only.

PHP code may be embedded into HTML code.

A PHP file contains PHP tags and ends with the extension ".php".

Why use PHP?

- PHP is **open source and free.**
- Short learning curve compared to other languages such as JSP, ASP etc.
- Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.
- PHP is regular updated to keep abreast with the latest technology trends.
- Other benefit that you get with PHP is that it's a **server side scripting language**; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is **cross platform;** this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.



Php data types:

There is no specific datatype in php. We can assign any datatype to the variable through var keyword.

Int datatype:

var a=10;

float datatype:

var a=10.53;

char datatype:

var a='m';

String datatype:

var a="muni";

Example:

```
<?php
$a=5;
$b=150.155;
$c='s';
$d="IV SEM MCA";
echo "Integer value is".$a."<br/>";
echo "Float value is".$b."<br/>";
echo "Single character is".$c."<br/>";
echo "String is".$d;
?>
```

Line Break; Used to give a line break

```
echo "variable or string <br/>"
```

Addition of two numbers

```
<?php
$n1=10;
$n2=20;
```

```
$sum=$n1+$n2;

echo "addition is $sum";

?>
```

Control Structures

The control structures within Javascript allow the program flow to change within a unit of code or function. These statements can determine whether or not given statements are executed, as well as repeated execution of a block of code.

Contents

1 if
2 while
3 do... while
4 for
5 switch

1. If...Else Statements

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

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

(i) If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
{
    code to be executed if condition is true
}
```

Example

```
<?php
    $a=10;
    $b=20;
    if($a<$b)
    echo "b is big";
?>
```

(ii) If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
        code to be executed if condition is true
  }
else
  {
        code to be executed if condition is not true
  }
```

Example

```
<?php
        $a=25;
        $b=20;
        if($a<$b)
        echo "b is big";
        else
        echo "a is big";
?>
```

(iii) If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

Syntax

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

Example

```
<?php
         $x=10;
         $y=5;
        if($x==$y)
        {
        echo "Condition1 true <br/>";
        }
        else if($x<=$y)
        {
        echo "Condition2 true <br/>";
        }
        else
        {
```

```
        echo "Condition 3 true";
        }
?>
```

2. Loops

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

(i) While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

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

Syntax

```
intialization
while (condition)
{
code to be executed
increment or decrement
}
```

Example

```
<?php
echo "Numbers are <br/>";
$a=1;
while($a<=5)
{
 echo "$a <br/>";
 $a++;
}
?>
```

Output

Numbers are
1
2
3
4
5

(ii). do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
intialization
```

```
do
{
        code to be executed
        increment or decrement
}
while (condition);
```

Example

```
<?php
        echo "Numbers are <br/>";
        $a=1;
        do
        {
         echo "$a <br/>";
         $a++;
        }while($a<=5);
?>
```

Numbers are
1
2
3
4
5

(ii) The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (intialization;condition;increment or decrement)
{
        code to be executed
}
```

Example

```
<?php
echo "Numbers are <br/>";
for($a=1;$a<=5;$a++)
{
 echo "$a <br/>";
}
?>
```

Numbers are
1
2
3
4
5

Switch Statement

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

Syntax

```
switch(variablename)
{
        case 1:
                execute code block 1
                break;
        case 2:
                execute code block 2
```

```
        break;
    default:
        code to be executed if variable value  is different
        from case 1 and 2
}
```

Example

```
<?php
    $a=1;
    switch($a)
    {
      case 1:echo "mega star";break;
      case 1:echo "power star";break;
      case 1:echo "super star";break;
    }
?>
```

Output

mega star

Php Programs

Example 1:Even or Odd Program

```
<?php
$number=1233456;
if($number%2==0)
{
 echo "$number is Even Number";
}
else
{
 echo "$number is Odd Number";
}
?>
```

Example 2: Factorial Number

```
<?php
$num = 4;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
  $factorial = $factorial * $x;
}
echo "Factorial of $num is $factorial";
?>
```

Example 3: Palindrome number Program in PHP

```
<?php
$num = 121;
$p=$num;
$revnum = 0;
while ($num != 0)
```

```
{
$revnum = $revnum * 10 + $num % 10;
//below cast is essential to round remainder towards zero
$num = (int)($num / 10);
}
if($revnum==$p)
{
echo $p," is Palindrome number";
}
else
{
echo $p." is not Palindrome number";
}
?>
```

Example 4: Fibonacci Series

```
<?php
$num = 0;
$n1 = 0;
$n2 = 1;
echo "<h3>Fibonacci series for first 12 numbers: </h3>";
echo "\n";
echo $n1.' '.$n2.' ';
while ($num < 10 )
{
$n3 = $n2 + $n1;
echo $n3.' ';
$n1 = $n2;
$n2 = $n3;
$num = $num + 1;
}
?>
```

Arrays

An array stores multiple values in one single variable:

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

Indexed arrays - Arrays with a numeric index

Associative arrays - Arrays with named keys

Multidimensional arrays - Arrays containing one or more arrays

1. PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

$arrayname = array(element1,element2, element3);

$arrayname = array("element1","element2", "element3"); //string

or the index can be assigned manually:

$arrayname[0] = element1;
$arrayname[1] = element2;
$arrayname [2] = element3;

Printing elements

echo arrayname[0].arrayname[1].arrayname[2].arrayname[3];

Example:

```
<?php
$a=array(5,15,4,12);
echo "Array elements are";
echo $a[0]."<br/>";
echo $a[1]."<br/>";
echo $a[2]."<br/>";
echo $a[3];
?>
```

Output

Array elements are
5
15
4
12

Get Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
$a=array(5,15,4,12,8,7,14);
echo count($a);
```

Output

7

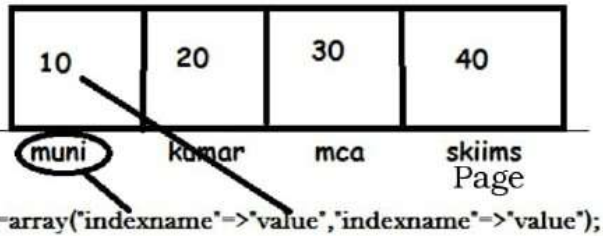
2. PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

$a=array("muni"=>"25","kumar"=>"15","hema"=>"21");

or:



$a['muni']= “25";
$a['kumar']= “15";
$ a['hema'] = “21";

The named keys can then be used in a script:

Example

```
<?php
$a=array("muni"=>"10","kumar"=>"20","mca"=>"30","skiims"=>"40");
echo "Array elements are"."<br/>";
echo $a['muni']."<br/>";
echo $a['kumar']."<br/>";
echo $a['mca'] ."<br/>";
echo $a['skiims'];
?>
```

3. Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP - Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

```
$arrayname = array(

        array(element1,element2,element3), //First row with row index
position 0
        array(element1,element2,element3) //Second row with row index
position 1
         );
```

Example

```
<?php
$a=array(array(10,20,30),array(40,50,60));
echo $a[0][0];
echo $a[0][1];
echo $a[0][2];
echo "<br/>";
echo $a[1][0];
echo $a[1][1];
echo $a[1][2];
?>
```

	0	1	2
0	10	20	30
1	40	50	60

Sorting Arrays

1. Sort Array in Ascending Order - sort()

sort(). This is a predefined method used to sort the array elements in ascending order.

Syntax: sort($a);

The following example sorts the elements of the $a array in ascending alphabetical order:

Example

$a=array(25,14,18,3,19,17,13,10);

sort($a)

```
<?php
$a=array(25,14,18,3,19,17,13,10);
$length=count($a);
echo "Before sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
sort($a);
echo "After sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
?>
```

Before sorting
25
14
18
3
19
17
13
10
After sorting
3
10
13
14
17
18
19
25

Before sorting
25
14
18
3
19
17
13
10
After sorting
3
10
13
14
17
18
19
25

2. Sort Array in Descending Order - rsort()

rsort(). This is a predefined method used to sort the array elements in descending order.

Syntax: rsort($a);

The following example sorts the elements of the $a array in ascending alphabetical order:

Example

$a=array(25,14,18,3,19,17,13,10);

rsort($a)

```
<?php
$a=array(25,14,18,3,19,17,13,10);
$length=count($a);
echo "Before sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
```

Before sorting
25
14
18
3
19
17
13
10
After sorting
25
19
18
17
14
13

```
 echo $a[$i]."<br/>";
}
rsort($a);
echo "After sorting"."<br/>";
for($i=0;$i<$length;$i++)
{
 echo $a[$i]."<br/>";
}
?>
```

Functions

User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute immediately when a page loads.

A function will be executed by a call to the function

1. Functions without parameters (Create a User Defined Function in PHP)

A user defined function declaration starts with the word "function":

Syntax

function *functionName*()

{
  code to be executed;
}

Example

```
<?php
function writeMsg() {
   echo "Hello world!";
}

writeMsg(); // call the function
?>
```

2. Function with parameters

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the muni() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
<?php
function muni($fname)

 {
    echo "$fname <br>";
}
muni("Jani");
muni("Hege");
muni("Stale");
muni("Kai Jim");
muni("Borge");
?>
```

3. Returning values

To let a function return a value, use the return statement:

Example

```
<?php
function sum($x, $y)

{
   $z = $x + $y;
   return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

GET & POST Methods

There are two ways the browser client can send information to the web server.

GET Method

POST Method

1. GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

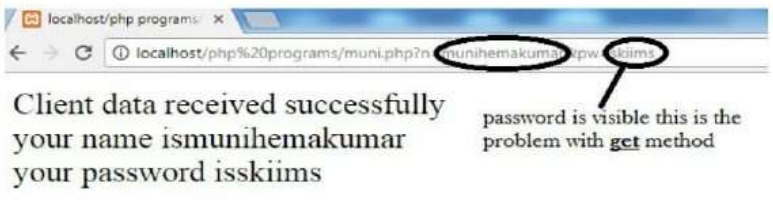
- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

getrequest.html

```
<form action="muni.php" method="get">
 Enter name<input type="text" name="n" /> <br/>
 Enter password<input type="password" name="pw" /> <br/>
 <input type="submit" value="submit" />
</form>
```

muni.php

```
<?php
 $yn=$_GET['n'];
 $ypw=$_GET['pw'];
 echo "Client data received successfully"."<br/>";
 echo "your name is".$yn."<br/>";
 echo "your password is".$ypw;
?>
```



The POST Method

- The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

postrequest.html

```
<form action="muni.php" method="post">
 Enter name<input type="text" name="n" /> <br/>
 Enter password<input type="password" name="pw" /> <br/>
 <input type="submit" value="submit" />
</form>
```

muni.php

```
<?php
 $yn=$_POST['n'];
 $ypw=$_POST['pw'];
 echo "Client data received successfully"."<br/>";
 echo "your name is".$yn."<br/>";
 echo "your password is".$ypw;
?>
```

MySql:

Introduction:

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at www.mysql.com, offers the database programmer with an array of options and capabilities rarely seen in other database servers. MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is charge for licensing the product.

These capabilities range across a number of topics, including the following:

a) Ability to handle an unlimited number of simultaneous users.

b) Capacity to handle 50,000,000+ records.

c) Very fast command execution, perhaps the fastest to be found on the market.

d)Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's free. That's right, T.c.X offers MySQL as a free product to the general public.

Reasons to Use MySQL

a) Scalability and Flexibility

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported.

b) High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results.

C) High Availability

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

d) Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa.

e) Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data inserts capability, and strong support for specialized web functions like fast full text searches.

f) Strong Data Protection

Because guarding the data assets of corporations is the number one job of database professionals, MySQL offers exceptional security features that ensure absolute data

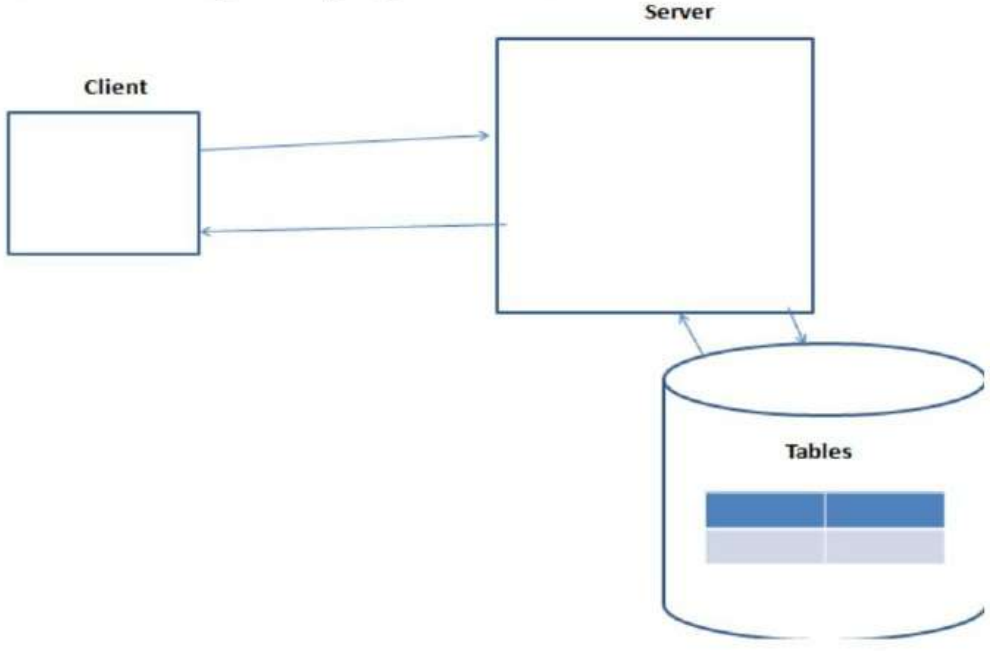
protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible.

g) Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX.

PHP Main Features of MySQL

- Tested with a broad range of different compilers.
- Works on many different platforms.
- The MySQL Server design is multi-layered with independent modules.
- Fully multi-threaded using kernel threads. It can easily use multiple CPUs if they are available.
- Provides transactional and non-transactional storage engines.
- Uses very fast B-tree disk tables with index compression.
- Relatively easy to add other storage engines. This is useful if you want to provide an SQL interface for an in-house database.
- A very fast thread-based memory allocation system.
- Very fast joins using an optimized one-sweep multi-join.
- In-memory hash tables, which are used as temporary tables.
- SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.
- The server is available as a separate program for use in a client/server networked environment.



Database Connectivity (PHP Database Interaction in FIVE steps)

PHP has a pretty straight forward method to working with MySQL databases. I will explain to establish a connection to the MySQL database, but I am not going to go into SQL which is the language used to put data in and get data out of a database. There are five steps to make PHP database interaction and I will explain each one in detail with example code:

1. Create connection
2. Select database
3. Perform database query
4. Use return data
5. Close connection

1. Create connection

It is very important when you want to start a dynamic website to create connection with your SQL (we are talking about mysql) by using mysql_connect() function. In mysql_connect() arguments should be string and it's important to use die() function with mysql_error() function to check if there is a problem with the connection or not.

2. Select database

Now we have to select the database which we are creating and saving our data by using mysql_select_db() function which the arguments are the name of database and connection we made earlier.

3. Perform database query

In this step we have to select out data from database and bring it into our web page. The best decision to use mysql_query() function which it will Send a MySQL query with 2 arguments as displayed in the above code.

4. Use return data

To display the result on your web page, you have to use while() loop function in addition to mysql_fetch_array() function which fetch a result row as an associative array, a numeric array, or both.

5. Close connection

To close connection, it is better to use mysql_close() function to close MySQL connection. This is a very important function as it closes the connection to the database server. Your script will still run if you do not include this function. And too

many open MySQL connections can cause problems for your account. This it is a good practice to close the MySQL connection once all the queries are executed.

Example

// Five steps to PHP database connections:

// 1. Create a database connection

```
// (Use your own servername, username and password if they are different.)
// $connection allows us to keep referring to this connection after it is established
$connection = mysql_connect("localhost","root","myPassword");
if (!$connection)
{
die("Database connection failed: " . mysql_error());
}
```

// 2. Select a database to use

```
$db_select = mysql_select_db("widget_corp",$connection);
if (!$db_select)
{
die("Database selection failed: " . mysql_error());
}
```

// 3. Perform database query

```
$result = mysql_query("SELECT * FROM subjects", $connection);
if (!$result)
{
die("Database query failed: " . mysql_error());
}
```

// 4. Use returned data

```
while ($row = mysql_fetch_array($result))
{
echo $row["menu_name"]." ".$row["position"]."<br />";
}
```

// 5. Close connection

```
mysql_close($connection);
```

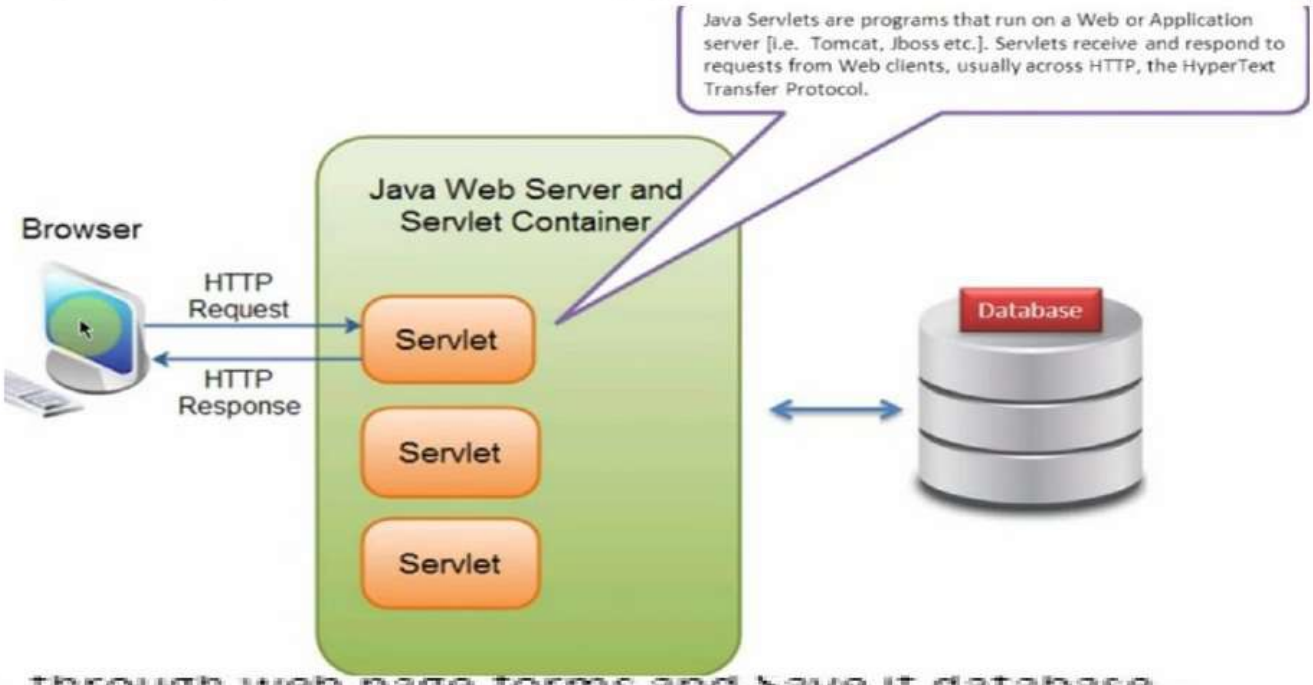
MySql Methods

1. mysqli_num_rows()

Definition and Usage

What is servlet

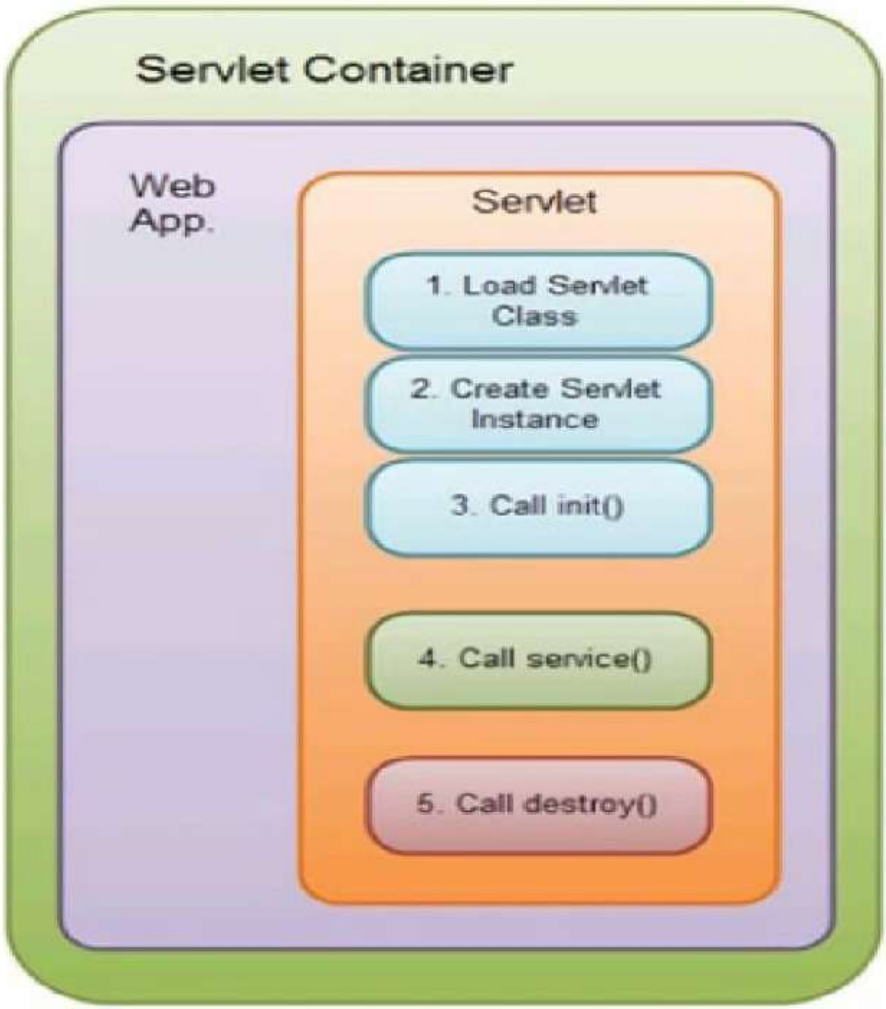
Servlet is a java file which can take the request from the client on the internet and it can process that request it can provide response in the format of html page.



through web page forms and save it database .

- ✓ Get records from a database and present to the
- ✓ Create web pages dynamically.

Servlet Life Cycle



1

Servlet Container load Servlet class

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class classname extends HttpServlet
{
}
```

2

Servlet Container creates an instance of the servlet

```
public void service(HttpServletRequest req,HttpServletResponse res)
{
}
```

3

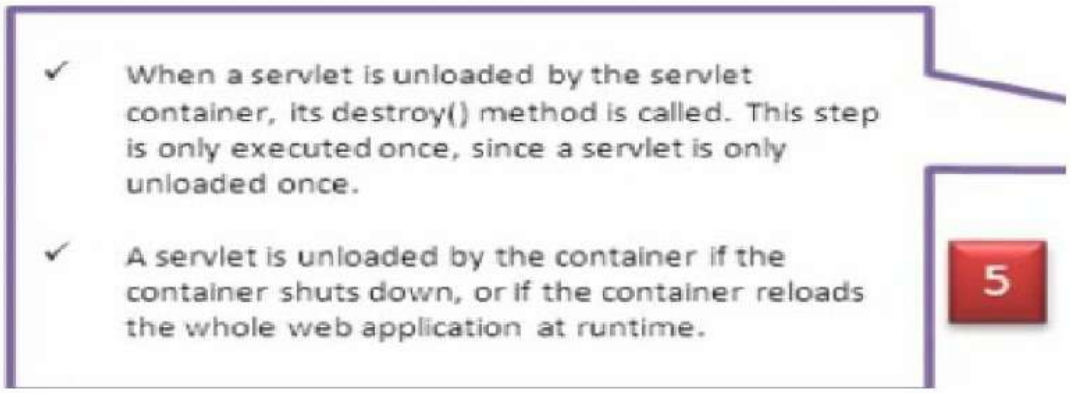
- ✓ Servlet Container will call the init() method of the servlet.
- ✓ init() method used to create or load some data that will be used throughout the life of the servlet.
- ✓ Init() method is executed once.

```
public void init()
{
int a=10;
}
```

4

- ✓ The service() method is the main method to perform the actual task.
- ✓ The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
- ✓ It is executed multiple times - once for every HTTP request to the servlet.

```
public void service(HttpServletRequest req,HttpServletResponse res)
  {
        PrintWriter out=res.getWriter();
        c=a+b;
        out.println("addition is"+c);
  }
```



Interfaces in javax.servlet package

There are many interfaces in javax.servlet package.

They are as follows:

- Servlet
- ServletRequest
- ServletResponse
- RequestDispatcher

1. Servlet Interface

Servlet interface provides common behaviour to all the servlets.

Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.

public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

2. ServletRequest Interface

An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc.

Methods of ServletRequest interface

There are many methods defined in the ServletRequest interface. Some of them are as follows:

Method	Description
public String getParameter(String name)	is used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
public int getContentLength()	Returns the size of the request entity data, or -1 if not known.
public ServletInputStream getInputStream() throws IOException	Returns an input stream for reading binary data in the request body.
public abstract String getServerName()	Returns the host name of the server that received the request.
public int getServerPort()	Returns the port number on which this request was received.

3. Servlet Response

Servlet API provides two important interfaces **ServletResponse** and **HttpServletResponse** to assist in sending response to client.

Some Important Methods of ServletResponse

Methods	Description
PrintWriter getWriter()	returns a PrintWriter object that can send character text to the client.
void setBufferSize(int size)	Sets the preferred buffer size for the body of the response
void setContentLength(int len)	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header

void setContentType(String type)	sets the content type of the response being sent to the client before sending the respond.
void setBufferSize(int size)	sets the preferred buffer size for the body of the response.
boolean isCommitted()	returns a boolean indicating if the response has been committed
void setLocale(Locale loc)	sets the locale of the response, if the response has not been committed yet.

4. RequestDispatcher

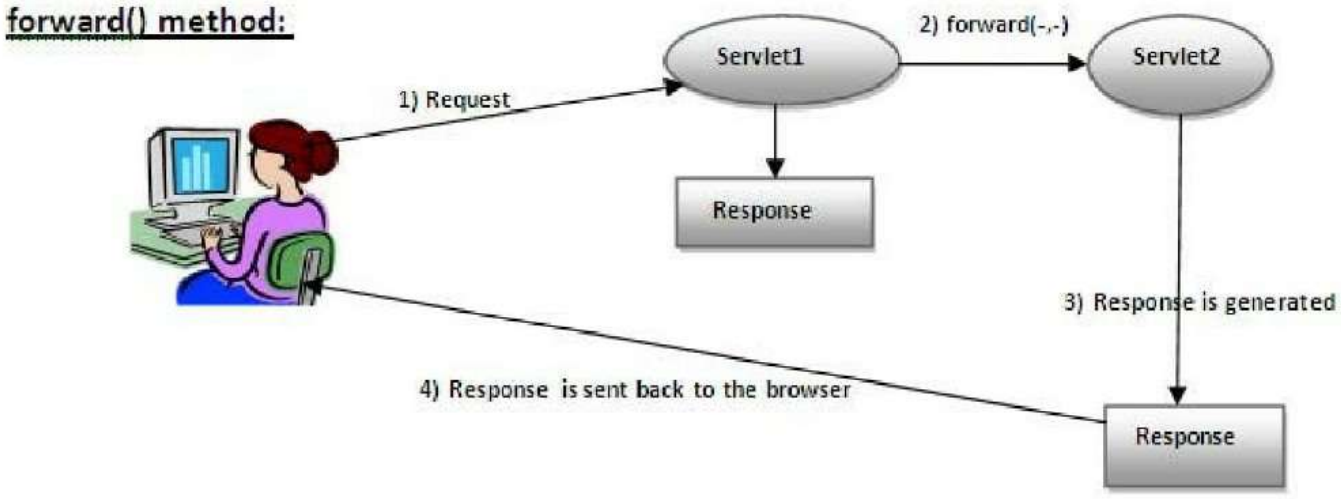
The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

Methods of RequestDispatcher interface

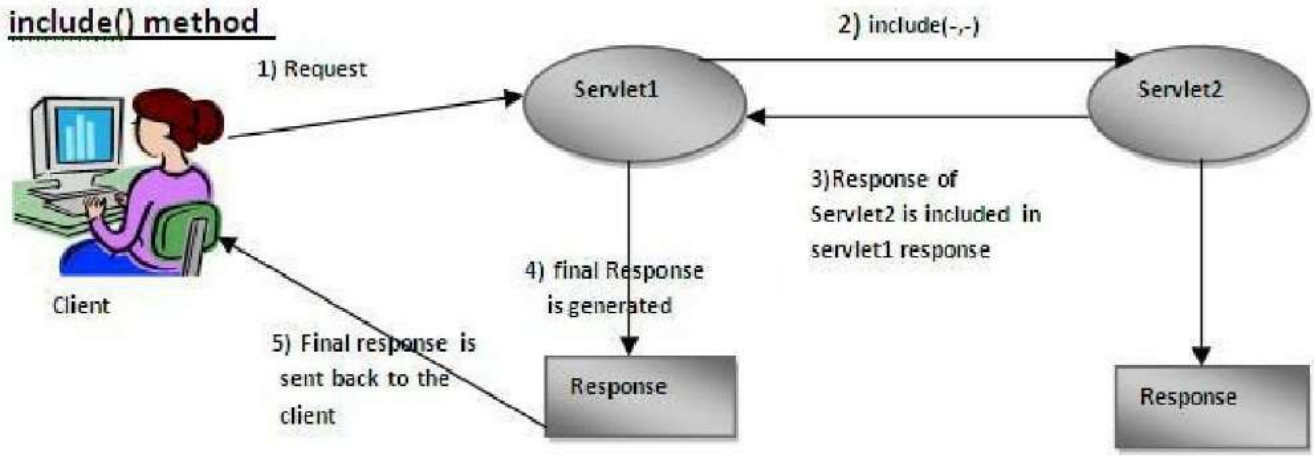
The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.

2. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.



Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. They are as follows:

- HttpServletRequest
- HttpServletResponse
- HttpSession

1. HttpServletRequest interface

HttpServletRequest interface adds the methods that relates to the **HTTP** protocol.

Some important methods of HttpServletRequest

Methods	Description
String getContextPath()	returns the portion of the request URI that indicates the context of the request
Cookies getCookies()	returns an array containing all of the Cookie objects the client sent with this request
String getQueryString()	returns the query string that is contained in the request URL after the path
HttpSession getSession()	returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session
String getMethod()	Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
String getServletPath()	returns the part of this request's URL that calls the servlet

2. HttpServletResponse Interface

HttpServletResponse interface adds the methods that relates to the **HTTP** response.

Some Important Methods of HttpServletResponse

Methods	Description
void addCookie(Cookie cookie)	adds the specified cookie to the response.
void sendRedirect(String location)	Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer
int getStatus()	gets the current status code of this response
String getHeader(String name)	gets the value of the response header with the given name.
void setHeader(String name, String value)	sets a response header with the given name and value
void setStatus(int sc)	sets the status code for this response
void sendError(int sc, String msg)	sends an error response to the client using the specified status and clears the buffer

3. HttpSession interface

HttpSession object is used to store entire session with a specific client. We can store, retrieve and remove attribute from **HttpSession** object. Any servlet can have access to **HttpSession** object throughout the getSession() method of the **HttpServletRequest** object.

The HttpSession object is used for session management. A session contains information specific to a particular user across the whole application. When a user enters into a website (or an online application) for the first time HttpSession is obtained via request.getSession(), the user is given a unique ID to identify his session. This unique ID can be stored into a cookie or in a request parameter.

The HttpSession stays alive until it has not been used for more than the timeout value specified in tag in deployment descriptor file(web.xml). The default timeout value is 30 minutes, this is used if you don't specify the value in tag. This means that when the user doesn't visit web application time specified, the session is destroyed by servlet container. The subsequent request will not be served from this session anymore, the servlet container will create a new session.

Creating a new session

getSession() method returns a session. If the session already exist, it return the esisting session else create a new sesion

```
HttpSession session = request.getSession();
```

Methods of HttpSession

1.setAttribute()

public void setAttribute(String name, Object value): Binds the object with a name and stores the name/value pair as an attribute of the HttpSession object. If an attribute already exists, then this method replaces the existing attributes.

You can store the user information into the session object by using setAttribute() method and later when needed this information can be fetched from the session. This is how you store info in session. Here we are storing username, emailid and userage in session with the attribute name uName, uemailId and uAge respectively.

```
session.setAttribute("uName", "ChaitanyaSingh");
session.setAttribute("uemailId", "myemailid@gmail.com");
session.setAttribute("uAge", "30");
```

This First parameter is the attribute name and second is the attribute value. For e.g. uName is the attribute name and ChaitanyaSingh is the attribute value in the code above.

2. getAttribute():

Returns the String object specified in the parameter, from the session object. If no object is found for the specified attribute, then the getAttribute() method returns null.

TO get the value from session we use the getAttribute() method of HttpSession interface. Here we are fetching the attribute values using attribute names.

```
String userName = (String) session.getAttribute("uName");
String userEmailId = (String) session.getAttribute("uemailId");
String userAge = (String) session.getAttribute("uAge");
```

Example:

index.html

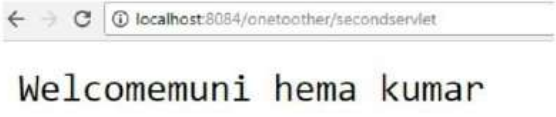
```
<form method="get" action="firstservlet">
        Enter name<input type="text" name="na" /> <br/>
      <input type="submit" value="submit"/>
     </form>
```

firstservlet.java

```
public class firstservlet extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,
IOException
 {
    String str=req.getParameter("na");
    HttpSession ses=req.getSession();
    ses.setAttribute("na", str);
    res.sendRedirect("secondservlet");
 }
}
```

secondservlet.java

```
public class secondservlet extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res) throws IOException
 {
    HttpSession ses=req.getSession();
    String str=ses.getAttribute("na").toString();
    PrintWriter out=res.getWriter();
```



```
        out.print("Welcome"+str);
    }
}
```

Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

- GenericServlet
- ServletInputStream
- ServletOutputStream

1. GenericServlet class

GenericServlet class implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.

GenericServlet class can handle any type of request so it is protocol-independent.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Methods of GenericServlet class

There are many methods in GenericServlet class. They are as follows:

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public String getServletName()** returns the name of the servlet object.

2. ServletInputStream class

ServletInputStream class provides stream to read binary data such as image etc. from the request object. It is an abstract class.

The **getInputStream()** method of **ServletRequest** interface returns the instance of ServletInputStream class. So can be get as:

1. ServletInputStream sin=request.getInputStream();

Method of ServletInputStream class

There are only one method defined in the ServletInputStream class.

1. **int readLine(byte[] b, int off, int len)** it reads the input stream.

3. ServletOutputStream class

ServletOutputStream class provides a stream to write binary data into the response. It is an abstract class.

The **getOutputStream()** method of **ServletResponse** interface returns the instance of ServletOutputStream class. It may be get as:

```
ServletOutputStream out=response.getOutputStream();
```

Methods of ServletOutputStream class

The ServletOutputStream class provides print() and println() methods that are overloaded.

```
void print(datatype name){}

void println(datatype name){}
```

Classes in javax.servlet.http package

There are many classes in javax.servlet.http package. They are as follows:

- HttpServlet
- Cookie

1. HttpServlet class

The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Methods of HttpServlet class

There are many methods in HttpServlet class. They are as follows:

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.

2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.

3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.

4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.

2. Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Open any shopping website.

Select any product and add it to cart.

Again select any other product (this is new request means we are in other page)

Our previous selected product is stored in cart. Here cart behave like a cookie now.

In order to save your data for this session either u can use sesssion or cookie.

Sending

Cookie is class

Cookie obj=new Cokiee("lable",ourdatavariable);

responseobject.addCookie(cookieobject);

Receiving (In second servlet)

Here the client will send all cookies.

Becauase client doesn't know which cookie client a server it is.

To receive all the cookies we use getCookies() method.

```
String str=null;
Cookie obj[]=requestobject.getCookies();
for(Cookie variable: cookiearrayobject)
{
  if(variable.getName().equals("textboxname"))
   {
```

```
            stringvaribale=variable.getValue();
        }
    }
```

Example:

index.html

```
<form method="get" action="firstservlet">
        Enter name<input type="text" name="na" /> <br/>
    <input type="submit" value="submit"/>
    </form>
```

firstservlet.java

```
public class firstservlet extends HttpServlet
{
public void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,
IOException
 {
   String str=req.getParameter("na");
   Cookie ck=new Cookie("na",str);
   res.addCookie(ck);
   res.sendRedirect("secondservlet");
 }
}
```

secondservlet.java

```
public class secondservlet extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res) throws IOException
 {
   Cookie cook[]=req.getCookies();
   String str=null;
   for(Cookie c:cook)
   {
     if(c.getName().equals("na"))
     {
       str=c.getValue();
     }
   }
   PrintWriter out=res.getWriter();
   out.print("Welcome"+str);
 }
```

Java Database Connectivity Steps(JDBC Connectivity using Servlet)

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

- Register the driver class
- Creating connection
- Creating statement
- Executing queries

- Closing connection

1) Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax

Example to register the OracleDriver class

```
Class.forName("com.mysql.jdbc.Driver");
```

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax

```
Connection con=DriverManager.getConnection
                ("jdbc:mysql://localhost:3306/databasename","userame","password");
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax:

```
Statement stmt=con.createStatement();
```

4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax:

```
ResultSet rs=stmt.executeQuery(queryobject);
while(rs.next())
{
  //records will be here
  // To receive database column use rs.getString(1)
1 means first column
}
```

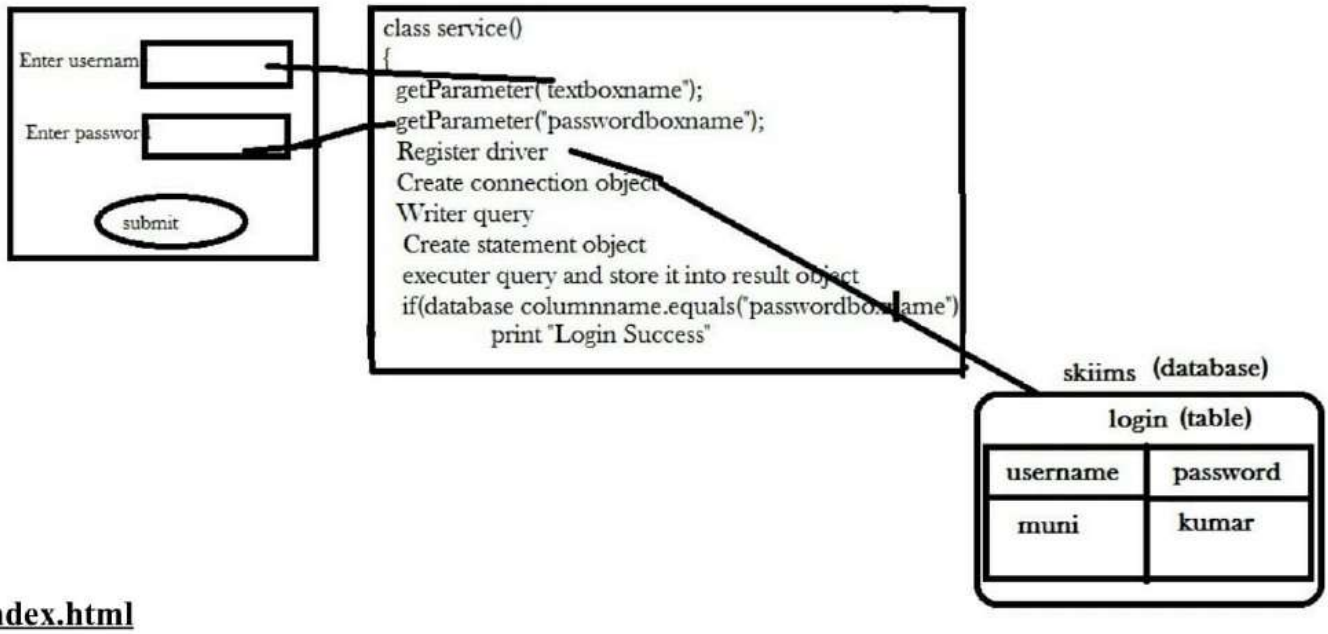
5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax

```
con.close();
```

Login Page using Servlet (Creating Login Page using Servlet with JDBC)



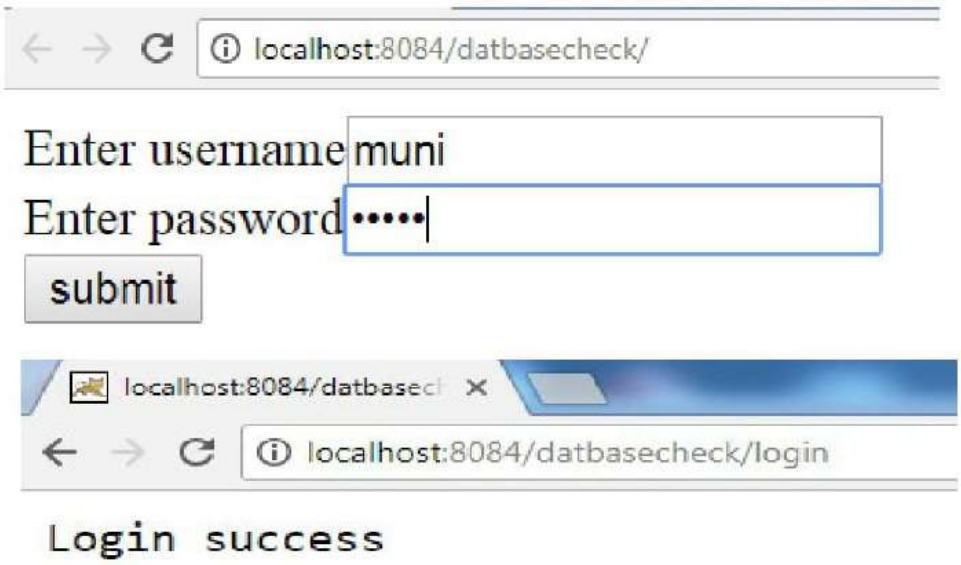
index.html

```
<form method="post" action="login">
        Enter username<input type="text" name="un" /><br/>
        Enter password<input type="password" name="up" /><br/>
        <input type="submit" value="submit" />
</form>
```

login.java

```
public class login extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res) throws IOException
{
 String uname=req.getParameter("un");
 String upass=req.getParameter("up");
 PrintWriter out=res.getWriter();
  try
  {
    Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/muni","root","admin");
    String q="select * from login where username='"+uname+"' ";
    Statement ps=con.createStatement();
    ResultSet rs = ps.executeQuery(q);
    if(rs.next())
    {
      if(rs.getString(2).equals(upass))
        out.println("Login success");
      }
  }
```

```
    catch (ClassNotFoundException | SQLException ex)
    {
        Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
    }
  }
}
```



JSP (Java Server Pages)

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc.

JSP technology is used to create web application just like Servlet technology.

JavaServer Pages (**JSP**) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types.

- JSPs are normal HTML pages with embedded Java code. To process a JSP file, developers need a JSP engine, which is connected to a Web server.
- The JSP page is then compiled into a servlet, which is handled by the servlet engine. This phase is known as translation.
- The servlet engine then loads the servlet class and executes it to create dynamic HTML, which is then sent to the browser.
- When the next page is requested, the JSP page is precompiled into the servlet and executed, unless the JSP page is changed.

Advantage of JSP over Servlet

There are many advantages of JSP over servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet

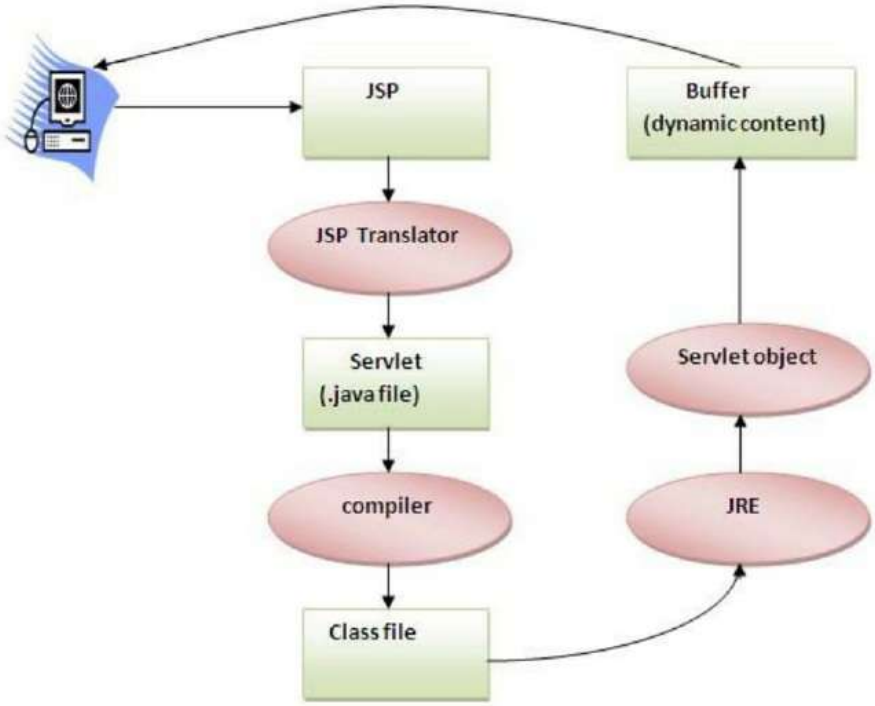
In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

Life cycle of a JSP Page

The JSP pages follows these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (class file is loaded by the classloader)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Reqeust processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into servlet by the help of JSP translator. The JSP translator is a part of webserver that is responsible to translate the JSP page into servlet. Afterthat Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.

JSP Scripting elements

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements

The scripting elements provide the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriplet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
<% java source code %>
```

Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

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

```
<% out.print("welcome to jsp"); %>
</body>
</html>
```

Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

File: **index.html**

```
<html>
<body>
        <form action="welcome.jsp">
                <input type="text" name="uname">
                <input type="submit" value="go"><br/>
        </form>
</body>
</html>
```

File: **welcome.jsp**

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
</form>
</body>
</html>
```

JSP expression tag

It is mainly used to print the values of variable or method.

Syntax

```
<%=  variable or method %>
```

Example:

```
<%
  int a=10;
 int b=20;
 int c=a+b;
%>
<%=  c

 %>
```

Example of JSP expression tag that prints current time

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

index.jsp

```
<html>
<body>
Current Time: <%= java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```

JSP Declaration Tag

The **JSP declaration tag** is used *to declare variables and methods*.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

```
<%!
        variable or method declaration
%>
```

Difference between JSP Scriptlet tag and Declaration tag

Jsp Scriptlet Tag	Jsp Declaration Tag
The jsp scriptlet tag can only declare variables not methods.	The jsp declaration tag can declare variables as well as methods.
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaration tag is placed outside the _jspService() method.

Example of JSP declaration tag that declares field

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag.

index.jsp

```
<html>
<body>
<%!
        int data=50;
 %>
<%=
         "Value of the variable is:"+data
         %>
```

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

Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

index.jsp

```
<html>
<body>
<%!
        int cube(int n)
        {
        return n*n*n*;
        }
%>
<%= "Cube of 3 is:"+cube(3)
 %>
</body>
</html>
```

JSP Implicit Objects

Thee are no.of **jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

Object	**Type**
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
session	HttpSession
exception	Throwable