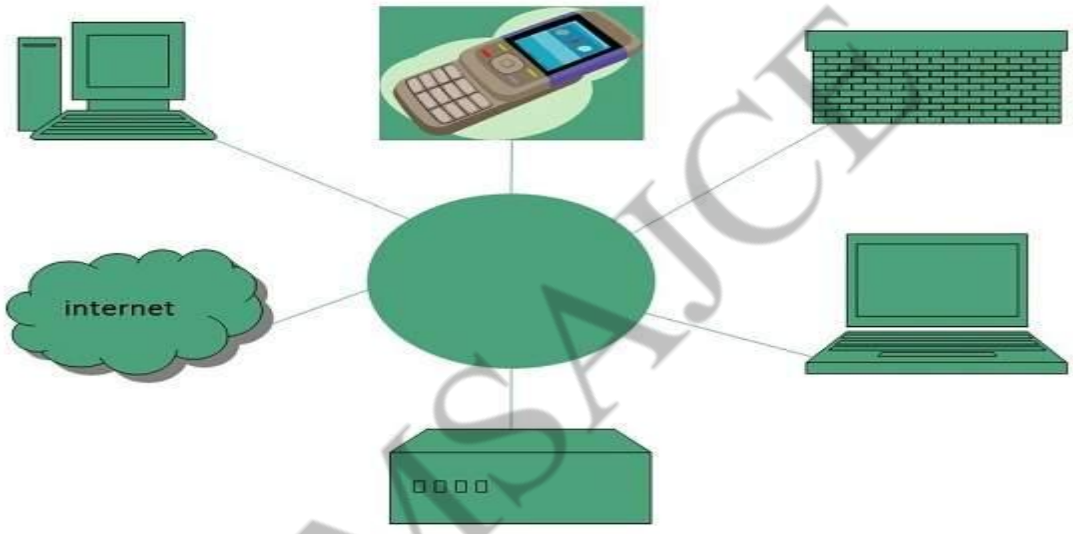
IT3401 Web Essentials

Unit-1 Website Basics

1. Explain Internet overview?

Internet

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



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

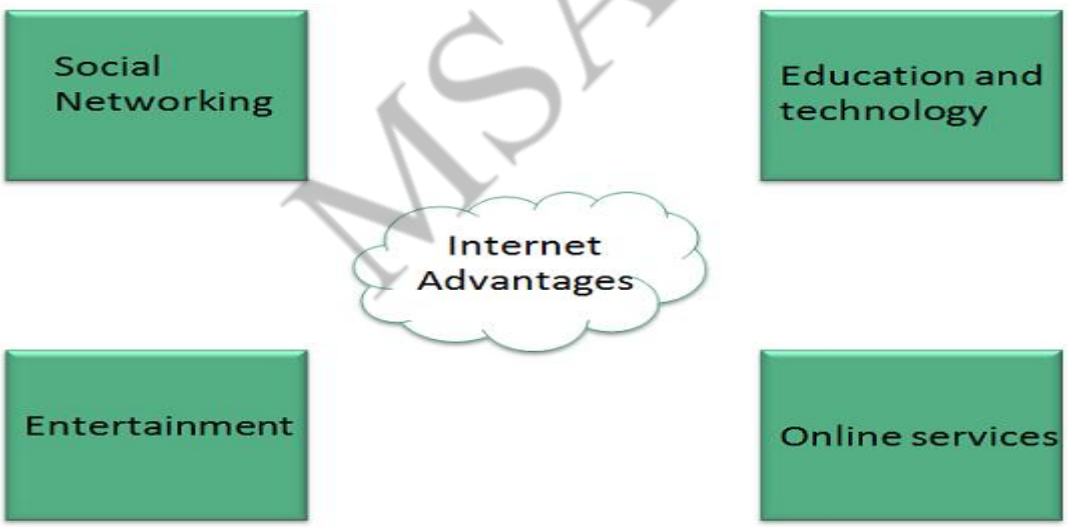
Evolution

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

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

Advantages

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



- Internet allows us to communicate with the people sitting at remote locations. There are various apps available on the wed that uses Internet as a medium for communication. One can find various social networking sites such as:
    - Facebook
    - Twitter
    - Yahoo

    - Google+
    - Flickr
    - Orkut
- One can surf for any kind of information over the internet. Information regarding various topics such as Technology, Health & Science, Social Studies, Geographical Information, Information Technology, Products etc can be surfed with help of a search engine.
- Apart from communication and source of information, internet also serves a medium for entertainment. Following are the various modes for entertainment over internet.
    - Online Television
    - Online Games
    - Songs
    - Videos
    - Social Networking Apps
- Internet allows us to use many services like:
    - Internet Banking
    - Matrimonial Services
    - Online Shopping
    - Online Ticket Booking
    - Online Bill Payment
    - Data Sharing
    - E-mail
- Internet provides concept of **electronic commerce**, that allows the business deals to be conducted on electronic systems

Disadvantages

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

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

2. Describe the Fundamentals of computer networking?

This tutorial will discover the definition of computer networking, its types, computer networks work, networking concepts, commonly used terms, networking and internet, how it works, and network topologies.

Computer Network

"A Computer Network is defined as a set of two or more computers that are linked together?either via wired cables or wireless networks i.e., WiFi?with the purpose of communicating, exchanging, sharing or distributing data, files and resources."

Computer Networks are built using a collection of hardware (such as **routers, switches, hubs, and so forth)** and networking software (such as **operating systems, firewalls, or corporate applications).**

Though one can also define the computer networks based on their geographic location, a **LAN (local area network)** connects computers in a definite physical dimension, such as **home or within an office.**

In contrast, a **MAN (Metropolitan area network)** connects computers ranging between multiple buildings in a city.

The **Internet** is the most significant example of **WAN (Wide Area Network),** connecting billions of networking devices across the world.

One can also describe the concept of computer networking by its communicating protocols, the physical arrangement of its networking elements, how it manages network traffic, and it's functioning.

Computer networks are globally used by businesses, the entertainment industry, education in the research field for communication and transferring their data from source to destination node.

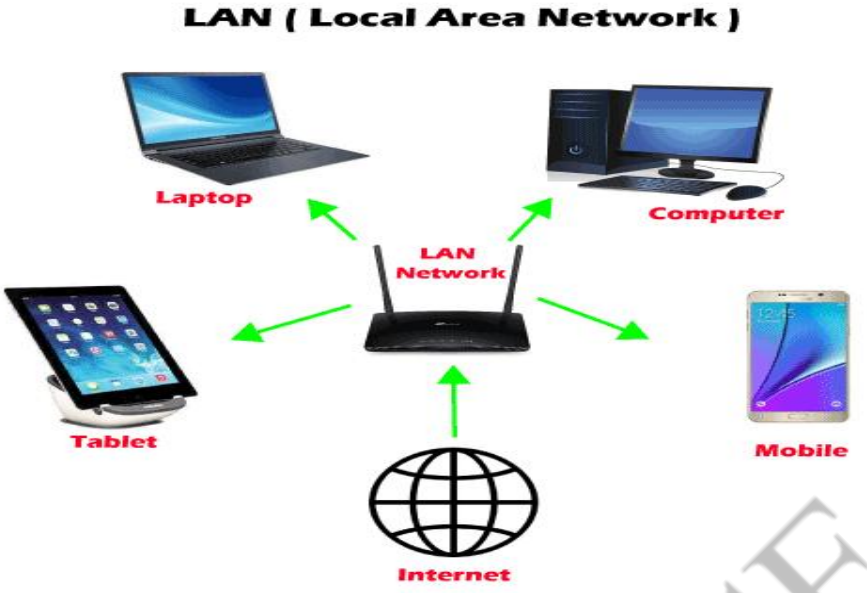
All the other technologies, including the **internet, Google search, instant messaging apps, online video streaming, social media, email, cloud kitchen, cloud data storage,** etc., all exist because of computer networks.

Computer Network Types

Below are the most common computer network types that are frequently used these days:

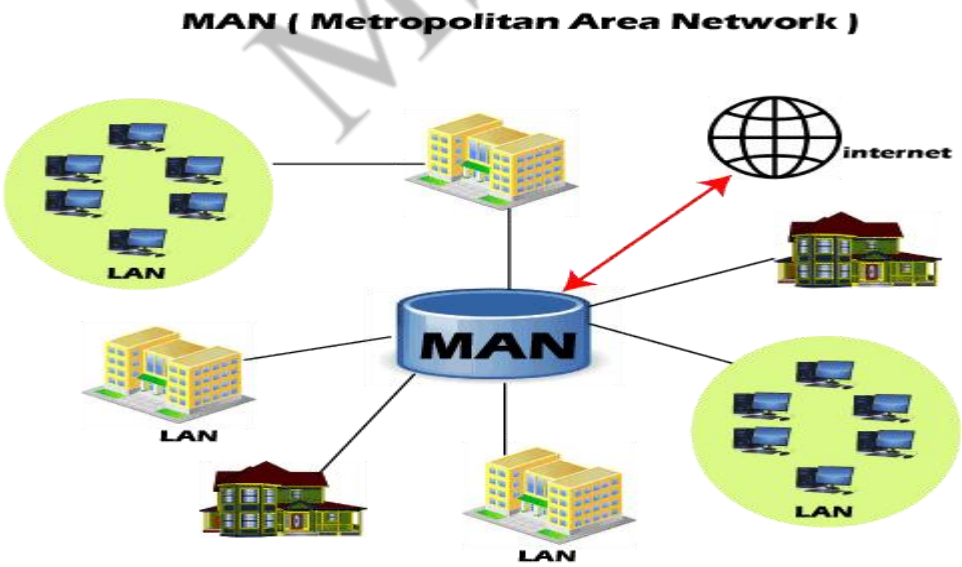
- **LAN [Local Area Network}**
- **WLAN [Wireless local area network]**
- **CAN [Campus Area Network]**
- **MAN [Metropolitan Area Network]**
- **PAN [Personal Area Network]**
- **SAN [Storage Area Network]**
- **VPN [Virtual Private Network]**
- **WAN [Wide Area Network]**

1. LAN



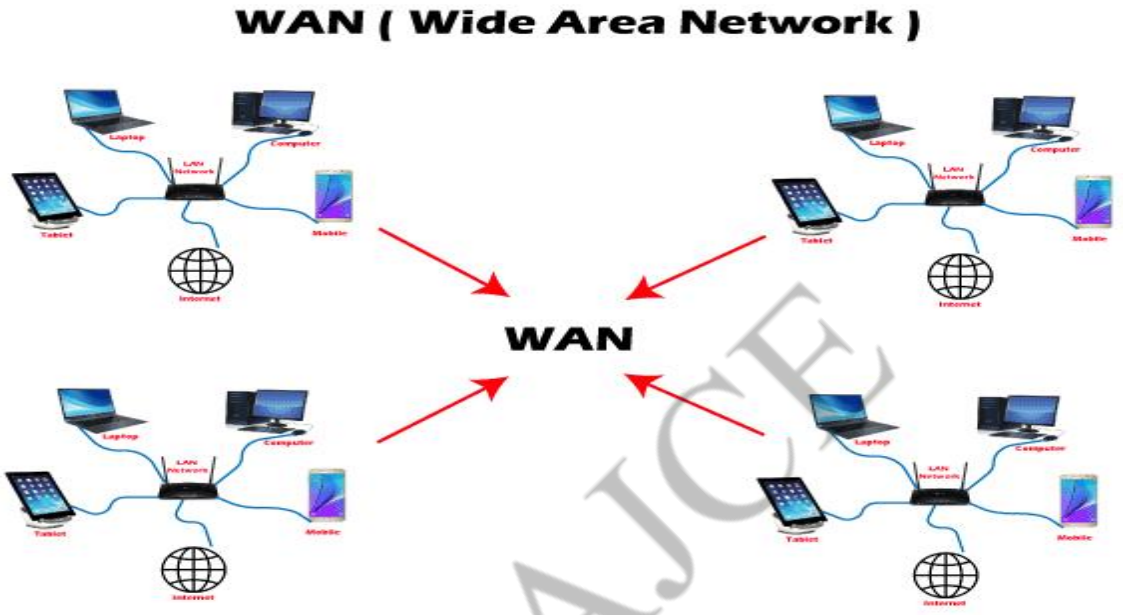
LAN or Local Area Network is a group of devices connecting the computers and other devices such as **switches, servers, printers,** etc., over a short distance such as office, home. The commonly used LAN is **Ethernet LAN.** This network is used as it allows the user to transfer or share data, files, and resources.

2. MAN



MAN or Metropolitan Area Network is typically a more extensive network when compared to LANs but is smaller than WANs. This network ranges between several buildings in the same city. Man networks are connected via fiber optic cable (usually high-speed connection). Cities and government bodies usually manage MANs.

3.WAN



WAN or Wide Area Network is the most significant network type connecting computers over a wide geographical area, such as a country, continent. WAN includes **several LANs, MANs, and CANs.** An example of WAN is the **Internet,** which connects billions of computers globally.

Networking terms and concepts

Some of the most commonly used terms in day-to-day networking life are as discussed below:

1. IP address

An IP address or ***Internet Protocol*** is a **unique number that represents the address where you live on the Internet.** Every device that is connected to the network has a string of numbers or IP addresses unlike house addresses.

You won't find two devices connected to a network with an identical IP address. When your computer sends data to another different, the sent data contains a 'header' that further contains the devices' IP address, i.e., the source computer and the destination device.

2. Nodes

A node refers to a networking **connection point where a connection occurs inside a network that further helps in receiving, transmitting, creating, or storing files or data.**

Multiple devices could be connected to the Internet or network using wired or wireless nodes. To form a network connection, one requires two or more nodes where each node carries its unique identification to obtain access, such as an IP **address.** Some examples of nodes are **computers, printers, modems, switches, etc.**

3. Routers

A router is a **physical networking device, which forwards data packets between networks.** Routers do the data analysis, perform the traffic directing functions on the network, and define the top route for the data packets to reach their destination node. A **data packet** may have to surpass multiple routers present within the network until it reaches its destination.

4. Switches

In a computer network, a switch is a device that **connects other devices and helps in node-to-node communication by deciding the best way of transmitting data within a network (usually if there are multiple routes in a more extensive network).**

Though a router also transmits information, it forwards the information only between networks, whereas a switches forwards data between nodes present in a single network.

Switching is further classified into three types, which are as follows:

- **Circuit Switching**

- **Packet Switching**
- **Message Switching**

- **Circuit Switching:** In this switching type, a secure communication path is established between nodes (or the sender and receiver) in a network. It establishes a dedicated connection path before transferring the data, and this path assures a good transmission bandwidth and prevents any other traffic from traveling on that path. For example, **the Telephone network.**
- **Packet Switching:** With this technique, a message is broken into independent components known as packets. Because of their small size, each packet is sent individually. The packets traveling through the network will have their source and destination IP address.
- **Message Switching:** This switching technique uses the store and forward mechanism. It sends the complete unit of the message from the source node, passing from multiple switches until it reaches its intermediary node. It is not suitable for real-time applications.

5. Ports

A port **allows the user to access multiple applications by identifying a connection between network devices.** Each port is allocated a set of string numbers. If you relate the IP address to a hotel's address, you can refer to ports as the hotel room number. Network devices use port numbers to decide which application, service, or method is used to forward the detailed information or the data.

6. Network cable types

Network cables are used as a **connection medium between different computers and other network devices.** Typical examples of network cable types are **Ethernet cables, coaxial, and fiber optic.** Though the selection of cable type usually depends on the size of the network, the organization of network components, and the distance between the network devices.

Computer Networks and the Internet

The Internet is the major example of a WAN, which connects billions of computers globally. Internet follows standard protocols that facilitate communication between these network devices. Those protocols include:

1. **HTTP (Hypertext Transfer Protocol)**
2. **IP (Internet protocol or IP addresses)**
3. **TCP (Transmission Control Protocol)**
4. **UDP (User Datagram Protocol)**
5. **FTP (File Transfer Protocol)**

ISPs (Internet Service Providers) NSPs (Network Service Providers) effectively support the internet infrastructure. The infrastructure allows the transportation of data packets to the recipient device over the Internet.

Internet is a giant hub of information, but this information is not sent to every computer connected to the Internet. The protocols and infrastructure are responsible for managing to share the precise information the user has requested.

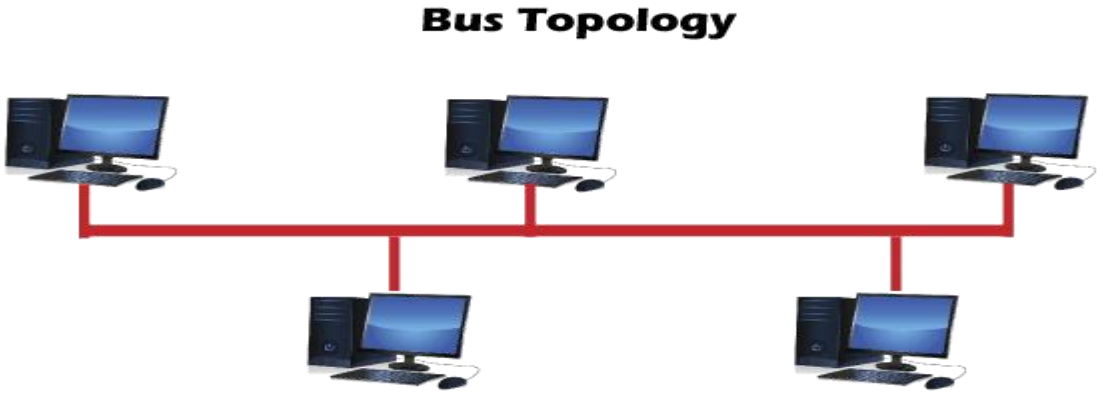
Network Topology

"Network topology is defined as the arrangement of computers or nodes of a computer network to establish communication among all."

A node refers to a device that can transmit, receive, create, or store information. The nodes are connected via a network link that could be either wired (cables, Ethernet) or wireless (Bluetooth, Wi-Fi).

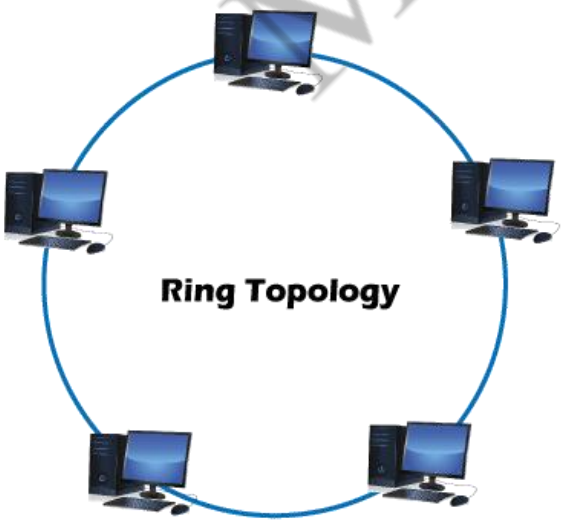
To help build a successful network in different situations, topologies are further classified into several types.
Though there are several topologies but in this tutorial, we will discuss the commonly used ones, which are as follows:

1. Bus Topology



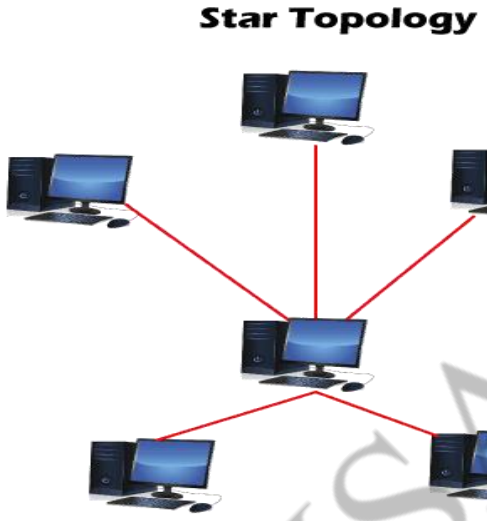
- A ***Bus network topology*** supports a common transmission medium where each node is directly connected with the main network cable.
- The data is transmitted through the main network cable and is received by all nodes simultaneously.
- A signal is generated through the source machine, which contains the address of the receiving machine. The signal travels in both the direction to all the nodes connected to the bus network until it reaches the destination node.
- Bus topology is not fault-tolerant and has a limited cable length.

2. Ring Topology



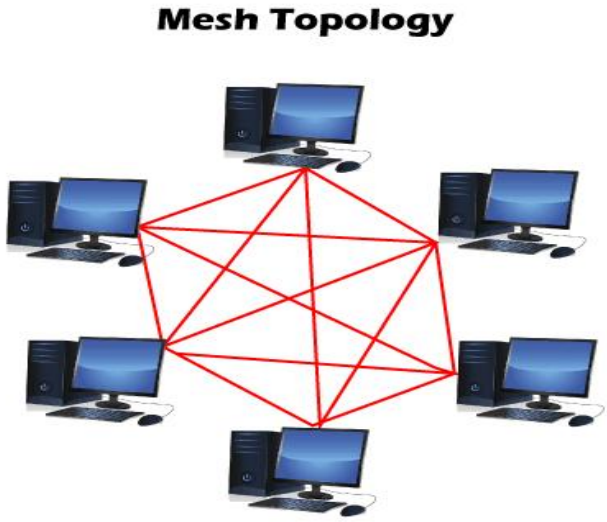
- A ***Ring topology*** is a modified version of bus topology where every node is connected in a closed-loop forming peer-to-peer LAN topology.
- Every node in a ring topology has precisely two connections. The Adjacent node pairs are connected directly, whereas the non-adjacent nodes are indirectly connected via various nodes.
- Ring topology supports a unidirectional communication pattern where sending and receiving of data occurs via **TOKEN.**

3. Star Topology



- In a ***Star network topology,*** every node is connected using a single central hub or switch.
- The hub or switch performs the entire centralized administration. Each node sends its data to the hub, and later hub shares the received information to the destination device.
- Two or more-star topologies can be connected to each other with the help of a repeater.

4. Mesh Topology



- In a ***Mesh topology,*** every node in the network connection is directly connected to one other forming overlapping connections between the nodes.
- This topology delivers better fault tolerance because if any network device fails, it won't affect the network, as other devices can transfer information.
- The Mesh networks self-configure and self-organize, finding the quickest, most secure way to transmit the data.
- One can forpm a full mesh topology by connecting every single node to another node in the network. **Full mesh** is expensive and is only used in the networks, which demands high data redundancy.
- Another type of mesh topology is **partial mesh topology,** where only a few devices are connected, and few are connected to the devices with which they share the most information. This mesh type is applicable in the networks, requiring less redundancy or a cost-effective network topology that is easy to execute.

3. Explain the Types of Network protocols and their uses?

, there are vast numbers of users' communicating with different devices in different languages. That also includes many ways in which they transmit data along with the different software they implement. So, communicating worldwide will not be possible if there were no fixed 'standards' that will govern the way user

communicates for data as well as the way our devices treat those data. Here we will be discussing these standard set of rules.

Yes, we're talking about "protocols" which are set of rules that help in governing the way a particular technology will function for communication. In other words, it can be said that the protocols are digital languages implemented in the form of networking algorithms. There are different networks and network protocols, user's use while surfing.

Types of Protocols

There are various types of protocols that support a major and compassionate role in communicating with different devices across the network. These are:

1. Transmission Control Protocol (TCP)
2. Internet Protocol (IP)
3. User Datagram Protocol (UDP)
4. Post office Protocol (POP)
5. Simple mail transport Protocol (SMTP)
6. File Transfer Protocol (FTP)
7. Hyper Text Transfer Protocol (HTTP)
8. Hyper Text Transfer Protocol Secure (HTTPS)
9. Telnet
10. Gopher

Let's discuss each of them briefly:

1. Transmission Control Protocol (TCP): TCP is a popular communication protocol which is used for communicating over a network. It divides any message into series of packets that are sent from source to destination and there it gets reassembled at the destination.
2. Internet Protocol (IP): IP is designed explicitly as addressing protocol. It is mostly used with TCP. The IP addresses in packets help in routing them through different nodes in a network until it reaches the destination system. TCP/IP is the most popular protocol connecting the networks.
3. User Datagram Protocol (UDP): UDP is a substitute communication protocol to Transmission Control Protocol implemented primarily for creating loss-tolerating and low-latency linking between different applications.
4. Post office Protocol (POP): POP3 is designed for receiving incoming E-mails.

5. Simple mail transport Protocol (SMTP): SMTP is designed to send and distribute outgoing E-Mail.
6. File Transfer Protocol (FTP): FTP allows users to transfer files from one machine to another. Types of files may include program files, multimedia files, text files, and documents, etc.
7. Hyper Text Transfer Protocol (HTTP): HTTP is designed for transferring a hypertext among two or more systems. HTML tags are used for creating links. These links may be in any form like text or images. HTTP is designed on Client-server principles which allow a client system for establishing a connection with the server machine for making a request. The server acknowledges the request initiated by the client and responds accordingly.
8. Hyper Text Transfer Protocol Secure (HTTPS): HTTPS is abbreviated as Hyper Text Transfer Protocol Secure is a standard protocol to secure the communication among two computers one using the browser and other fetching data from web server. HTTP is used for transferring data between the client browser (request) and the web server (response) in the hypertext format, same in case of HTTPS except that the transferring of data is done in an encrypted format. So it can be said that https thwart hackers from interpretation or modification of data throughout the transfer of packets.
9. Telnet: Telnet is a set of rules designed for connecting one system with another. The connecting process here is termed as remote login. The system which requests for connection is the local computer, and the system which accepts the connection is the remote computer.
10. Gopher: Gopher is a collection of rules implemented for searching, retrieving as well as displaying documents from isolated sites. Gopher also works on the client/server principle.

Some Other Protocols

Some other popular protocols act as co-functioning protocols associated with these primary protocols for core functioning. These are:

- ARP (Address Resolution Protocol)
- DHCP (Dynamic Host Configuration Protocol)
- IMAP4 (Internet Message Access Protocol)
- SIP (Session Initiation Protocol)
- RTP (Real-Time Transport Protocol)
- RLP (Resource Location Protocol)
- RAP (Route Access Protocol)
- L2TP (Layer Two Tunnelling Protocol)

- PPTP (Point To Point Tunnelling Protocol)
- SNMP (Simple Network Management Protocol)
- TFTP (Trivial File Transfer Protocol)

4. Explain Web Essentials briefly?

Web Essentials:

Server:

The software that distributes the information and the machine where the information and software reside is called the server.

- provides requested service to client
- e.g., Web server sends requested Web page

Client:

The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.

- initiates contact with server ("speaks first")
- typically requests service from server
- Web: client implemented in browser

Web server:

Software that delivers Web pages and other documents to browsers using the HTTP protocol

Web Page:

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

Website:

A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.

URL:

Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.

Client-server paradigm:

The Client-Server paradigm is the most prevalent model for distributed computing protocols. It is the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.

A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process.The interaction of the process proceeds according to a protocol.

The primary idea of a client/server system is that you have a central repository of information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.

The Internet:

- Medium for communication and interaction in inter connected network.
- Makes information constantly and instantly available to anyone with a connection.

Web Browsers:

- User agent for Web is called a browser: o Internet Explorer

  o Firefox

Web Server:

- Server for Web is called Web server: o Apache (public domain)
  o MS Internet Information Server

Protocol:

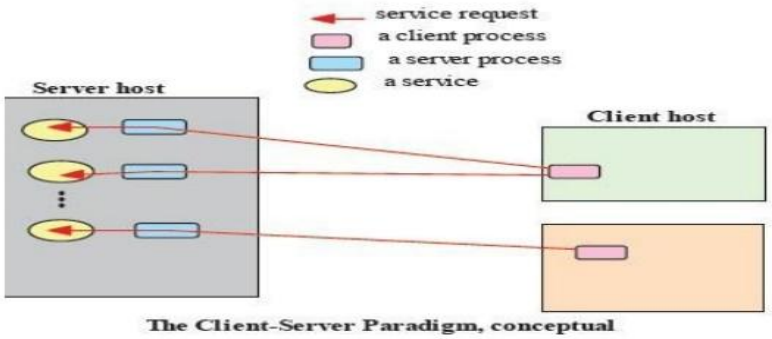
Protocols are agreed formats for transmitting data between devices. The protocol determines:

i. The error checking required

ii. Data compression method used

iii. The way the end of a message is signaled

iv. The way the device indicates that it has received the message



The Client-Server Paradigm, conceptual

Internet Protocol:

There are many protocols used by the Internet and the WWW:

o TCP/IP

o HTTP

o FTP

o Electronic mail protocols IMAP

o POP

TCP/IP

The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn) Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

Hypertext Transfer Protocol (HTTP)

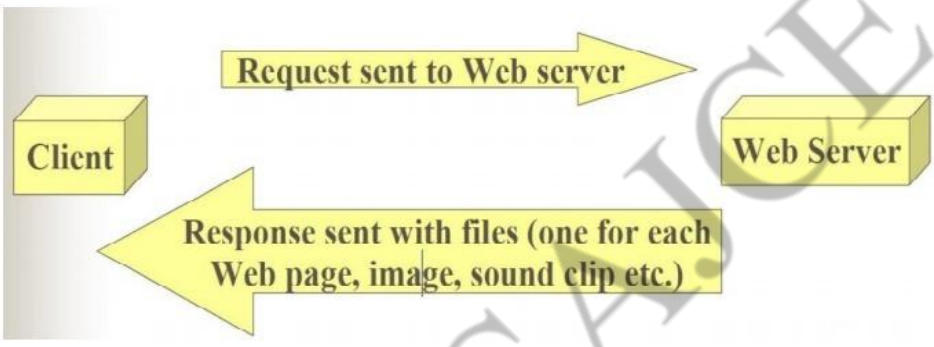
• The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991

• HTTP was designed to transfer pages between machines

• The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client

• The browser connects and requests a page from the server

• The server reads the page from the file system, sends it to the client and terminates the connection.



Electronic Mail Protocols:

• Electronic mail uses the client/server model

• The organisation has an email server devoted to handling email o Stores and forwards email messages

• Individuals use email client software to read and send email

o (e.g. Microsoft Outlook, or Netscape Messenger)

• Simple Mail Transfer Protocol (SMTP)

o Specifies format of mail messages

• Post Office Protocol (POP) tells the email server to:

o Send mail to the user's computer and delete it from the server

o Send mail to the user's computer and do not delete it from the server o Ask whether new mail has arrived.

Interactive Mail Access Protocol (IMAP)

Newer than POP, provides similar functions with additional features.

o e.g. can send specific messages to the client rather than all the messages. A user can view email message headers and the sender's name before

downloading the entire message.

Allows users to delete and search mailboxes held on the email server.

The disadvantage of POP: You can only access messages from one PC.

The disadvantage of IMAP : Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.

World Wide Web: comprises software (Web server and browser) and data (Web sites).

Internet Protocol (IP) Addresses:

- Every node has a unique numeric address
- Form: 32-bit binary number
- New standard, IPv6, has 128 bits (1998)

- Organizations are assigned groups of IPs for their computers

- **Domain names**
  - Form: host-name. domain-names
  - First domain is the smallest (Google)
  - Last domain specifies the type of organization (.com)
  - Fully qualified domain name - the host name and all of the domain names
  - DNS servers - convert fully qualified domain names to IPs

HTTP:

Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents.

A protocol to transfer hypertext requests and information between servers and browsers

Hypertext is text, displayed on a computer, with references (hyperlinks) to

other text that the reader can immediately follow, usually by a mouse HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.

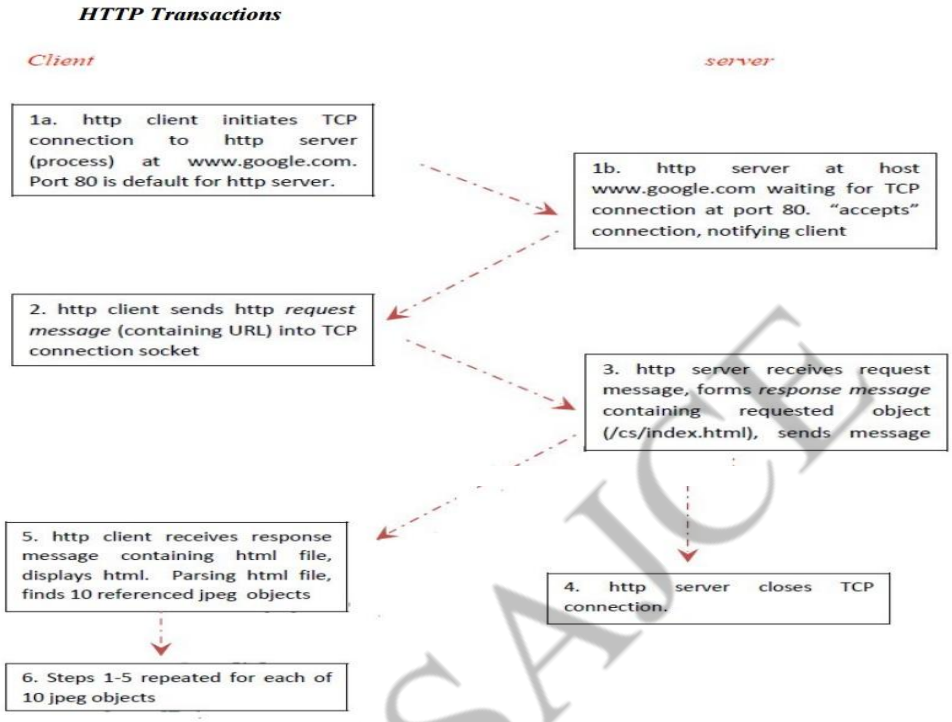
HTTP specifies how clients **request** data, and how servers **respond** to these requests.

The client makes a request for a given page and the server is responsible for finding it and returning it to the client.

The browser connects and requests a page from the server.

The server reads the page from the file system and sends it to the client and then terminates the connection

HTTP Transactions



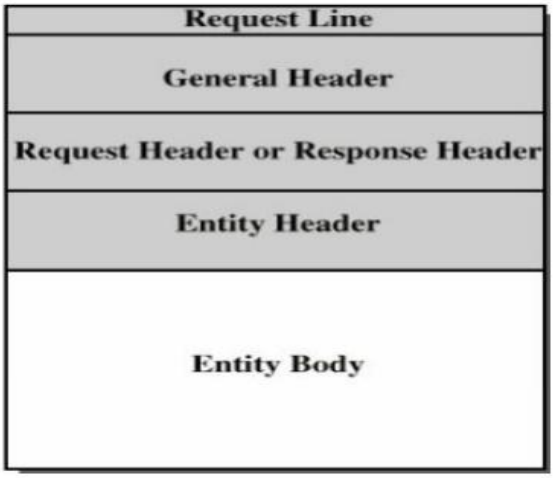
HTTP Message:

HTTP message is the information transaction between the client and server.

Two types of HTTP Message:

1. Requests
    a. Client to server
2. Responses

a. Server to client



Fields

· Request line or Response line

· General header

· Request header or Response header

· Entity header

· Entity body

Request Message:

Request Line:

- A request line has three parts, separated by spaces o a *method* name

  o the local path of the requested resource o the version of HTTP being used

• A typical request line is:

o GET /path/to/file/index.html HTTP/1.1

• Notes:

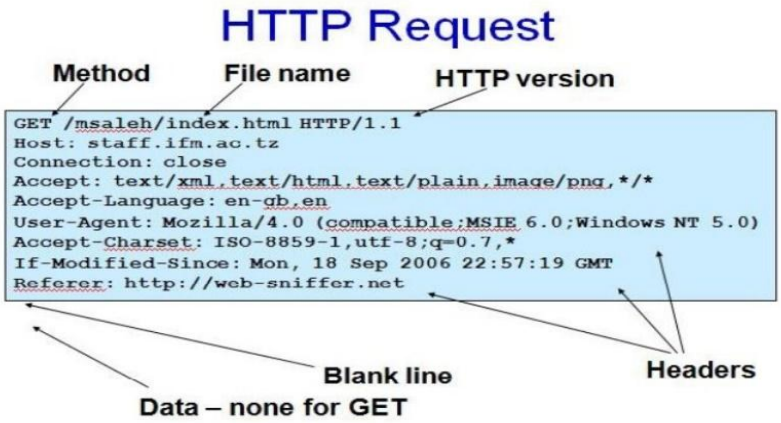
o **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD.** Method names are always uppercase

o The path is the part of the URL after the host name, also called the *request URI* o The HTTP version always takes the form "**HTTP/x.x**", uppercase.

Request Header:

HTTP Request Headers

Header	Description
From	Email address of user
User-Agent	Client s/w
Accept File	File types that client will accept
Accept-encoding	Compression methods
Accept-Language	Languages
Referrer	URL of the last document the client displayed
If-Modified-Since	Return document only if modified since specified
Content-length	Length (in bytes) of data to follow



Response Message:

Response Line:

- A request line has three parts, separated by spaces o the HTTP version,

  o a *response status code* that gives the result of the request, and o an English *reason phrase* describing the status code

- Typical status lines are:

  o HTTP/1.0 200 OK or

  o HTTP/1.0 404 Not Found

- Notes:

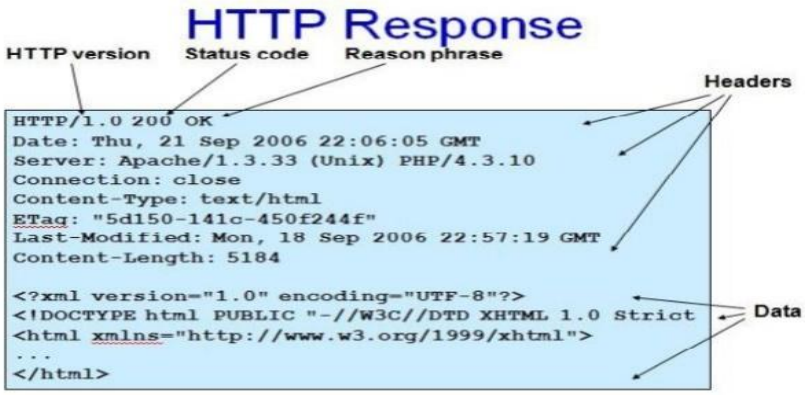
  o The HTTP version is in the same format as in the request line, "**HTTP/x.x**".

  o The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

HTTP Request Header:

HTTP Response Headers

Header	Description
Server	Server software
Date	Current Date
Last-Modified	Modification date of document
Expires	Date at which document expires
Location	The location of the document in redirection responses
Pragma	A hint, e.g., no cache
MIME-version	
Link	URL of document's parent
Content-Length	Length in bytes
Allowed	Requests that user can issue, e.g., GET



EXAMPLE

HTTP Method:

• HTTP method is supplied in the request line and specifies the operation that the client has requested.

Some common methods:

- Options
- Get
- Head
- Post
- Put
- Move
- Delete

Two methods that are mostly used are the GET and POST: o **GET** for queries that can be safely repeated

o **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

The GET Method

• It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components

• Operations have no side effects and GET requests can be re-issued.

• For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.

• Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning

• Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.

• GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]

Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

The POST Method

• Used for operations that have side effects and cannot be safely repeated.

• For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.

If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may be side effects:

The **POST** request message has a content body that is normally used to send parameters and data

• The IIS server returns two status codes in its response for a **POST** request

o The first is **100 Continue** to indicate that it has successfully received the **POST**

request

o The second is **200 OK** after the request has been processed.

HTTP response status codes

• Informational (1xx)

• Successful (2xx)

• Redirection (3xx)

o 301: moved permanently

• Client error (4xx)

- 403 : forbidden
- 404: Not found

• Server error (5xx)

- 503: Service unavailable
- 505: HTTP version not supported

5. Describe client side scripting?

Client side scripting is a process in which the code along with HTML web page is sent to the client by the server. Here, the code refers to the script. In other simple words, client side scripting is a process in which scripts are executed by browsers without connecting the server.

The code executes on the browser of client's computer either during the loading of web page or after the web page has been loaded.

Client side scripting is mainly used for dynamic user interface elements, such as pull-down menus, navigation tools, animation buttons, data validation purpose, etc.

Today, it is rapidly growing and evolving day by day. As a result, writing client side web programming is now easier and faster, thereby, reducing load on the server.

JavaScript and jQuery are by far the most important client-side *scripting languages* or **web scripting languages** and widely used to create a dynamic and responsive webpage and websites.
The browser (temporarily) downloads the code in the local computer and starts processing it without the server. Therefore, the client side scripting is browser dependent.

A **client-side script** is a small program (or set of instructions) that is embedded (or inserted) into a web page. It is processed within the client browser instead of the web server.

The client side script downloads at the client end from the server along with the HTML web page it is embedded in. The web browser interprets and executes the code and then displays the results on the monitor.

The script that executes on the user's computer system is called **client**. It is embedded (or inserted) within the HTML document or can be stored in an external separate file (known as external script).
The script files are sent to the client machine from the web server (or severs) when they are requested. The client's web browser executes the script, then displays the web page, including any visible output from the script.

Look at the below figure to understand better.

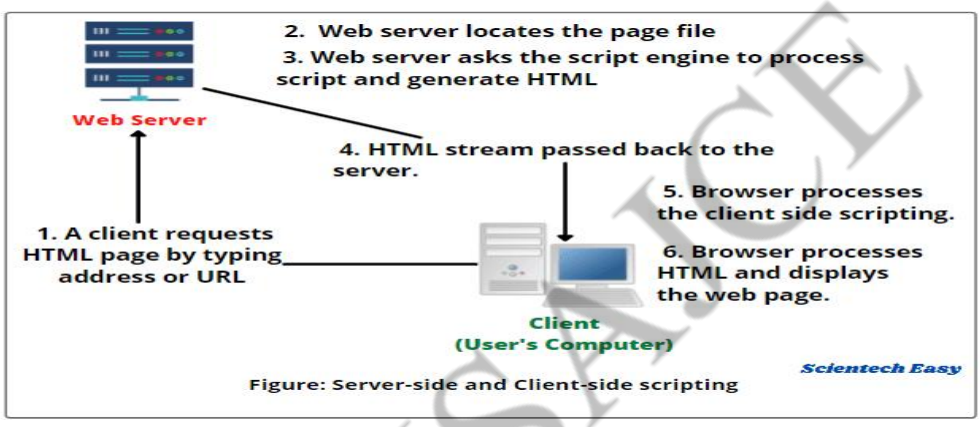


Figure: Server-side and Client-side scripting

Client side scripts may also have some instructions for the web browser to follow in response to certain user actions, such as pressing a page button. They can often be looked if client want to view the source code of web page.

Popular Client side Scripting Language

A language in which a client side script or program is written using syntax is called **client side scripting language** or client side programming.
The most popular client side scripting languages is as follows:

1. JavaScript: It is the most widely client side scripting or programming language. It is based on ECMAScript standard language.

JavaScript is an object based oriented, dynamically typed (or also called weakly typed) scripting language. It runs directly on the browser with the help of an inbuilt interpreter.
Here, weakly typed means the variables are easily converted implicitly from one data type to another.

2. VBScript: This scripting language is developed by Microsoft, based on the Visual Basic. It is basically used to enhance the features of web pages in Internet Explorer. VBScript is interpreted by Internet Explorer web browser.
3. jQuery: jQuery is a fast, small, lightweight JavaScript library. It is used to facilitate a lot of JavaScript code into simple-to-use-functionality.
Most of the biggest companies such as Google, Microsoft, IBM, Netflix, etc. on the Web are using jQuery language.

Client side Scripting Language Example

Let's take a very simple example of JavaScript client side script. In this example, a simple JavaScript client side script will run in the browser to display the name of cities.

The HTML file located on the server will be the same one sent to the browser, but JavaScript changes the HTML web page that is loaded in the browser.

Program code:

```
<DOCTYPE html>

<html>

<head>

<title>List of cities</title>

<script>

  function displayCities()

  {
```

```
    var cities = ["New York", "Dhanbad", "Paris", "London", "Mumbai"];
    var ulElement = document.getElementById("cityList");
    for(var city in cities)
    {
      var listItem = ulElement.appendChild(document.createElement("li"));
            listItem.appendChild(document.createTextNode(cities[city]));
    }
  }
</script>
</head>
<body onload = "displayCities()">
        <ul id = "cityList"></ul>
</body>
</html>
```


Look at the below diagram that shows the flow of data between server and browser.

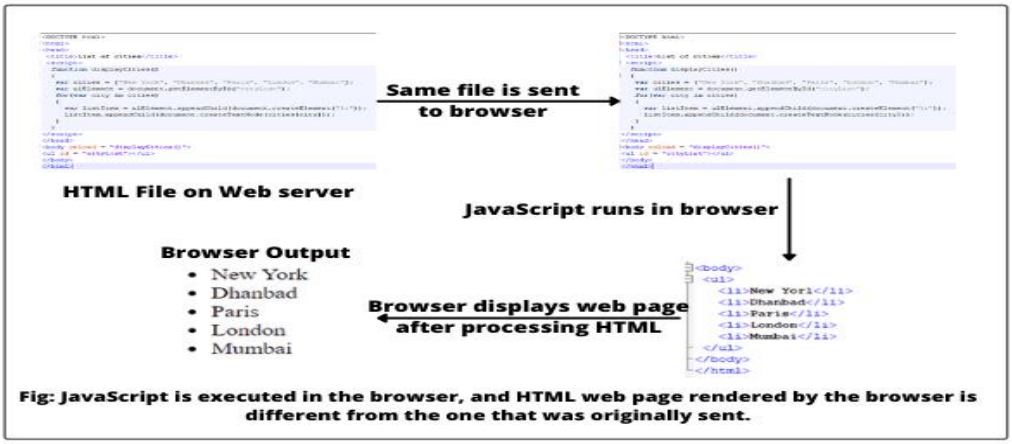


Fig: JavaScript is executed in the browser, and HTML web page rendered by the browser is different from the one that was originally sent.

Application of Client side Scripting

Client side scripting is used to make web pages or website more interactive. It is primarily used at the frontend, where the user can see using the browser. Some important applications of client side scripting are listed, as below:

- To retrieve data from web browser or user's screen.
- Used in the field of online games.
- To customize the web page without reloading the page.
- Client side scripting is used for validation purpose. If the user enters incorrect credentials on the login page, the web page displays an error message on the client machine without submitting it to the web server.
- To create ad banners that interact with the user, rather than simply displaying graphics.
- To create animated images that change when we move the mouse over them.
- Client side script can be used to detect installed plug-ins and notify the user if a plugin is required.

Advantage of Client side Scripting

There are several great advantages of client side scripting that are as follows:

1. The client side scripting language is quite easy to learn and use. It requires minimum programming knowledge or experienced required.

2. The main advantage of client side scripting is that it is lightweight and relatively easy to implement (syntax not too complex). The editing and executing the code is fast.

3. Data processing is done on the client side from the server, which makes it easier to scale applications with large numbers of users. Thereby, load on the server reduces.

4. The client side data validation can be possible using the client side scripting language like JavaScript.

5. The execution of client side script is more quickly because once the script is downloaded from the server, it is executed by the browser directly on the user's computer.

6. Mathematical assessment is also possible using client side scripting.

7. The client side programming helps to perform the complex tasks in relatively few steps.

8. Script code only executed by the browser without connecting the server.

9. It takes too less time to execute script code.

10. Browser respond immediately when user presses any key, mouse movement, clicks, etc.

Disadvantage of Client side Scripting

There are certain disadvantages of client side scripting that are as follows:

1. The main disadvantage of client side scripting is that it is unsecure because the code is sent as is to the client and, therefore, visible to it if the client looks at the sources of his web page. In short, code is usually visible.

2. Client side programming cannot be used if we need to access databases or needs to transmit sensitive data over the internet.

3. There is no guarantee that user has enabled JavaScript on his computer's browser. Therefore, any required functionality must be loaded on the server despite the possibility that it could be offloaded.

4. The smooth running of the script (or program) depends entirely on the client's browser, its configuration, and security level.

5. The web application based on the heavy JavaScript can be complicated to debug and maintain.

6. Client side scripting languages are usually more limited in functionality than server side scripting languages.

Client side Web Attacks

In this section, we will understand web attacks on the client side. They are as follows:

1. Malicious HTML tags embedded in web request can cause the server to generate malformed pages. It can be dangerous if run on the server side. Malformed pages sent back to the client may produce the further problems if executes on the client side.

2. Malicious code can be sent to the server from the attackers in a discussion group website. An example of malicious code can be like this:

```
Hello Group — Here is my message!

<script>Malicious code</script>
```

If JavaScript is enabled on the victim client's browser, the browser will run this code unexpectedly.

3. An attacker can send a file to a client and encourage him to post it to the server. The file may contain malicious code that can hack the website.

4. When a client visits a website, a small text file called a cookie is often placed in the client's computer. At the next visiting, the web server scans that cookie. If it found on the computer, the attacker can use the cookie data to trigger the download of malicious code.

5. Tags like <FORM> can trouble if they are inserted at the wrong place. These HTML tags can change the appearance of the web page.

6. Browsers interpret the information according to the character set chosen by the client. If the client does not specify the character set, the web browser uses the default setting, that can display the garbled appearing or unintended meanings.

General Client side Attack Prevention

There are the following general measures to prevent client side attacks. They are as follows:

1. Client can use client side scripting to clean up form data before it is transmitted.

2. Users can turn off JavaScript functions in the browser. It may disable some web functionality.

3. They should set the security level high in the browser and lower it only for those users you are sure will not violate that trust.

4. Scan all the files including cookie for viruses to prevent the injection of malicious code.

5. Client should declare their character set when configuring browsers.

UNIT-2 Web Desigining

1. Explain the structure of the html webpage with an example?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

A Simple HTML Document

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Example explanation:

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The element defines a paragraph

HTML Element

An HTML element is defined by a start tag, some content, and an end tag:

<tagname> Content goes here... </tagname>

The HTML **element** is everything from the start tag to the end tag:

<h1>My First Heading</h1>

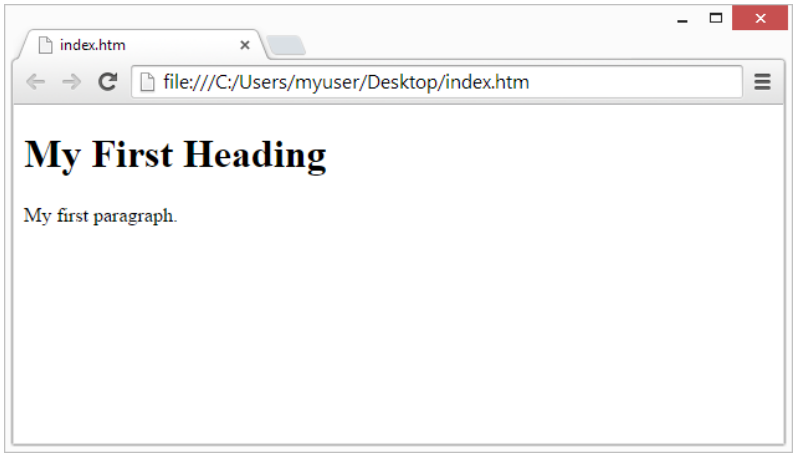
My first paragraph.

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	none	*none*

Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them correctly.

A browser does not display the HTML tags, but uses them to determine how to display the document:



HTML Page Structure

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

2. Define form tag. Design a registration page by using all form controls?

The form is basically used for the registration process, logging into your profile on a website or to create your profile on a website, etc … The information that is collected from the form is -1. Name 2.Email Addresses etc. Now the form will take input from the form and post that data in backend applications (like PHP). So the backend application will process the data which is received by them. There are various form elements that we can use like text fields, text area, drop-down list, select, checkboxes, radio, etc.

Syntax:

<form> Form Content... </form>

Registration page

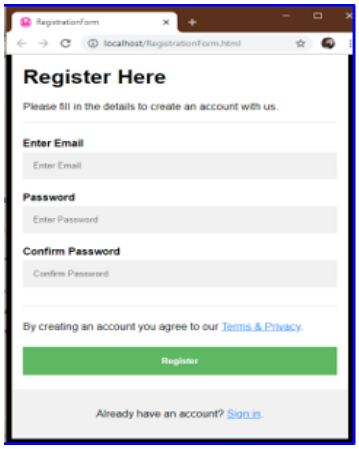
Use a <form> element to process the input.

Example:

```
<form>
<div class="container">
<h1>Register Here</h1>
<p>Please fill in the details to create an account with us.</p>
<hr>
<label for="email"><b>Enter Email</b></label>
<input type="text" placeholder="Enter Email" name="email">
<label for="pwd"><b>Password</b></label>
<input type="password" placeholder="Enter Password" name="pwd">
<label for="confirm"><b>Confirm Password</b></label>
<input type="password" placeholder="Confirm Password" name="confirm">
<hr>
<p>By creating an account you agree to our <a href="#">Terms & Privacy</a>.</p>
<button type="submit" class="registerbtn"><strong>Register</strong></button>
</div>
```

```
<div class="container signin">

<p>Already have an account? <a href="#">Sign in</a>.</p>

</div>

</form>
```

Output:



3. Explain about Cascading Style Sheet with an example?

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

CSS can be added to HTML documents in 3 ways:

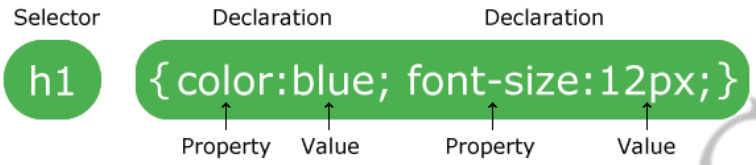
- **Inline** - by using the style attribute inside HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using a <link> element to link to an external CSS file

CSS Example

```
<style>

body {background-color:lightblue; text-align:center;}
h1 {color:blue; font-size:40px;}
p {font-family:verdana; font-size:20px;}

</style>
```

CSS Syntax

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



The selector points to the HTML element to style (h1).

The declaration block (in curly braces) contains one or more declarations separated by semicolons.

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

In the following example all <p> elements will be center-aligned, red and have a font size of 32 pixels:

Example

```
<style>
p {font-size:32px; color:red; text-align:center;}
</style>
```

Same example can also be written like this:

```
<style>
p {
   font-size: 32px;
   color: red;
   text-align: center;
}
</style>
```

CSS Background

CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

1) CSS background-color

The background-color property is used to specify the background color of the element.

```
<!DOCTYPE html>
<html>
<head>
<style>
h2,p{
    background-color: #b0d4de;
}
</style>
</head>
<body>
<h2>My first CSS page.</h2>
<p>Hello Javatpoint. This is an example of CSS background-color.</p>
</body>
</html>
```


2) CSS background-image

The background-image property is used to set an image as a background of an element. By default the image covers the entire element. You can set the background image for a page like this.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-image: url("paper1.gif");
margin-left:100px;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
</body>
</html>
```

3) CSS background-repeat

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("gradient-bg.png");
    background-repeat: repeat-x;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
</body>
</html>
```


4) CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser.

```
background: white url('bbb.gif');
background-repeat: no-repeat;
background-attachment: fixed;
```

5) CSS background-position

The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

You can set the following positions:

1. center
2. top
3. bottom
4. left
5. right

```
background: white url('good-morning.jpg');
background-repeat: no-repeat;
background-attachment: fixed;
background-position: center;
```


4. How to create CSS animation and briefly explain about user interface.?

CSS animation:

Animation is a way using which we can create the illusion of motion by placing still images one after another in a typical format. For example, we can have a ball rising up at some instant than falling down as an animation effect. CSS provides us some properties to control the animation effect by changing some of its variables like timing and key frames etc.

- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-delay`

- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

The @keyframes

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

```
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```


Animation-duration

The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s

Delay an Animation

The animation-delay property specifies a delay for the start of an animation.

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
```

```
  animation-delay: 2s;
}
```

Animation-iteration-count:

The `animation-iteration-count` property specifies the number of times an animation should run.

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 3;
}
```

Animation-direction:

The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- normal - The animation is played as normal (forwards). This is default
- reverse - The animation is played in reverse direction (backwards)
- alternate - The animation is played forwards first, then backwards
- alternate-reverse - The animation is played backwards first, then forwards

The Speed Curve of the Animation:

The animation-timing-function property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- linear - Specifies an animation with the same speed from start to end
- ease-in - Specifies an animation with a slow start
- ease-out - Specifies an animation with a slow end

- ease-in-out - Specifies an animation with a slow start and end
- cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function

```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

The fill-mode For an Animation:

CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.

The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

- none - Default value. Animation will not apply any styles to the element before or after it is executing
- forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- both - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

Animation Shorthand Property:

```
div {
  animation: example 5s linear 2s infinite alternate;
}
```

User interface:

User Interface (UI) defines the way humans interact with the information systems. In Layman's term, User Interface (UI) is a series of pages, screens, buttons, forms and other visual elements that are used to interact with the device. Every app and every website has a user interface. The user interface property is used to change any element into one of several standard user interface elements.

- resize
- outline-offset

Resize Property: The resize property is used to resize a box by user. This property does not apply to inline elements or block elements where overflow is visible. In this property, overflow must be set to “scroll”, “auto”, or “hidden”.

Syntax:

resize: horizontal,vertical,both;

- **horizontal:** This property is used to resize the width of the element.

Syntax:

resize: horizontal;

Supported Browsers: The browser supported by resize property are listed below:

- Apple Safari 4.0
- Google Chrome 4.0
- Firefox 5.0 4.0 -moz-
- Opera 15.0
- Internet Explorer Not Supported

outline-offset: The outline-offset property in CSS is used to set the amount of space between an outline and the edge or border of an element. The space between the element and its outline is transparent.

Syntax:

outline-offset: length;

Supported Browsers: The browser supported by outline-offset property are listed below:

- Apple Safari 3.1
- Google Chrome 4.0
- Firefox 3.5
- Opera 10.5
- Internet Explorer 15.0

UNIT-3

Introduction to JavaScript

Part-b

1. Explain in details about variables and data types in java Script.?

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

1. var a=40;//holding number
2. var b="Rahul";//holding string

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

Variable:

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

1. var x = 10;
2. var _value="sonoo";

Example of JavaScript variable

Let’s see a simple example of JavaScript variable.

```
<script>
var x = 10;
```

```
var y = 20;
var z=x+y;
document.write(z);
</script>
```

Output of the above example

30

JavaScript local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

JavaScript global variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

```
<script>
var data=200;//gloabal variable
function a(){
document.writeln(data);
}
function b(){
```

```
document.writeln(data);
}
a();//calling JavaScript function
b();
</script>
```

2. Explain in detail about operators and operator precedence in java Script.?

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

```
var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10

*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
Delete	Delete Operator deletes a property from the object.
In	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
New	creates an instance (object)

Typeof	checks the type of object.
Void	it discards the expression's return value.
Yield	checks what is returned in a generator by the generator's iterator.

Operator precedence:

Operator precedence describes the order in which operations are performed in an arithmetic expression.

Multiplication (*) and division (/) have higher **precedence** than addition (+) and subtraction (-).

Expressions in parentheses are computed **before** the rest of the expression
Function are executed **before** the result is used in the rest of the expression

Val	Operator	Description	Example
18	( )	Expression Grouping	(100 + 50) * 3
17	.	Member Of	person.name
17	[]	Member Of	person["name"]
17	?.	Optional Chaining ES2020	x ?. y
17	()	Function Call	myFunction()
17	new	New with Arguments	new Date("June 5,2022")
16	new	New without Arguments	new Date()

Increment Operators Postfix increments are executed **before** prefix increments			
15	++	Postfix Increment	i++
15	--	Postfix Decrement	i--
14	++	Prefix Increment	++i
14	--	Prefix Decrement	--i
NOT Operators			
14	!	Logical NOT	!(x==y)
14	~	Bitwise NOT	~x
Unary Operators			
14	+	Unary Plus	+x
14	-	Unary Minus	-x
14	typeof	Data Type	typeof x
14	void	Evaluate Void	void(0)
14	delete	Property Delete	delete myCar.color

3. Brief about various date and math related object with examples?

Math Object

- Math object is a built-in static object.
- It is used for performing complex math operations.

Math Properties

Math Property	**Description**
SQRT2	Returns square root of 2.
PI	Returns Π value.
E \	Returns Euler's Constant.
LN2	Returns natural logarithm of 2.

LN10	Returns natural logarithm of 10.
LOG2E	Returns base 2 logarithm of E.
LOG10E	Returns 10 logarithm of E.

Math Methods

Methods	**Description**
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns cosine of a number.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is base exponent.

Example: Simple Program on Math Object Methods

```
<html>
   <head>
      <title>JavaScript Math Object Methods</title>
   </head>
   <body>
      <script type="text/javascript">
```

```
        var value = Math.abs(20);
        document.write("ABS Test Value : " + value +"<br>");

        var value = Math.acos(-1);
        document.write("ACOS Test Value : " + value +"<br>");

        var value = Math.asin(1);
        document.write("ASIN Test Value : " + value +"<br>");

        var value = Math.atan(.5);
        document.write("ATAN Test Value : " + value +"<br>");
    </script>
  </body>
</html>
```

Output

ABS Test Value : 20
ACOS Test Value : 3.141592653589793
ASIN Test Value : 1.5707963267948966
ATAN Test Value : 0.4636476090008061

2. Date Object

- Date is a data type.
- Date object manipulates date and time.
- Date() constructor takes no arguments.
- Date object allows you to get and set the year, month, day, hour, minute, second and millisecond fields.

Syntax:
var variable_name = new Date();

Example:
var current_date = new Date();

Date Methods

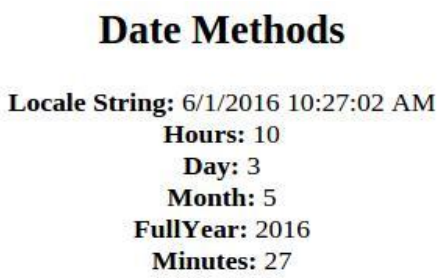
Methods	Description
Date()	Returns current date and time.
getDate()	Returns the day of the month.
getDay()	Returns the day of the week.
getFullYear()	Returns the year.
getHours()	Returns the hour.
getMinutes()	Returns the minutes.
getSeconds()	Returns the seconds.
getMilliseconds()	Returns the milliseconds.
getTime()	Returns the number of milliseconds since January 1, 1970 at 12:00 AM.
getTimezoneOffset()	Returns the timezone offset in minutes for the current locale.
getMonth()	Returns the month.
setDate()	Sets the day of the month.
setFullYear()	Sets the full year.
setHours()	Sets the hours.
setMinutes()	Sets the minutes.
setSeconds()	Sets the seconds.
setMilliseconds()	Sets the milliseconds.
setTime()	Sets the number of milliseconds since January 1, 1970 at 12:00 AM.

setMonth()	Sets the month.
toDateString()	Returns the date portion of the Date as a human-readable string.
toLocaleString()	Returns the Date object as a string.
toGMTString()	Returns the Date object as a string in GMT timezone.
valueOf()	Returns the primitive value of a Date object.

Example : JavaScript Date() Methods Program

```
<html>
   <body>
   <center>
      <h2>Date Methods</h2>
      <script type="text/javascript">
         var d = new Date();
         document.write("<b>Locale String:</b> " + d.toLocaleString()+"<br>");
         document.write("<b>Hours:</b> " + d.getHours()+"<br>");
         document.write("<b>Day:</b> " + d.getDay()+"<br>");
         document.write("<b>Month:</b> " + d.getMonth()+"<br>");
         document.write("<b>FullYear:</b> " + d.getFullYear()+"<br>");
         document.write("<b>Minutes:</b> " + d.getMinutes()+"<br>");
      </script>
   </center>
   </body>
</html>
```

Output:



4. Write a JavaScript code for validating new user registration form which includes the fields like username, password, confirm password, gender, date of birth, contact number and mail id.

```
<!DOCTYPE html>
<html>
<head>
  <title>Reg Form</title>
  <style type="text/css">
    body{
      font-family: Calibri;
    }
    input[type="text"] {
      width: 250px;
    }
    input[type="submit"], input[type="reset"] {
      width: 77px;
      height: 27px;
      position: relative;left: 180px;
    }
    form{
      text-align: center;
      font-family: Calibri;
      font-size: 20px;
      border: 3px solid black;
      width: 600px;
      margin: 20px auto;
    }
```

```
    td {
        padding: 10px;
    }
    td:first-child {
        text-align: right;
        font-weight: bold;
    }
    td:last-child {
        text-align: left;

</style>
<script>
    function validate() {
        var fname = document.reg_form.fname;
        var lname = document.reg_form.lname;
        var address = document.reg_form.address;
        var gender = document.reg_form.gender;
        var email = document.reg_form.email;
        var mobile = document.reg_form.mobile;
        var course = document.reg_form.course;

        if (fname.value.length <= 0) {
            alert("Name is required");
            fname.focus();
            return false;
        }
        if (lname.value.length <= 0) {
```

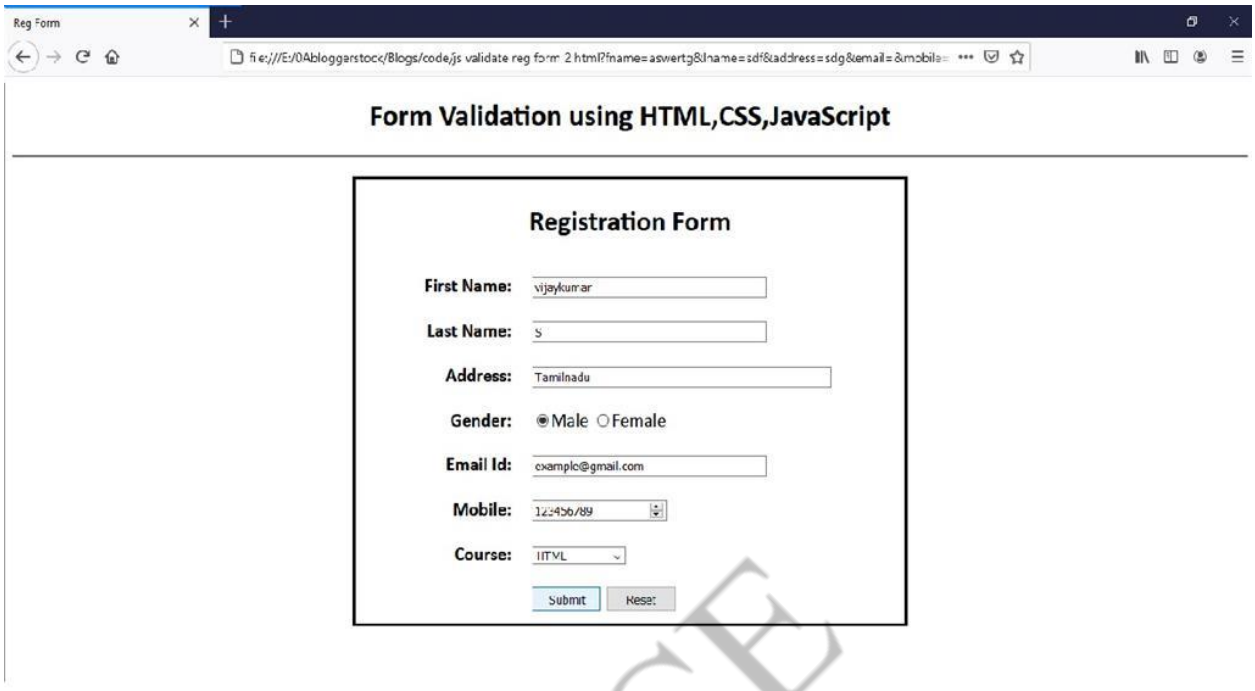
```
    alert("Last Name is required");
    lname.focus();
    return false;
}
if (address.value.length <= 0) {
    alert("Address is required");
    address.focus();
    return false;
}
if (gender.value.length <= 0) {
    alert("Gender is required");
    gender.focus();
    return false;
}
if (email.value.length <= 0) {
    alert("Email Id is required");
    email.focus();
    return false;
}
if (mobile.value.length <= 0) {
    alert("Mobile number is required");
    mobile.focus();
    return false;
}
if (course.value == "select course") {
    alert("Course is required");
    course.focus();
```

```
            return false;
        }
        return false;
    }
  </script>
</head>
<body>
  <center><h1>Form Validation using HTML,CSS,JavaScript</h1></center>
  <hr>
  <form method="" action="" name="reg_form" onsubmit="return validate()">
    <h2>Registration Form</h2>
    <table>
      <tr>
        <td><label>First Name: </label></td>
        <td>
          <input type="text" name="fname" placeholder="First Name">
        </td>
      </tr>
      <tr>
        <td><label>Last Name: </label></td>
        <td>
          <input type="text" name="lname" placeholder="Last Name">
        </td>
      </tr>
      <tr>
        <td><label>Address: </label></td>
        <td>
```

```
      <input type="textarea" size="50" name="address" placeholder="Address">
    </td>
  </tr>
  <tr>
    <td><label>Gender: </label></td>
    <td>
      <input type="radio" name="gender" value="male">Male
      <input type="radio" name="gender" value="femele">Female
    </td>
  </tr>
  <tr>
    <td><label>Email Id: </label></td>
    <td>
      <input type="text" name="email" placeholder="example@gmail.com">
    </td>
  </tr>
  <tr>
    <td><label>Mobile: </label></td>
    <td>
      <input type="number" name="mobile">
    </td>
  </tr>
  <tr>
  <td><label>Course: </label></td>
    <td>
      <select name="course">
        <option value="select course">select course</option>
```

```
            <option value="HTML">HTML</option>
            <option value="CSS">CSS</option>
            <option value="JavaScript">JAVASCRIPT</option>
            <option value="Java">JAVA</option>
          </select>
        </td>
      </tr>
      <tr>
        <td>
          <input type="submit" name="submit" value="Submit">
          <input type="reset" name="reset" value="Reset">
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Output


Reg Form
Form Validation using HTML,CSS,JavaScript
Registration Form
First Name:
vijaykumar
Last Name:
S
Address:
Tamilnadu
Gender:
Male
Female
Email Id:
example@gmail.com
Mobile:
Course:
IITVL
Submit
Resat

UNIT-IV

SERVER-SIDE PROCESSING AND SCRIPTING-PHP

PART-B

1. What is PHP and how it works explain with example?

PHP is an open source programming and server scripting language that is particularly well adapted to Create Dynamic web pages and can be includes in HTML codes. It's used to handle complex content, databases, and session logging, as well as to create entire e-commerce websites. PHP is generally known as Hypertext preprocessor but it was first known as Personal Home Page. It was Designed in 1994 by the programmer named Rasmus Lerdorf. PHP would be relatively simple .PHP is compatible with a variety of databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server, and can communicate with other services through protocols including LDAP, POP3, HTTP, IMAP, and COM. And, PHP operates on a variety of platforms, including Windows, Linux, Unix, and Mac OS X,etc.

Example of PHP Code!

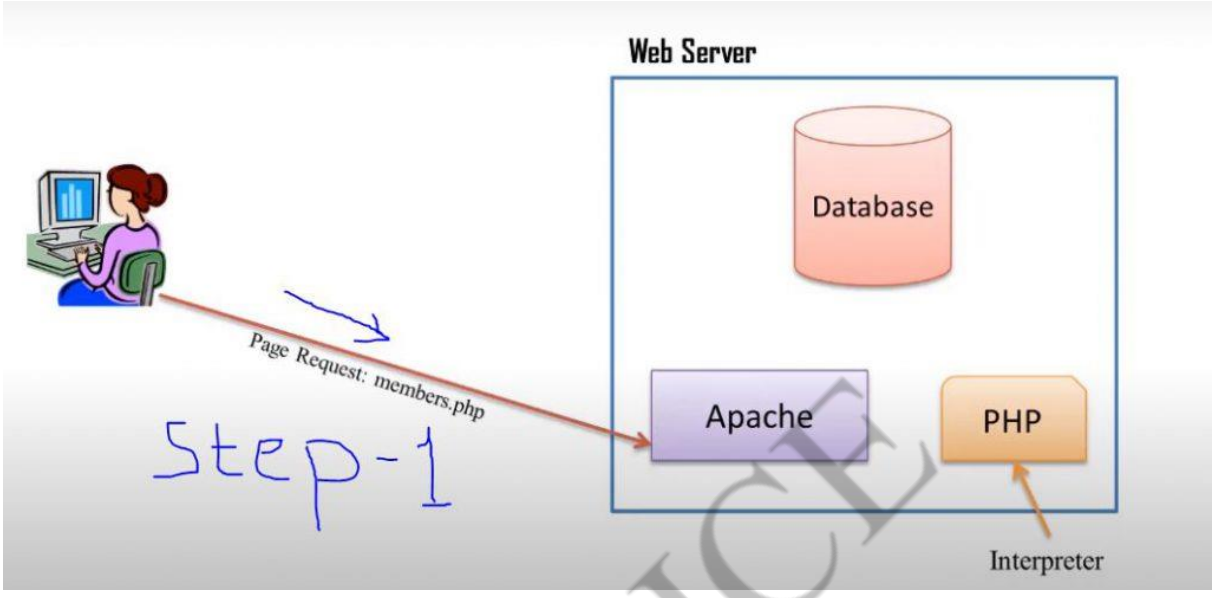
```
<!DOCTYPE html>

<html>

<body>

<?php

echo "My first PHP script!";

?>

</body>

</html>
```

PHP Works:

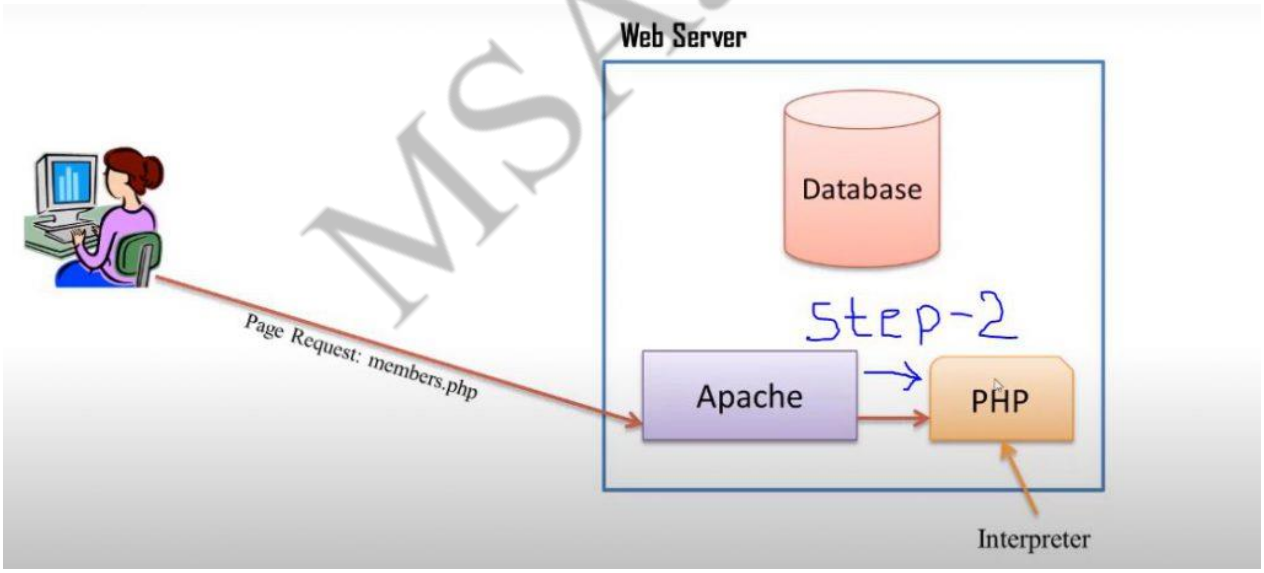
The PHP program interacts with the site server, which is the program that sends out web pages to the rest of the world. When you type a URL into the address bar of your web browser, you're telling the web server at that URL to email you an HTML file. The requested file is sent by the web server in response. The HTML file is read by your browser, and then shows the web page. When you press a source on a web page, you're also requesting a file from the web server. Additionally, when you press a web page button that submits a form, the web server processes a file. When PHP is mounted, the procedure is exactly the same.

You submit a file, and the web server, which happens to be running PHP, responds with HTML, because of PHP it all happens.

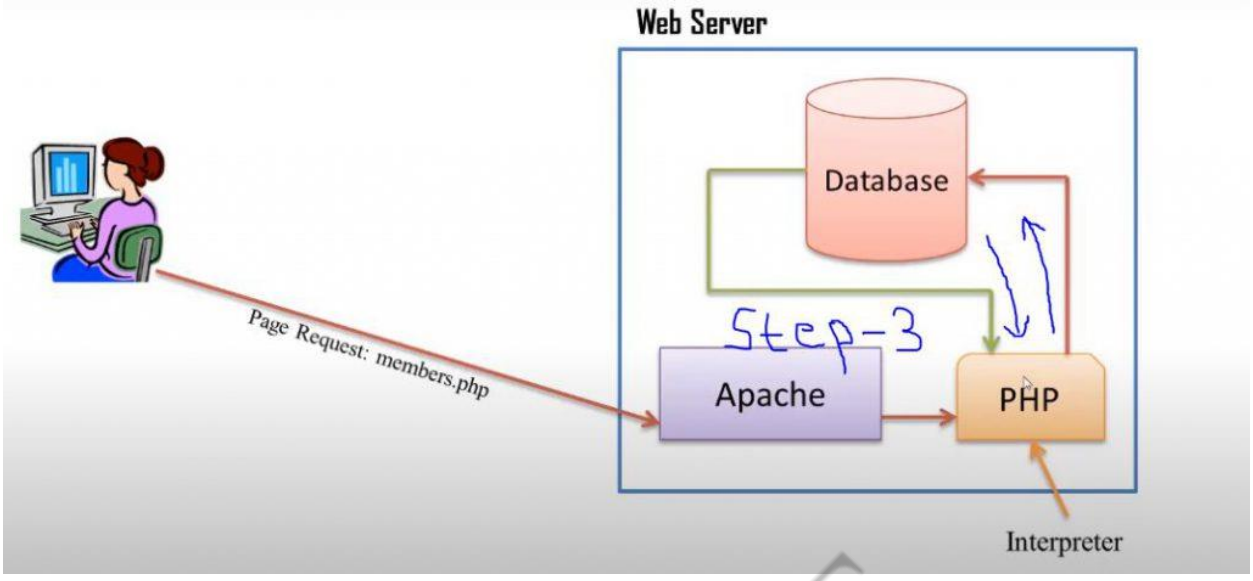
Step 1 – Client send a page request to the web server.



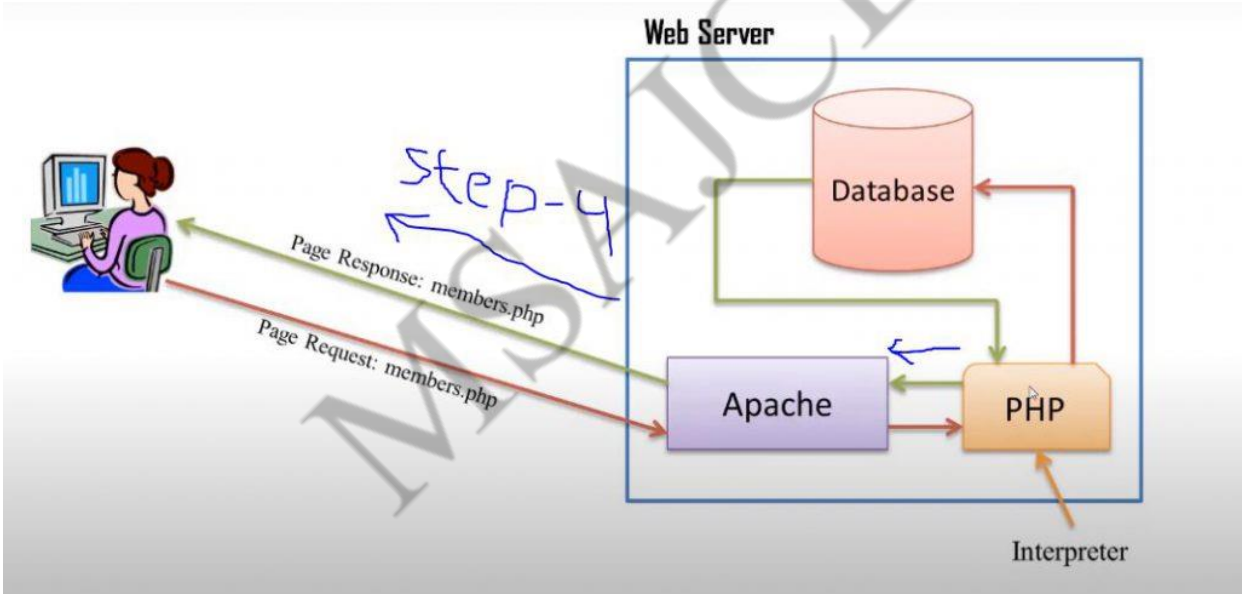
Step 2 – Web server forwards that request to the PHP interpreter.



Step 3 – Now PHP interpreter will take the Date from Database and responce it back to the Web server.



Step 4 – At last Web server responce to the client who has asked for the page request.



2. Explain PHP variables and constants?

Variables are "containers" for storing information.

PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable

Example:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)
- PHP variable names are case-sensitive!

Output variables:

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example:

```
<?php
$txt = "Msajce";
echo "I love $txt!";
?>
```

PHP Variables :

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable :

- local
- global
- static

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Unlike variables, constants are automatically global across the entire script.

Constants are like variables except that once they are defined they cannot be changed or undefined.

Create a PHP Constant:

To create a constant, use the define() function.

Syntax

define(*name*, *value*, *case-insensitive*)

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example:

```
<?php
// case-sensitive constant name
define("GREETING", "Welcome to msajce!");
echo GREETING;
?>
```

o/p:

Welcome to msajce!

3. Explain PHP operators with example?

PHP Operators:

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators:

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result	S
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE	
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7	

4. Explain Flow controls and Looping in PHP? With examples.

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

Conditional statements are flow controls.

PHP Conditional Statements

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The `if` statement executes some code if one condition is true.

Syntax

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

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
}
?>
```


PHP - The if...else Statement

The `if...else` statement executes some code if a condition is true and another code if that condition is false.

Syntax

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

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

PHP - The if...elseif...else Statement

The `if...elseif...else` statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```


The PHP switch Statement

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

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

Example:

```php
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- `while` - loops through a block of code as long as the specified condition is true
- `do...while` - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- `for` - loops through a block of code a specified number of times
- `foreach` - loops through a block of code for each element in an array

The PHP while Loop

The `while` loop executes a block of code as long as the specified condition is true.

Syntax

```php
while (condition is true) {
  code to be executed;
}
```

Example

```php
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

o/p:

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

The PHP do...while Loop

The `do...while` loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

Example

```php
<?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

o/p:

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

The PHP for Loop

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

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

o/p:

The number is: 0
The number is: 1
The number is: 2
The number is: 3
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

The PHP foreach Loop

The `foreach` loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

o/p:

red
green
blue
yellow

5. Explain arrays and strings in PHP?

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:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

An array can hold many values under a single name, and you can access the values by referring to an index number.

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

PHP String Functions:

strlen() - Return the Length of a String

The PHP `strlen()` function returns the length of a string.

Example

Return the length of the string "Hello world!":

```php
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

str_word_count() - Count Words in a String

The PHP `str_word_count()` function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```php
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

strrev() - Reverse a String

The PHP `strrev()` function reverses a string.

Example

Reverse the string "Hello world!":

```php
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

strpos() - Search For a Text Within a String

The PHP `strpos()` function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```php
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

str_replace() - Replace Text Within a String

The PHP `str_replace()` function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```php
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```


6. Explain the functions in PHP?

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Create a User Defined Function in PHP

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

Syntax

function *functionName*(){

    code to be exeuted;

}

A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

Example

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

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

PHP Function Arguments

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 familyName() 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:

Example

```php
<?php
function familyName($fname) {
  echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the `strict` declaration, it will throw a "Fatal Error" if the data type mismatches.

In the following example we try to send both a number and a string to the function without using `strict`:

Example

```php
<?php
function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```


PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```php
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

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

Example

```php
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $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);
?>
```

PHP Return Type Declarations

PHP 7 also supports Type Declarations for the `return` statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

To declare a type for the function return, add a colon (`:`) and the type right before the opening curly (`{`)bracket when declaring the function.

In the following example we specify the return type for the function:

Example

```php
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
  return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the `&` operator is used:

Example

Use a pass-by-reference argument to update a variable:

```php
<?php
function add_five(&$value) {
  $value += 5;
}

$num = 2;
add_five($num);
echo $num;
?>
```

7. Explain File Handling and uploading in PHP?

PHP file handling:

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP Manipulating Files:

PHP has several functions for creating, reading, uploading, and editing files.

:

PHP readfile() Function

The `readfile()` function reads a file and writes it to the output buffer.

Assume we have a text file called "webdictionary.txt", stored on the server, that looks like this:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The PHP code to read the file and write it to the output buffer is as follows (the `readfile()` function returns the number of bytes read on success):

Example

```php
<?php
echo readfile("webdictionary.txt");
?>
```


PHP Open File - fopen()

A better method to open files is with the `fopen()` function. This function gives you more options than the `readfile()` function.

We will use the text file, "webdictionary.txt", during the lessons:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

The first parameter of `fopen()` contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened. The following example also generates a message if the fopen() function is unable to open the specified file:

Example

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

Modes	Description
r	**Open a file for read only.** File pointer starts at the beginning of the file
w	**Open a file for write only.** Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only.** The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only.** Returns FALSE and an error if file already exists
r+	**Open a file for read/write.** File pointer starts at the beginning of the file
w+	**Open a file for read/write.** Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write.** The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write.** Returns FALSE and an error if file already exists

PHP Read File - fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

The following PHP code reads the "webdictionary.txt" file to the end:

fread($myfile,filesize("webdictionary.txt"));

PHP Close File - fclose()

The fclose() function is used to close an open file.

It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!

The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

```php
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

PHP Read Single Line - fgets()

The fgets() function is used to read a single line from a file.

The example below outputs the first line of the "webdictionary.txt" file:

Example

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file.

The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached:

Example

```php
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files.

If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a).

The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides:

Example

```php
$myfile = fopen("testfile.txt", "w")
```


PHP Write to File - fwrite()

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt":

Example

```php
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

Example

```php
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```


PHP Append Text

You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.

Example

```php
<?php
$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");
$txt = "Donald Duck\n";
fwrite($myfile, $txt);
$txt = "Goofy Goof\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP File Upload:

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file_uploads directive.

8. Explain E-mail basics in PHP?

PHP Mail Introduction

The mail() function allows you to send emails directly from a script.

Requirements

For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file.

Installation

The mail functions are part of the PHP core. There is no installation needed to use these functions.

Runtime Configuration

The behavior of the mail functions is affected by settings in php.ini:

Name	Default	Description	Changeable
mail.add_x_header	"0"	Add X-PHP-Originating-Script that will include UID of the script followed by the filename. For PHP 5.3.0 and above	PHP_INI_PERDIR
mail.log	NULL	The path to a log file that will log all mail() calls. Log include full path of script, line number, To address and headers. For PHP 5.3.0 and above	PHP_INI_PERDIR
SMTP	"localhost"	Windows only: The DNS name or IP address of the SMTP server	PHP_INI_ALL
smtp_port	"25"	Windows only: The SMTP port number. For PHP 4.3.0 and above	PHP_INI_ALL
sendmail_from	NULL	Windows only: Specifies the "from" address to be used when sending mail from mail()	PHP_INI_ALL
sendmail_path	"/usr/sbin/sendmail -t -i"	Specifies where the sendmail program can be found. This directive works also under Windows. If set, SMTP, smtp_port and sendmail_from are ignored	PHP_INI_SYSTEM

PHP Mail Functions

Function	Description
ezmlm_hash()	Calculates the hash value needed by EZMLM
mail()	Allows you to send emails directly from a script

PHP mail() Function

Example

Send a simple email:

```php
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```


Definition and Usage

The mail() function allows you to send emails directly from a script.

Syntax

```
mail(to,subject,message,headers,parameters);
```

Parameter Values

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php $txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

9. Explain Database with PHP in Detail?

PHP MySQL Database

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP.

MySQL

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Databases are useful for storing information categorically. A company may have a database with the following tables:

- Employees
- Products
- Customers
- Orders

PHP + MySQL Database System

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Database Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

The query above selects all the data in the "LastName" column from the "Employees" table.

MySQL is the de-facto standard database system for web sites with HUGE volumes of both data and end-users (like Facebook, Twitter, and Wikipedia).

Another great thing about MySQL is that it can be scaled down to support embedded database applications.

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

Example (MySQLi Object-Oriented)

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```php
$conn->close();
```

MySQLi Procedural:

```php
mysqli_close($conn);
```

PDO:

```php
$conn = null;
```

UNIT-V

Servlets and Database Connectivity

1. EXPLAIN JAVA SERVLET ARCHITECTURE?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

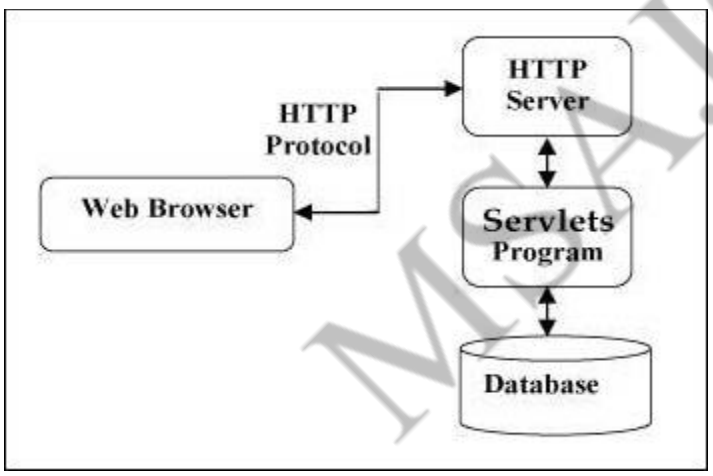
Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI.

- Performance is significantly better.
- Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
- Servlets are platform-independent because they are written in Java.

- Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
- The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

Servlets Architecture

The following diagram shows the position of Servlets in a Web Application.



Servlets Tasks

Servlets perform the following major tasks –

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlets Packages

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages, which are a standard part of the Java's enterprise edition, an

expanded version of the Java class library that supports large-scale development projects.

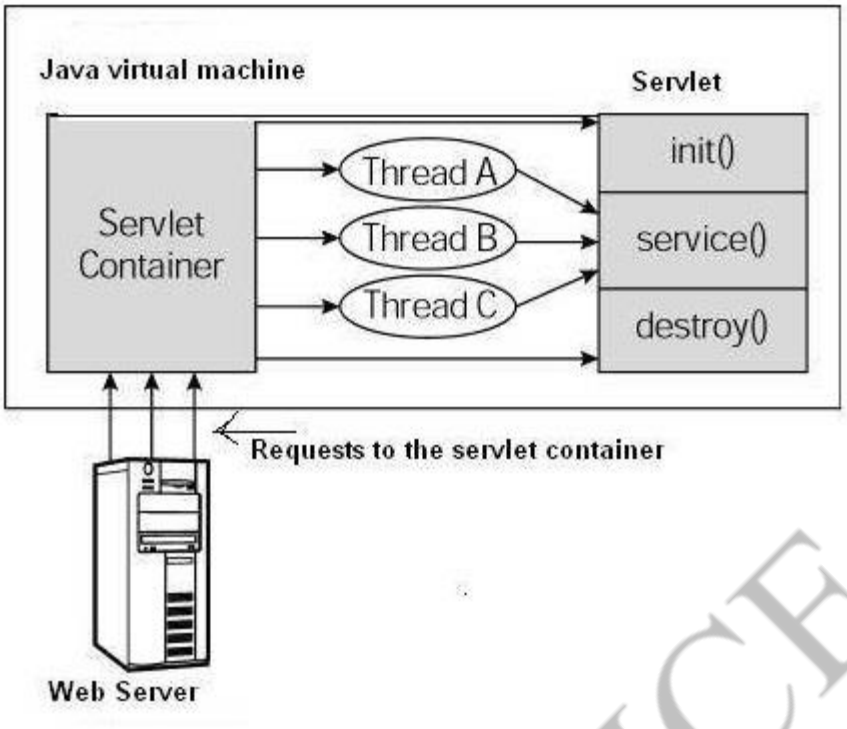
These classes implement the Java Servlet and JSP specifications. At the time of writing this tutorial, the versions are Java Servlet 2.5 and JSP 2.1.

Java servlets have been created and compiled just like any other Java class. After you install the servlet packages and add them to your computer's Classpath, you can compile servlets with the JDK's Java compiler or any other current compiler.

Architecture Diagram

The following figure depicts a typical servlet life-cycle scenario.

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



Advantages Of Servlets :

1. As servlets support all protocols like FTP, SMTP, HTTP etc. they can be used to develop any kind of web applications like E-commerce, Content management systems, chat based or file based web applications etc.
2. As servlets are fully compatible with Java, you can make use of wide range of available Java APIs inside the servlets.
3. As they run on Java enabled servers, You need not to worry about garbage collection and memory leaks. JVM handles them for you.
4. Since servlets are written in Java, they are portable and platform independent. You can run them on any

operating systems and on any web servers available today.

5. Servlets inherit security features from JVM and web server.
6. As servlets are written in Java, you can extend them according to your requirements.
7. As servlets are compiled into bytecodes, they are faster than any other server-side scripting languages.

2.EXPLAIN SERVLET LIFECYCLE.?

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

Now let us discuss the life cycle methods in detail.

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this −

```
public void init() throws ServletException {
  // Initialization code...
}
```

The service() Method

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.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method −

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this −

```
public void destroy() {
  // Finalization code...
}
```

3.Explain Form GET and POST actions in servlets?

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 **?** (question mark) symbol as follows –

http://www.test.com/hello?key1 = value1&key2 = value2

The GET method is the default method to pass information from browser to web server and it produces a long string that appears in your browser's Location:box. Never use the GET method if you have password or other sensitive information to pass to the server. The GET method has size limitation: only 1024 characters can be used in a request string.

This information is passed using QUERY_STRING header and will be accessible through QUERY_STRING environment variable and Servlet handles this type of requests using **doGet()** method.

POST Method

A generally more reliable method of passing information to a backend program is the POST method. This packages the information in exactly the same way as GET method, but instead of sending it as a text string after a ? (question mark)

in the URL it sends it as a separate message. This message comes to the backend program in the form of the standard input which you can parse and use for your processing. Servlet handles this type of requests using **doPost()** method.

Reading Form Data using Servlet

Servlets handles form data parsing automatically using the following methods depending on the situation −

- **getParameter()** − You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues()** − Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames()** − Call this method if you want a complete list of all parameters in the current request.

GET Method Example using URL

Here is a simple URL which will pass two values to HelloForm program using GET method.

http://localhost:8080/HelloForm?first_name = ZARA&last_name = ALI

Given below is the **HelloForm.java** servlet program to handle input given by web browser. We are going to use **getParameter()** method which makes it very easy to access passed information −

```
// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Extend HttpServlet class
public class HelloForm extends HttpServlet {

  public void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {

    // Set response content type
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    String title = "Using GET Method to Read Form Data";
    String docType =
```

```
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";


    out.println(docType +
      "<html>\n" +
        "<head><title>" + title + "</title></head>\n" +
        "<body bgcolor = \"#f0f0f0\">\n" +
          "<h1 align = \"center\">" + title + "</h1>\n" +
          "<ul>\n" +
            "  <li><b>First Name</b>: "
            + request.getParameter("first_name") + "\n" +
            "  <li><b>Last Name</b>: "
            + request.getParameter("last_name") + "\n" +
          "</ul>\n" +
        "</body>" +
      "</html>"
    );
  }
}
```

Assuming your environment is set up properly, compile HelloForm.java as follows −

$ javac HelloForm.java

If everything goes fine, above compilation would produce **HelloForm.class** file. Next you would have to copy this class file in <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
  <servlet-name>HelloForm</servlet-name>
  <servlet-class>HelloForm</servlet-class>
</servlet>


<servlet-mapping>
  <servlet-name>HelloForm</servlet-name>
  <url-pattern>/HelloForm</url-pattern>
</servlet-mapping>
```

Now type *http://localhost:8080/HelloForm?first_name=ZARA&last*

_*name=ALI* in your browser's Location:box and make sure you already started tomcat server, before firing above command in the browser. This would generate following result –

Using GET Method to Read Form Data

- **First Name**: ZARA
- **Last Name**: ALI

GET Method Example Using Form

Here is a simple example which passes two values using HTML FORM and submit button. We are going to use same Servlet HelloForm to handle this input.

```
<html>
  <body>
    <form action = "HelloForm" method = "GET">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>
  </body>
```

</html>

Keep this HTML in a file Hello.htm and put it in <Tomcat-installationdirectory>/webapps/ROOT directory. When you would access *http://localhost:8080/Hello.htm*, here is the actual output of the above form.

First Name: Last Name:

Try to enter First Name and Last Name and then click submit button to see the result on your local machine where tomcat is running. Based on the input provided, it will generate similar result as mentioned in the above example.

POST Method Example Using Form

Let us do little modification in the above servlet, so that it can handle GET as well as POST methods. Below is **HelloForm.java** servlet program to handle input given by web browser using GET or POST methods.

```
// Import required java libraries

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;
```

```
// Extend HttpServlet class
public class HelloForm extends HttpServlet {

  // Method to handle GET method request.
  public void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {

    // Set response content type
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    String title = "Using GET Method to Read Form Data";
    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";

    out.println(docType +
      "<html>\n" +
```

```
        "<head><title>" + title + "</title></head>\n" +
        "<body bgcolor = \"#f0f0f0\">\n" +
          "<h1 align = \"center\">" + title + "</h1>\n" +
          "<ul>\n" +
            "  <li><b>First Name</b>: "
            + request.getParameter("first_name") + "\n" +
            "  <li><b>Last Name</b>: "
            + request.getParameter("last_name") + "\n" +
          "</ul>\n" +
        "</body>"
      "</html>"
    );
  }

  // Method to handle POST method request.
  public void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
```

```
    doGet(request, response);
  }
}
```

Now compile and deploy the above Servlet and test it using Hello.htm with the POST method as follows –

```
<html>
  <body>
    <form action = "HelloForm" method = "POST">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>
  </body>
</html>
```

Here is the actual output of the above form, Try to enter First and Last Name and then click submit button to see the result on your local machine where tomcat is running.

First Name: Last Name:

4. Explain Cookies in servlets?

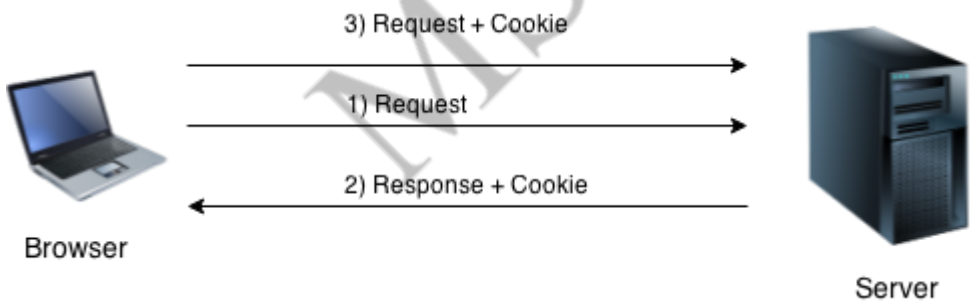
Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

PlayNext

Unmute

Current TimeÂ 0:00

/

DurationÂ 18:10

Loaded: 0.37%

Â

Fullscreen

Backward Skip 10sPlay VideoForward Skip 10s

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description

public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
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.

Other methods required for using Cookies

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

1. **public void addCookie(Cookie ck)**

2. **public Cookie[] getCookies()**

To create Cookie

Let's see the simple code to create cookie.

```
Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object

response.addCookie(ck);//adding cookie in the response
```

To delete Cookie

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

```
Cookie ck=new Cookie("user","");//deleting value of cooki e

ck.setMaxAge(0);//changing the maximum age to 0 secon ds

response.addCookie(ck);//adding cookie in the response
```

How to get Cookies?

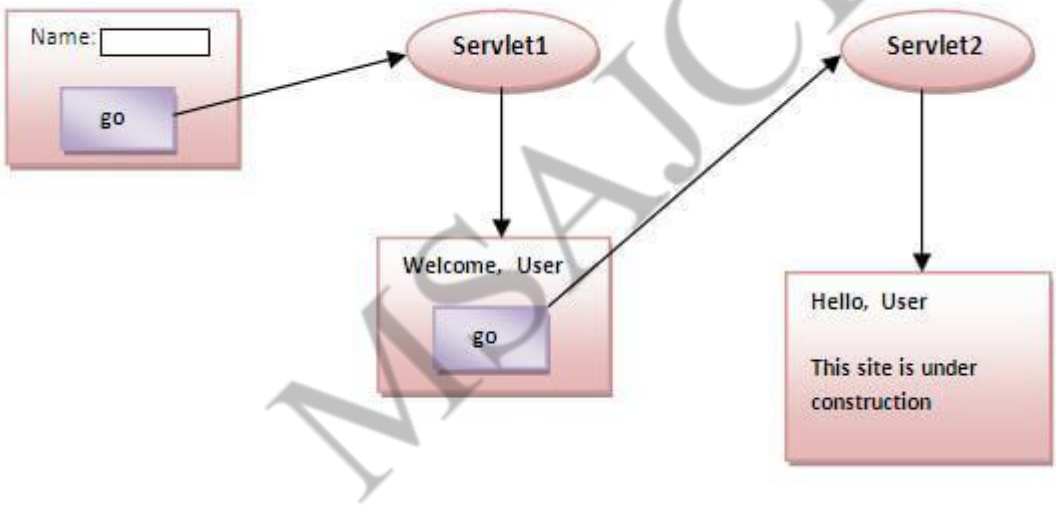
Let's see the simple code to get all the cookies.

```
Cookie ck[]=request.getCookies();

for(int i=0;i<ck.length;i++){

 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());// printing name and value of cookie
```

```
}
```

Simple example of Servlet Cookies

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.



index.html

```
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;



public class FirstServlet extends HttpServlet {


  public void doPost(HttpServletRequest request, HttpSer
vletResponse response){
    try{


    response.setContentType("text/html");
    PrintWriter out = response.getWriter();


    String n=request.getParameter("userName");
    out.print("Welcome "+n);
```

```
    Cookie ck=new Cookie("uname",n);//creating cookie ob
ject
    response.addCookie(ck);//adding cookie in the respons
e


    //creating submit button
    out.print("<form action='servlet2'>");
    out.print("<input type='submit' value='go'>");
    out.print("</form>");


    out.close();


      }catch(Exception e){System.out.println(e);}
  }
}
```

SecondServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class SecondServlet extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServ
letResponse response){
  try{

  response.setContentType("text/html");
  PrintWriter out = response.getWriter();

  Cookie ck[]=request.getCookies();
  out.print("Hello "+ck[0].getValue());

  out.close();

    }catch(Exception e){System.out.println(e);}
  }
```

}

web.xml

<web-app>

<servlet>

<servlet-name>s1</servlet-name>

<servlet-**class**>FirstServlet</servlet-**class**>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/servlet1</url-pattern>

</servlet-mapping>

<servlet>

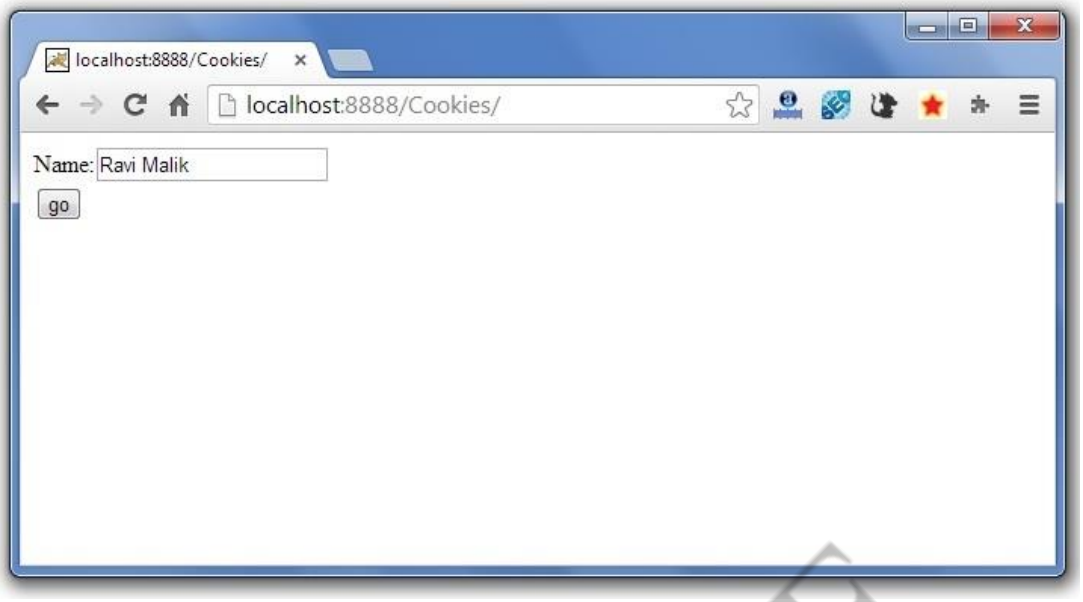
<servlet-name>s2</servlet-name>

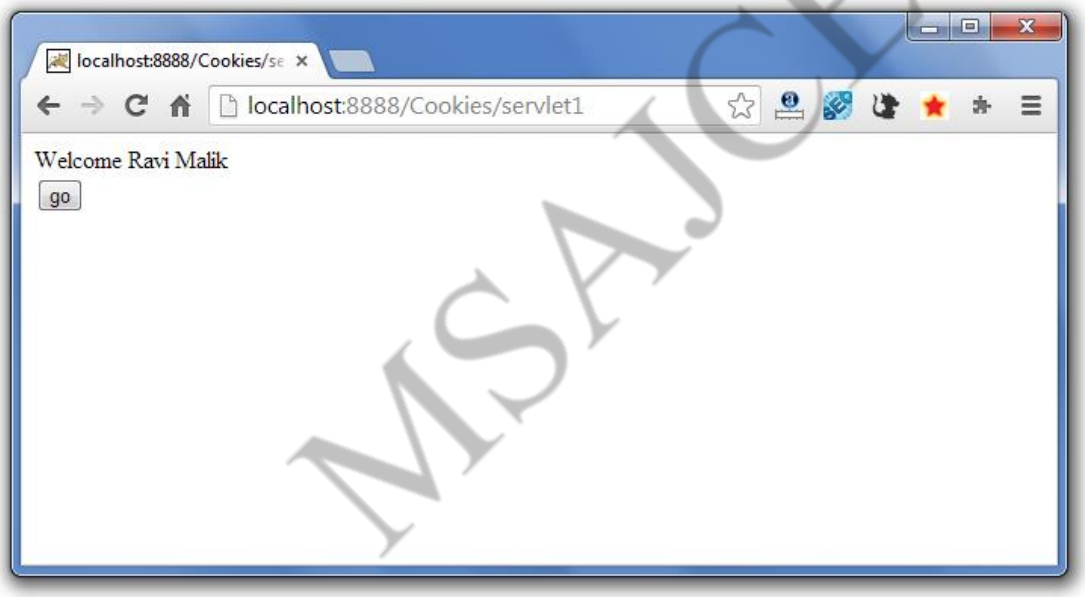
<servlet-**class**>SecondServlet</servlet-**class**>

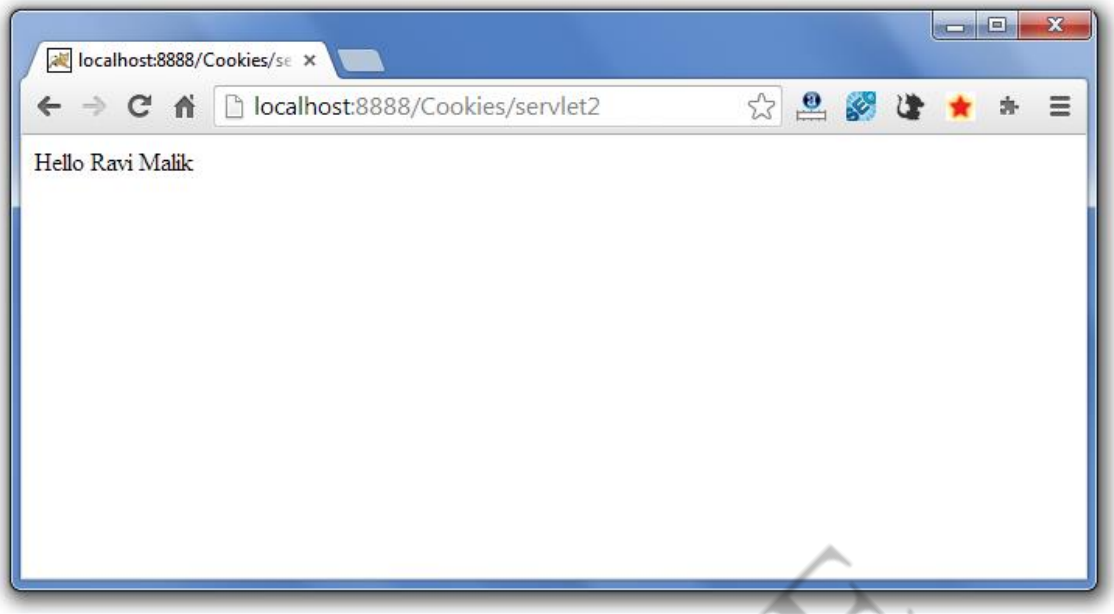
</servlet>

```
    <servlet-mapping>
    <servlet-name>s2</servlet-name>
    <url-pattern>/servlet2</url-pattern>
    </servlet-mapping>


    </web-app>
```

Output


localhost:8888/Cookies/
localhost:8888/Cookies/
Name: Ravi Malik
go


localhost:8888/Cookies/se
localhost:8888/Cookies/servlet1
Welcome Ravi Malik
go



5.Explain Database Connectivity using JDBC?

Database connectivity in servlet. There are different approaches to communicating with databases through servlet components. We need to perform the servlet to database communication due to the following main reasons:-

- To save the inputs coming from the end-user through forms to database software. Example:- Email Id registration.
- To save the result generated by servlet Component in database software. Example:- Bill generation
- To get inputs from the database software to the Servlet component. Example:- Balance inquiry, account statement generation

For servlet to database software communication, we need to place JDBC code/Hibernate code/Spring JDBC/Spring ORM/Spring Data code in Servlet Component. Since JDBC is the most basic one therefore here we will discuss servlet to database communication through JDBC code.

Different Approaches for Database Connectivity in Servlet

There are 4 approaches for Database Connectivity in Servlet.

Approach-1:- Writing logics in multiple methods.

- Create JDBC Connection in the init()
- Use the JDBC Connection in the service(-,-)/doXxx(-,-)
- Close JDBC Connection in the destroy method.

Here JDBC Connection (con) must be taken as an instance variable of the Servlet Component class.

The disadvantage of this approach:- Since JDBC Connection is an instance variable, it is not thread-safe by default. So, we should use the synchronization concept.

Advantage:- All the requests coming to Servlet Component will use a single JDBC Connection to interact with database software, due to this the performance will be improved.

Note:- This approach is **outdated** and nowadays no one is using this approach (except the maintenance side).

Approach-2:- Write all the logics in service(-,-) or doXxx(-,-) method.

- Create JDBC Connection in service(-,-)/doXxx(-,-) method.
- Use JDBC Connection in service(-,-)/doXxx(-,-) method.
- Close JDBC Connection in service(-,-)/doXxx(-,-) method.

Note:- Here JDBC Connection (con) will be a local variable to service(-,-)/doXxx(-,-) method.

Disadvantage:- The JDBC Connection (con) is a local variable so, every request which is given to the servlet will create one JDBC Connection object with database software, hence the performance will be poor.

Advantage:- Since JDBC Connection is a local variable, it is thread-safe by default. Hence there is no need to work with the synchronization concept.

Approach-3:- Use server-managed JDBC connection from the connection pool.

- Get JDBC Connection object from JDBC Connection pool being from service(-,-)/doXxx(-,-)
- Use JDBC Connection object in service(-,-)/doXxx(-,-)
- Return JDBC connection back to JDBC connection pool being from service(-,-)/doXxx(-,-)

Advantage:- Here JDBC connection is local to service(-,-)/doXxx(-,-) method. So, it becomes thread-safe.

We can get all the advantages of the JDBC connection pool. The main advantages of the JDBC connection pool are,

- Reusability of JDBC connection objects.
- With minimum JDBC connection objects, we can make max clients/requests talking with database software.
- Programmer is free from creating connection objects, managing connection objects, and destroying connection objects.

Web servers are also supplying existing created JDBC connections. Therefore programmers should not worry about how to create JDBC connections, manage them and destroy them. These tasks will be done by the web server itself. The programmer will just fetch those JDBC connections and use them in the servlet component. Moreover, one JDBC connection can be used for multiple requests, therefore performance will be good compared to approach2.

Approach4:- Use DAO class for the persistence operation.

What is DAO:- The Java class/component that separates persistence logic from other logics of the application and makes that logic as flexible logic to modify and reusable logic is called DAO (Data access object).

In this approach, we can use either approach-2 or approach-3 to get the JDBC connection. The main task in this approach,

- Write JDBC code (persistence logic) in DAO class either by using direct connection object or server-managed pooled connection object.
- Create a DAO class object in Servlet Component and use its persistence logic in Servlet Component