

# Internet Programming

**S. Sharanya**, M.E.,

Assistant Professor

Department of Computer Science and Engineering  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
Kattankulathur, Chennai.

For online purchase

[www.charulathapublications.com](http://www.charulathapublications.com)

October 2019

Price : **Rs.325/-**

ISBN No. : 978-93-89051-99-5

**CHARULATHA PUBLICATIONS**

New No.24, Thambiah Road,  
West Mambalam, Chennai - 600 033.  
Phone : 24745589, 24746546  
Email : charulathapublication@yahoo.com  
info@charulathapublications.com  
www.charulathapublications.com

# SYLLABUS

## **UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0**

Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls – CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.

## **UNIT II CLIENT SIDE PROGRAMMING**

Java Script: An introduction to JavaScript–JavaScript DOM Model–Date and Objects,- Regular Expressions- Exception Handling-Validation-Built-in objects-Event Handling-DHTML with JavaScript- JSON introduction – Syntax – Function Files – Http Request – SQL.

## **UNIT III SERVER SIDE PROGRAMMING**

Servlets: Java Servlet Architecture- Servlet Life Cycle- Form GET and POST actions- Session Handling- Understanding Cookies- Installing and Configuring Apache Tomcat Web Server- DATABASE CONNECTIVITY: JDBC perspectives, JDBC program example – JSP: Understanding Java Server Pages-JSP Standard Tag Library (JSTL)-Creating HTML forms by embedding JSP code.

## **UNIT IV PHP and XML**

An introduction to PHP: PHP- Using PHP- Variables- Program control- Built-in functions- Form Validation- Regular Expressions – File handling – Cookies – Connecting to Database. XML: Basic XML- Document Type Definition- XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).

## **UNIT V INTRODUCTION TO AJAX and WEB SERVICES**

AJAX: Ajax Client Server Architecture-XML Http Request Object-Call Back Methods; Web Services: Introduction- Java web services Basics – Creating, Publishing, Testing and Describing a Web services (WSDL)-Consuming a web service, Database Driven web service from an application –SOAP.

## TABLE OF CONTENTS

### UNIT – I

#### WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0

1.1	Web Essentials	1.1
1.1.1	Internet	1.2
1.1.2	Basic Internet Protocols	1.4
1.1.3	HTTP Request Message	1.10
1.1.4	HTTP Response Message	1.13
1.1.5	Web Clients	1.15
1.1.6	Web Servers	1.17
1.2	HTML	1.20
1.2.1	HTML Tags	1.20
1.3	Cascading Style Sheets (CSS)	1.35
	Review Questions	1.47

### UNIT – II

#### CLIENT SIDE PROGRAMMING

2.1	Scripting Languages	2.1
2.2	Introduction To Javascript	2.3
2.3	Javascript Document Object Model (DOM)	2.14
2.4	Objects In Javascript	2.23
2.4.1	JavaScript Number Object	2.24
2.4.2	Javascript String Object	2.25
2.7	Validation In Javascript	2.37
2.8	Event Handling In Javascript	2.39
2.9	DHTML With Javascript	2.41
	Review Questions	2.68

## **UNIT – III**

### **SERVER SIDE PROGRAMMING**

3.1	Servlets	3.1
3.2	Session Handling	3.16
3.3	Cookies	3.23
3.4	Configuring And Installing Apache Tomcat Server	3.29
3.5	Jdbc Connectivity	3.35
3.6	Java Server Pages (JSP)	3.44
	Review Questions	3.61

## **UNIT – IV**

### **PHP and XML 8**

4.1	Introduction To PHP	4.1
4.2	Programming With PHP	4.2
4.3	XML (Extensible Markup Language)	4.37
4.4	Newsfeeds	4.57
4.5	Atom	4.60
	Review Questions	4.62

## **UNIT – V**

### **INTRODUCTION TO AJAX and WEB SERVICES**

5.1	Introduction To AJAX	5.1
5.2	Client Server Architecture	5.3
5.3	Xmlhttprequestobject And Callback()	5.8
5.4	Web Services	5.13
5.5	Java Web Services	5.15
5.6	Developing A Web Service	5.24
5.7	Web Service Description Language (WSDL)	5.29
5.8	Consuming A Web Service	5.32

5.9	Database Driven Web Application	5.33
5.10	Simple Object Access Protocol (Soap)	5.35
5.10.1	Soap Message Structure	5.35
	Review Questions	5.42

Arunai Engineering College

---

---

# 1

## WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0

---

---

### 1.1 WEB ESSENTIALS

#### WEBSITES

A website is a set of related web pages typically served from a single web server.

A website is hosted on at least one web server, accessible via a network such as the internet or a private local area network. The pages of a website can usually be accessed from a simple Uniform Resource Locator (URL) otherwise called as web address. The URLs of the pages organize them into a hierarchy.

#### Terminologies:

**Internet:** The Internet is a collection of computers around the world connected to each other via high speed series of networks.

**World Wide Web (WWW):** The World Wide Web – or Web consists of a vast assortment of files and documents that are stored on the computers and written in some form of Hyper Text Markup Language (HTML).

**Servers:** The computers that store the files are called servers because they can serve requests from many users at the same time.

**Browsers:** A Web browser is a program that displays Web pages and other documents on the web. Examples: Internet explorer, Firefox, Google Chrome etc.

**HTML:** HTML, or Hyper Text Markup Language, is the authoring language that describes how a Web page should be displayed by a Web browser. It has two essential features:

- **Hypertext:** When a visitor clicks a link on a Web page, it leads to another web page

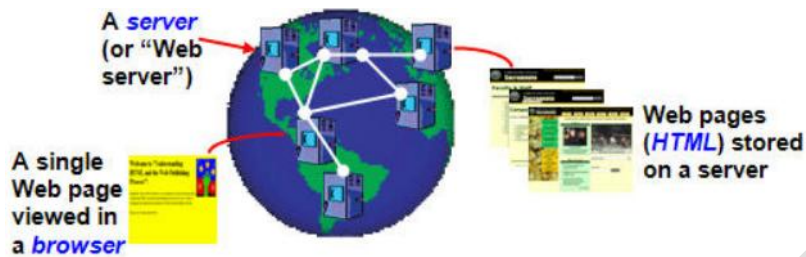


Fig 1.1 Internet and its components

### 1.1.1 INTERNET

*The Internet is a vast, electronic network connecting many millions of computers from every corner of the world. The Internet is a global network of networks.*

The Internet is a publicly-accessible network that consists of millions of smaller domestic, academic, business, and government networks. The Internet links are computer networks all over the world so that users can share resources and communicate with each other. People and organizations connect into the Internet so they can access its massive store of shared information.

The internet is a participative medium. Anybody can publish information or create new services. The internet is a **cooperative endeavor** - no organization is in charge of the internet. The following components are essential for an internet connection: Computer, Connection - Phone Line, Cable, DSL, Wireless, Modem, Network Software - TCP/IP, Application Software - Web Browser, Email, etc and Internet Service Provider (ISP).

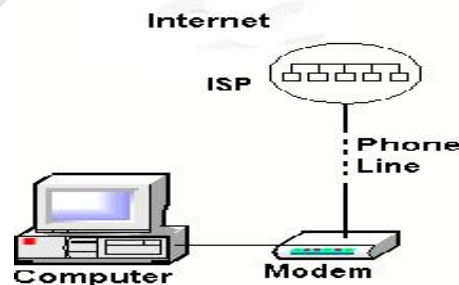


Fig 1.2 Components of internet

### Evolution of Internet

The concept of Internet was originated in 1969 and has undergone several technological & infrastructural changes:



- 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.
- 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.

### Internet Terminologies

- **Host:** A computer connected to the Internet is commonly referred to as a host.
- **Communication services:** The data is passed back and forth between host computers using packets and protocols, such as electronic mail (e-mail) for messaging, file transfer protocol (FTP) for moving files, telnet for accessing information, hypertext transfer protocol (HTTP) for serving up Web sites, custom protocols, etc. They are called communication services.
- **Internet Service Provider (ISP):** The Internet itself is decentralized-no one is completely responsible or has total control; however, the connection to the Internet is partly controlled by an Internet Service Provider (ISP). Example for ISP: Reliance, Airtel, Idea (IIN) etc.
- **Online:** When the computer is connected to the internet then it is in online.
- **Hyperlinks:** Allow a user to quickly move from one web page to another, even if the pages are on different servers in different parts of the world.
- **Protocols:** They are pre-established means of communication. **Example:** TCP/IP, SMTP.
- **TCP/IP:** TCP is the protocol that establishes a virtual connection between a destination and a source. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. Internet Protocol (IP) is responsible for packaging the little packets of information and delivering them.
- **Client/ Server model:** TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network.

- **IP address:** It is the address of the machine. It is a four byte unique number that identifies a system on the Internet.
- **Domain Name Services (DNS):** They link text to our numeric IP addresses, allowing users to use the DNS as a proxy for the IP address. The IP addresses are often provided by the ISP. Each site must register the name for a cost through a DNS hosting service. DNS host servers then are used to convert our text DNS address to its digital IP address equivalent.
  - **Universal Resource Locators:** URL's are a way of identifying information on a server. A URL gives the protocol, the domain, the directory, and even the file. A URL consists of the following parts: protocol (such as http:// or ftp://), host name (the Web server's IP address or domain name), directory (i.e. folder) and file name
- **World Wide Web:** The World Wide Web consists of all the Web sites and pages served on the Internet via HTTP. It is a hypermedia-based system for browsing Internet sites. It is named the web because it is made of many sites linked together; users can travel from one site to another by clicking on hyperlinks. Text, graphics, sound, and video can all be accessed. **Tim Berners-Lee** invented the World Wide Web in 1989 while working at CERN, the European Particle Physics Laboratory.

### 1.1.2 BASIC INTERNET PROTOCOLS

**Protocol is a set of mutually accepted and implemented rules at both ends of the communications channel for the proper exchange of information.**

#### TCP / IP

Transmission Control Protocol/Internet Protocol (TCP /IP) is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network. TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination.

TCP/IP requires little central management, and it is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network. TCP defines how applications can create channels of communication across a network. It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.

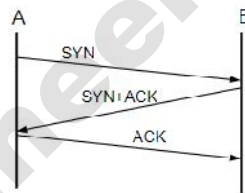
IP defines how to address and route each packet to make sure it reaches the right destination. Each gateway computer on the network checks this IP address to determine where to forward the message. A key element of IP is the IP address, which is simply a 32-bit number. IP addresses are normally written as a sequence of four decimal numbers separated by periods as in 192.0.34.166. When an application on the source computer wants to send

information to a destination, the application calls IP software on the source machine and provides it with data to be transferred along with an IP address for each of the source and destination computers.

The IP software running on the source creates a packet, which is a sequence of bits representing the data to be transferred along with the source and destination IP addresses and some other header information, such as the length of the data. If the destination computer is on the same local network as the source, then the IP software will send the packet to the destination directly via this network.

If the destination is on another network, the IP software will send the packet to a gateway, which is a device that is connected to the source computer's network as well as to at least one other network. The gateway will select a computer on one of the other networks to which it is attached and send the packet on to that computer. This process will continue, until the packet reaches the destination computer.

IP software on that computer will receive the packet and pass its data up to an application that is waiting for the data. TCP, the Transmission Control Protocol, is a higher-level protocol that extends IP to provide additional functionality, including reliable communication based on the concept of a connection.



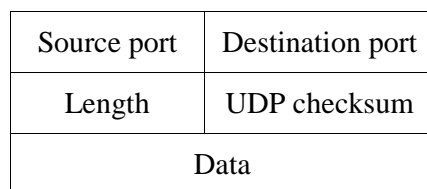
**Fig 1.5: Three way handshaking of TCP**

A connection is established between TCP software running on two machines by one of the machines sending a connection-request message via IP to the other. If the connection is accepted by B, then B returns a message to A requesting a connection in the other direction. If A responds affirmatively, then the connection is established. Notice that this means that A and B can both send messages. Once a connection has been established, TCP provides reliable data transmission by demanding an acknowledgment for each packet it sends via IP. TCP has contains port to communicate with many different applications on a machine.

### UDP, DNS and Domain Names

UDP is connectionless and unreliable protocol. It doesn't require making a connection with the host to exchange data. Since UDP is unreliable protocol, there is no mechanism for ensuring that data sent is received. UDP transmits the data in form of a datagram.

UDP provides protocol port used i.e. UDP message contains both source and destination port number, that makes it possible for UDP software at the destination to deliver the message to correct application program.



**Fig 1.3: UDP Datagram**

**Differences between TCP and UDP**

TCP	UDP
TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
As a message makes its way across the internet from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship.
TCP is suited for applications that require high reliability, and transmission time is relatively less critical	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
TCP is used by HTTP, HTTPS, FTP, SMTP, Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, VOIP.
TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
The speed for TCP is slower than UDP.	UDP is faster because there is no error-checking for packets.
There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
TCP header size is 20 bytes	UDP Header size is 8 bytes.
Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent

TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
TCP does error checking	UDP does error checking, but no recovery options.
Acknowledge segments	No Acknowledgment
Handshaking is done	No handshake (connectionless protocol)

### File Transfer Protocol (FTP)

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

- FTP creates two processes such as **Control Process and Data Transfer Process** at both ends i.e. at client as well as at server.
- FTP establishes two different connections: one is for data transfer and other is for control information.
- Control connection is made between control processes while Data Connection is made between data transfer process.
- FTP uses port 21 for the control connection and Port 20 for the data connection.

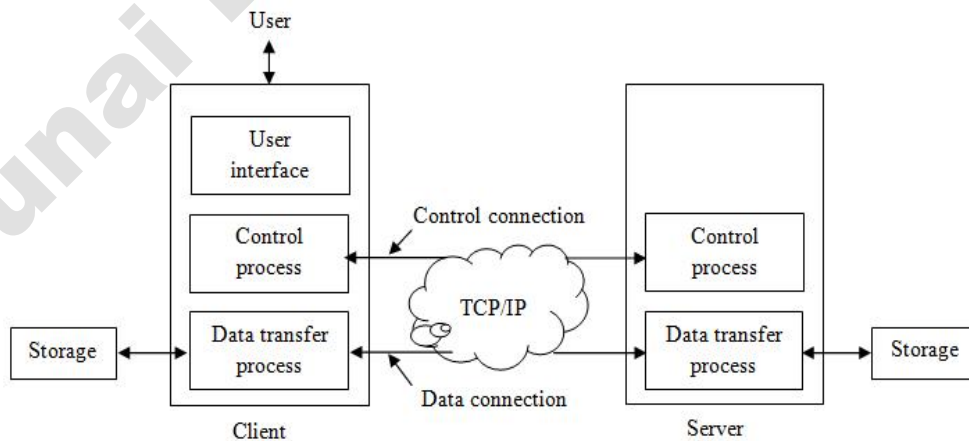


Fig 1.4 FTP

### Trivial File Transfer Protocol (TFTP)

Trivial File Transfer Protocol is also used to transfer the files but it transfers the files **without authentication**. Unlike FTP, TFTP does not separate control and data information. Since there is no authentication exists, TFTP lacks in security features therefore it is not recommended to use TFTP.

TFTP makes use of UDP for data transport. Each TFTP message is carried in separate UDP datagram. The first two bytes of a TFTP message specify the type of message. The TFTP session is initiated when a TFTP client sends a request to upload or download a file. The request is sent from an ephemeral UDP port to the UDP port 69 of a TFTP server.

### Differences between FTP and TFTP

FTP	TFTP
Authentication is done before transferring files.	No authentication is done before transferring files
The underlying protocol employed is TCP.	The underlying protocol employed is UDP.
Port 20 is used as control port and 21 for data transfers.	Port numbers: 3214, 69, 4012 are used.
Reliable data transfer is provided.	Unreliable data transfer

### Telnet

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface. The following diagram shows a person is logged in to computer A, and from there, he remotely logged into another computer B.

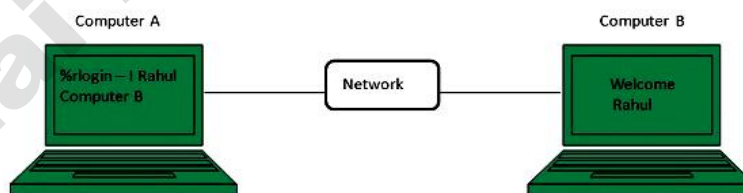


Fig 1.5 Telnet

### Hyper Text Transfer Protocol (HTTP)

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs. It is a

**stateless** protocol (i.e.) the history of the communication between server and client is not stored in any form.

### HTTP Request

HTTP request comprises of lines which contains: Request line, Header Fields and Message body. The first line i.e. the Request line specifies the request method i.e. Get or Post. The second line specifies the header which indicates the domain name of the server from where index.htm is retrieved.

### HTTP Response

Like HTTP request, HTTP response also has certain structure. HTTP response contains: Status line, Headers and Message body.

### Domain Name System (DNS)

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

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

### Domain Name System Architecture

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

#### ➤ Domain Names

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

#### ➤ Domain Name Space

The domain name space refers a hierarchy in the internet naming structure. This hierarchy has multiple levels (from 0 to 127), with a root at the top. Each domain can be partitioned into sub domains and these can be further partitioned and so on.

#### ➤ Name Server

Name server contains the **DNS database**. This database comprises of various names and their corresponding IP addresses. Since it is not possible for a single server to

maintain entire DNS database, therefore, the information is distributed among many DNS servers.

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

### Zones

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone. If the domain is not further divided into sub domains then domain and zone refers to the same thing. The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

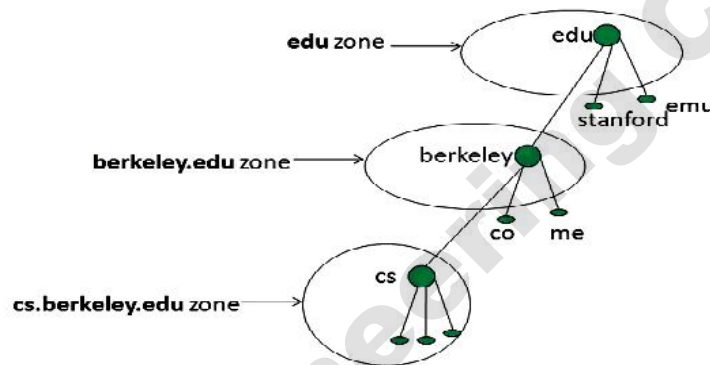


Fig 1.6 Zones in DNS

### ➤ Types of Name Servers

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

- **Root Server:** Root Server is the top level server which consists of the entire DNS tree. It does not contain the information about domains but delegates the authority to the other server.
- **Primary Server:** Primary Server stores a file about its **zone**. It has authority to create, maintain, and update the zone file.
- **Secondary Server:** Secondary Server transfers complete information about a zone from another server which may be primary or secondary server. The secondary server does not have authority to create or update a zone file.

### ➤ DNS Working

DNS translates the domain name into IP address automatically. Following are the steps in domain resolution process:



1. When we type `www.abc.com` into the browser, it asks the local DNS Server for its IP address. Here the local DNS is at ISP end.
2. When the local DNS does not find the IP address of requested domain name, it forwards the request to the root DNS server and again enquires about IP address of it.
3. The root DNS server replies with delegation that 'I do not know the IP address of `www.abc.com` but know the IP address of DNS Server'.
4. The local DNS server then asks the `com` DNS Server the same question.
5. The `com` DNS Server replies the same that it does not know the IP address of `www.abc.com` but knows the address of server that contains `www.abc.com`.
6. Then the local DNS asks the `abc.com` DNS server the same question.
7. Then `abc.com` DNS server replies with IP address of `www.abc.com`.
8. Now, the local DNS sends the IP address of `www.abc.com` to the computer that sends the request.
9. When a DNS server receives a DNS reply it cache the information in the reply in its local memory. This is called **DNS cache**.

### 1.1.3 HTTP Request Message

Whenever a URL is entered in the address box of the browser, the browser translates the URL into a request message according to the HTTP protocol; and sends the request message to the server. When the request message reaches the server, the server can take either one of these actions:

- The server interprets the request received, maps the request into a file under the server's document directory, and returns the file requested to the client.
- The server interprets the request received, maps the request into a program kept in the server, executes the program, and returns the output of the program to the client.
- The request cannot be satisfied, the server returns an error message.

The following are the parts of the HTTP request message:

**Request-Line or Start line:** This begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF (Carriage Return and Line Feed). The elements are separated by space.

**Request Method:** This indicates the method to be performed on the resource identified by the given Request-URI. The method is case-sensitive and should always be mentioned in uppercase. The following are the methods:

- **GET** - return the resource specified by the Request-URI as the body of a response message.
- **POST**- pass the body of this request message on as data to be processed by the resource specified by the Request-URI.
- **HEAD**- return the same HTTP header fields that would be returned if a GET method were used, but not return the message body that would be returned to a GET (this provides information about a resource without the communication overhead of transmitting the body of the response, which may be quite large).
- **OPTIONS**- return (in Allow header field) a list of HTTP methods that may be used to access the resource specified by the Request-URI.
- **PUT**- store the body of this message on the server and assign the specified Request-URI to the data stored so that future GET request messages containing this Request-URI will receive this data in their response messages.
- **DELETE**- respond to future HTTP request messages that contain the specified Request-URI with a response indicating that there is no resource associated with this Request-URI.
- **TRACE**- return a copy of the complete HTTP request message, including start line, header fields, and body, received by the server. Used primarily for test purposes.

### Header Fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers. Here is a list of some important Request-header fields that can be used based on the requirement: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, If-Match, If-Modified-Since, If-None-Match, If-Range, If-Unmodified-Since, Max-Forwards, Proxy-Authorization, Range, Referer, TE, User-Agent.

### Multimedia Internet Mail Extension (MIME) type

MIME are standards that are used to pass variety of types of information through internet message protocol. It has two-parts specified as the content type of the message.

**Examples:** text/html and image/jpeg. The following are the content types supported by the HTTP:

- **Application**- Data that does not fit within another content type and that is intended to be processed by application software, or that is itself an executable binary.
- **Audio**- Audio data. Subtype defines audio format.

- Image- Image data, typically static. Subtype defines image format. Requires appropriate software and hardware in order to be displayed.
- Message- Another document that represents a MIME-style message.
- Video- Animated images, possibly with synchronized sound.
- Model- Structured data, generally numeric, representing physical or behavioral models.
- Multipart- Multiple entities, each with its own header and body.
- Text- Displayable as text. That is, a human can read this document without the need for special software, although it may be easier to read with the assistance of other software.

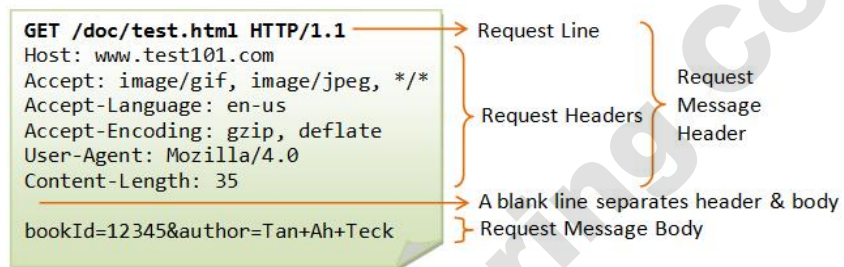


Fig 1.7 HTTP Request Message

### 1.1.4 HTTP Response Message

The following are contents of HTTP Response message:

- **Status line:** It has three fields namely HTTP version, numeric status code and text string which informs about the information represented by numeric status code.

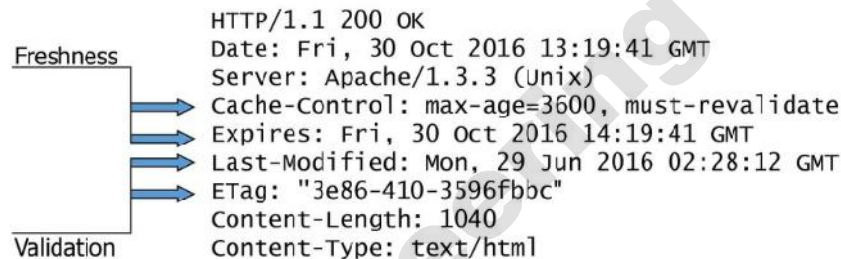
Status code	Recommended reason phrase	Usual meaning
200	OK	Request processed normally.
301	Moved Permanently	URI for the requested resource has been changed. All future requests should be made to URI contained in the location header field of the response. Most browsers will automatically send a second request to the new URI and display the second response.
307	Temporary redirect	URI for the request has been changed at least temporarily. This request should be fulfilled by making a second request to the URI contained in the location header field of the response.

401	Unauthorized	The resource is password protected, and the user has not yet supplied a valid password.
403	Forbidden	The resource is present on the server but is read protected.
404	Not found	No resource corresponding to the given request-URI was found at this server.
500	Internal server error	Server software detected an internal failure.

- Header field(s) (one or more): It contains Connection, ContentType, and Content-Length, are also valid in response messages. The Content-Type of a response can be any one of the MIME type values specified by the Accept header field of the corresponding request. Some of the common response header fields are:
  - Host- Specify authority portion of URL
  - User-Agent -A string identifying the browser or other software that is sending the request.
  - Accept MIME- types of documents that are acceptable as the body of the response, possibly with indication of preference ranking.
  - Accept Language- Specifies preferred language(s) for the response body. A server may have several translations of a document, and among these should return the one that has the highest preference rating in this header field.
  - Accept-Encoding- Specifies preferred encoding(s) for the response body.
  - Accept-Charset- Allows the client to express preferences to a server that can return a document using various character sets.
  - Connection- Indicates whether or not the client would like the TCP connection kept open after the response is sent.
  - Keep-Alive- Number of seconds TCP connection should be kept open.
  - Content-Type -The MIME type of the document contained in the message body, if one is present.
  - Content-Length- Number of bytes of data in the message body, if one is present.
  - Referer -The URI of the resource from which the browser obtained the Request-URI value for this HTTP request.
- Blank line
- Message body (optional)

**Cache control:** Web browsers automatically cache on the client machine many of the resources that they request from servers via HTTP. But information in a cache can become invalid. One approach to guaranteeing that a cached copy of a resource is valid is for the client to ask the server whether or not the client's copy is valid. This can be done with relatively little communication by sending an HTTP request for the resource using the HEAD method, which returns only the status line and header portion of the response.

**Character Sets:** A character set defines the mapping between these integers, or code points, and characters. Each US-ASCII character can be represented by a 7-bit integer, which is convenient in part because the messages transmitted by the Internet Protocol are viewed as streams of 8-bit bytes, and therefore each character can be represented by a single byte. The Unicode Standard's Basic Multilingual Plane (BMP), uses characters in every modern language, uses 16-bit character codes, and the full character code space of the Unicode Standard extends to 21-bit integers.



**Fig 1.8 HTTP Response**

### 1.1.5 Web Clients

**A web client is software that accesses a web server by sending an HTTP request message and processing the resulting HTTP response.**

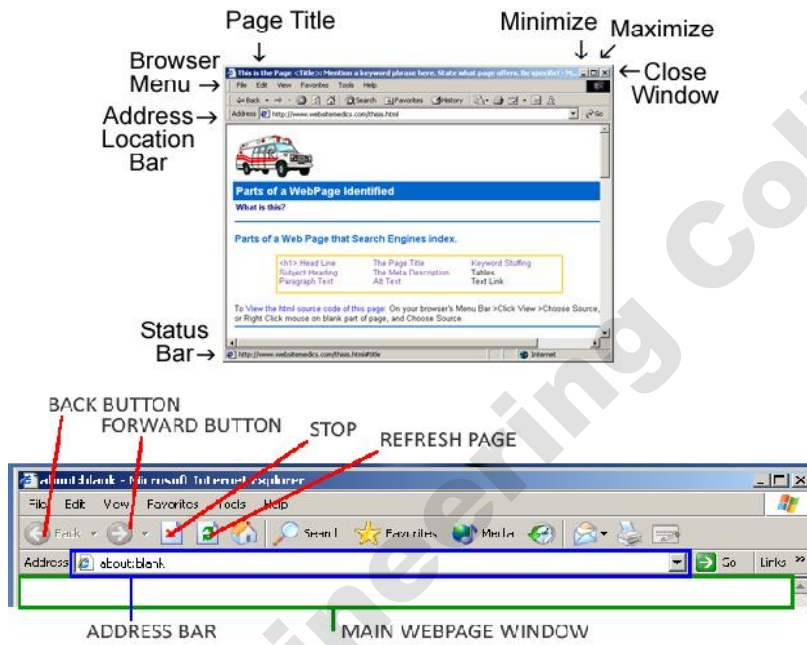
A web client is an application that communicates with a web server, using Hypertext Transfer Protocol (HTTP). Any web client that is designed to directly support user access to web servers is known as a **user agent**. Browser is the most commonly used web client. The most common interface to the World Wide Web is a browser, such as Mosaic, Netscape Navigator, or Internet Explorer.

#### **Functions of browser:**

The primary function of a web browser is to render HTML, the code used to design or mark up webpages. Each time a browser loads a web page, it processes the HTML, which may include text, links, and references to images and other items, such as cascading style sheets and JavaScript functions. The browser processes these items, then renders them in the browser window.

**Parts of browser:**

Each webpage has an address. This is indicated in the Address Bar. Most webpages will have titles. The actual webpage itself is displayed in the Main Webpage Window. You can reload a page by hitting the "Refresh" button. This will update it to the latest version. If a webpage takes a long time to load you can stop loading it by hitting the "Stop" button.



**Fig 1.9 Browser Window**

**Uniform Resource Locators (URL)**

The URI (Uniform Resource Identifier) is a string that associates a particular resource on the web. There are two types of URI:

➤ **URN (Uniform Resource Name)**

This identifies the resource using unique names. They do not signify the location of the resource. It consists of three parts: Scheme name, Namespace identifier and Namespace string

<b>Syntax:</b> Urn: name: resource name	<b>Example:</b> URN: ISBN: 5427877
---	------------------------------------

➤ **URL (Uniform Resource Locator )**

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

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

## URL Types

There are two forms of URL as listed below:

### ➤ Absolute URL:

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

**Example:** `http:// www.abc.com / xyz /index.htm`.

Here `http` is the protocol, `abc.com` is the server name and `index.htm` is the file name.

The protocol part tells the web browser how to handle the file. Other protocols also that can be used to create URL are: FTP, https, Gophe, mailto, news

### ➤ Relative URL

Relative URL is a partial address of a webpage. Unlike absolute URL, the protocol and server part are omitted from relative URL. Relative URLs are used for internal links i.e. to create links to file that are part of same website as the WebPages on which you are placing the link.

## Differences between Absolute and Relative URL

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

### 1.1.6 Web Servers

*The computer that supplies files or services to the requesting computer over the internet is called as a web server.*

The request to the web pages is sent by the browser to the server. The server will transfer the requested page to the computer over the internet.



Fig 1.10 Web server

### Working of web servers

The browser broke the URL into three parts: protocol ("http"), server name ("www.abc.com") and file name ("web-server.htm"). The browser communicated with a name server to translate the server name "www.abc.com" into an IP Address, which it uses to connect to the server machine.

Following the HTTP protocol, the browser sends a GET request to the server, asking for the file <http://www.abc.com/web-server.htm>. The server machine transfers the HTML content to the browser. The browser reads the HTML tags and formats the page onto the screen. In general, all of the machines on the Internet can be categorized as two types: **servers and clients**. The machines that provide services (like Web servers or FTP servers) to other machines are **servers**. The machines that are used to connect to those services are **clients**. It is possible and common for a machine to be both a server and a client.

A server machine may provide one or more services on the internet. For example, a server machine might have software running on it that allows it to act as a Web server, an e-mail server and an FTP server. Clients that connect to a server machine do so with a specific intent, so clients direct their requests to specific software running on the overall server machine. The web server contains **log** records that store information about server activity.

**Access log file** contains information about every HTTP requests processes by the server. **Message log file** contains a variety of debugging and other information generated by the web applications and the web server. The standard input, output and error streams are also logged.

### Differences between web sites and web servers

Web Sites	Web Servers
A website is a set of linked documents associated with a particular person, organization or topic that is held on a computer system and can be accessed as part of the world wide web.	The web server is a computer program, which delivers content, such as websites or web pages. It responds to the request for web pages.



All the web sites reside on the web server.	The web server is a computer with high configuration.
Web sites can contain text files, images, videos and audios.	Web servers are hardware or software unit.

### Features of web servers:

1. The server calls on TCP software and waits for connection requests to one or more ports.
2. When a connection request is received, the server dedicates a subtask to handling this connection.
3. The subtask establishes the TCP connection and receives an HTTP request.
4. The subtask examines the Host header field of the request to determine which virtual host should receive this request and invokes software for this host.
5. The virtual host software maps the Request-URI field of the HTTP request start line to a resource on the server.
6. If the resource is a file, the host software determines the MIME type of the file and creates an HTTP response that contains the file in the body of the response message.
7. If the resource is a program, the host software runs the program, providing it with information from the request and returning the output from the program as the body of an HTTP response message.
8. The server normally logs information about the request and response—such as the IP address of the requester and the status code of the response—in a plain-text file.
9. If the TCP connection is kept alive, the server subtask continues to monitor the connection until a certain length of time has elapsed, the client sends another request, or the client initiates a connection close.

### Configuring a server

The following are to be given higher importance while configuring a server:

- IP addresses and TCP ports that may be used to connect to this server.
- Number of subtasks that will be created when the server is initialized.
- Maximum number of threads that will be allowed to exist simultaneously
- Maximum number of TCP connection requests that will be queued if the server is already running its maximum number of threads.
- Length of time the server will wait after serving an HTTP request over a TCP connection before closing the connection if another request is not received.

## Logging

Web log file is log file automatically created and maintained by a web server. Every hit to the Web site, including each view of a HTML document, image or other object, is logged. The raw web log file format is essentially one line of text for each hit to the web site. This contains information about who was visiting the site, where they came from, and exactly what they were doing on the web site. The following are the key fields:

- Directory- Directory where log file will be written
- Pattern Information- to be written to the log
- Prefix- String that will be used to begin log file name
- Resolve Hosts -Whether IP addresses (False value) or host names (True value) should be written to the log file
- Rotatable -Whether or not date should be added to file name and file should be automatically rotated each day
- Suffix- String that will be used to end log

## 1.2 HTML

HTML stands for Hyper Text Markup Language. It allows us to organize text, graphics, audio, and video on a web page.

*HTML is a formatting language used to define the appearance and contents of a web page.*

- The word **Hypertext** refers to the text which acts as a link. The word **markup** refers to the symbols that are used to define structure of the text. The markup symbols tell the browser how to display the text and are often called tags. The word **Language** refers to the syntax that is similar to any other language.
- HTML was created by Tim Berners-Lee at CERN.

### 1.2.1 HTML Tags

*Tag is a command that tells the web browser how to display the text, audio, graphics or video on a web page.*

Tags are indicated with pair of angle brackets. They start with a less than (<) character and end with a greater than (>) character. The tag name is specified between the angle brackets. Most of the tags usually occur in pair: the start tag and the closing tag. The start tag is simply the tag name is enclosed in angle bracket whereas the closing tag is specified including

a forward slash (/).Some tags are the empty i.e. they don't have the closing tag.Tags are not case sensitive.

The starting and closing tag name must be the same. For example `<b> hello </i>` is invalid as both are different.If the angle brackets are not specified (`<>`) for a tag, the browser will treat the tag name as a simple text.The tag can also have attributes to provide additional information about the tag to the browser.

### ➤ Basic Tags

Tag	Description
<code>&lt;html&gt;&lt;/html&gt;</code>	Specifies the document as a web page
<code>&lt;head&gt;&lt;/head&gt;</code>	Specifies the descriptive information about the web documents.
<code>&lt;title&gt;&lt;/title&gt;</code>	Specifies the title of the web page.
<code>&lt;body&gt;&lt;/body&gt;</code>	Specifies the body of a web document.
<code>&lt;!DOCTYPE&gt;</code>	Defines the document type
<code>&lt;h1&gt;</code> to <code>&lt;h6&gt;</code>	Defines HTML headings
<code>&lt;p&gt;</code>	Defines a paragraph
<code>&lt;br&gt;</code>	Inserts a single line break
<code>&lt;hr&gt;</code>	Defines a thematic change in the content
<code>&lt;!--...--&gt;</code>	Defines a comment

### Basic HTML tags

<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h1&gt;My First Heading&lt;/h1&gt; &lt;p&gt;My first paragraph.&lt;/p&gt; &lt;/body&gt; &lt;/html&gt; </pre>	<p>Output:</p> <p><b>My First Heading</b></p> <p>My first paragraph.</p>
---	--

➤ **Formatting Tags**

Tag	Description
<b></b>	Specifies the text as bold.
<em></em>	It is a phrase text. It specifies the emphasized text
<strong></strong>	It is a phrase tag. It specifies an important text
<i></i>	The content of italic tag is displayed in italic.
<sub></sub>	Specifies the subscripted text
<sup></sup>	Defines the superscripted text.
<ins></ins>	Specifies the inserted text
<del></del>	Specifies the deleted text.
<mark></mark>	Specifies the marked text.
<acronym>	Not supported in HTML5. Use <abbr> instead. Defines an acronym
<abbr>	Defines an abbreviation or an acronym
<address>	Defines contact information for the author/owner of a document / article
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<bdo>	Overrides the current text direction
<big>	Not supported in HTML5. Use CSS instead. Defines big text
<blockquote>	Defines a section that is quoted from another source
<center>	Not supported in HTML5. Use CSS instead. Defines centered text
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<dfn>	Represents the defining instance of a term
<font>	Not supported in HTML5. Use CSS instead. Defines font, color, and size for text
<kbd>	Defines keyboard input

<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<q>	Defines a short quotation
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct
<samp>	Defines sample output from a computer program
<small>	Defines smaller text
<strike>	Not supported in HTML5. Use <del> instead. Defines strikethrough text
<time>	Defines a date/time
<tt>	Not supported in HTML5. Use CSS instead. Defines teletype text
<u>	Defines text that should be stylistically different from normal text
<var>	Defines a variable
<wbr>	Defines a possible line-break

### Formatting tags

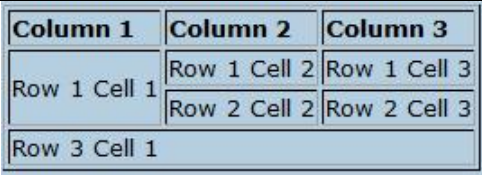
<html>	An example of <b>Bold Text</b>
<p>An example of <b>Bold Text</b></p>	An example of <i>Emphasized Text</i>
<p>An example of <em>Emphasized Text</em></p>	An example of <b>Strong Text</b>
<p>An example of <strong>Strong Text</strong></p>	An example of <i>Italic Text</i>
<p>An example of <i>Italic Text</i></p>	An example of <sup>superscripted Text</sup>
<p>An example of <sup>superscripted Text</sup></p>	An example of <sub>subscripted Text</sub>
<p>An example of <sub>subscripted Text</sub></p>	An example of <del>struckthrough Text</del>
<p>An example of <del>struckthrough Text</del></p>	An example of Computer Code
<p>An example of <code>Computer Code</code>	

<code>Text&lt;/code&gt;&lt;/p&gt;&lt;/html&gt;</code>	Text
---	------

➤ **Table Tags**

Tag	Description
<code>&lt;table&gt;&lt;/table&gt;</code>	Specifies a table.
<code>&lt;tr&gt;&lt;/tr&gt;</code>	Specifies a row in the table.
<code>&lt;th&gt;&lt;/th&gt;</code>	Specifies header cell in the table.
<code>&lt;td&gt;&lt;/td&gt;</code>	Specifies the data in an cell of the table.
<code>&lt;caption&gt;&lt;/caption&gt;</code>	Specifies the table caption.
<code>&lt;colgroup&gt;&lt;/colgroup&gt;</code>	Specifies a group of columns in a table for formatting.
<code>&lt;thead&gt;</code>	Groups the header content in a table
<code>&lt;tbody&gt;</code>	Groups the body content in a table
<code>&lt;tfoot&gt;</code>	Groups the footer content in a table
<code>&lt;col&gt;</code>	Specifies column properties for each column within a <code>&lt;colgroup&gt;</code> element

**Table tag**

<pre> &lt;html&gt;&lt;table border="1"&gt;&lt;tr&gt; &lt;td&gt;&lt;b&gt;Column 1&lt;/b&gt;&lt;/td&gt; &lt;td&gt;&lt;b&gt;Column 2&lt;/b&gt;&lt;/td&gt; &lt;td&gt;&lt;b&gt;Column 3&lt;/b&gt;&lt;/td&gt;&lt;/tr&gt; &lt;tr&gt;&lt;td rowspan="2"&gt;Row 1 Cell 1&lt;/td&gt; &lt;td&gt;Row 1 Cell 2&lt;/td&gt; &lt;td&gt;Row 1 Cell 3&lt;/td&gt;&lt;/tr&gt; &lt;tr&gt;&lt;td&gt;Row 2 Cell 2&lt;/td&gt; &lt;td&gt;Row 2 Cell 3&lt;/td&gt;&lt;/tr&gt; &lt;tr&gt;&lt;td colspan="3"&gt;Row 3 Cell 1&lt;/td&gt;&lt;/tr&gt; &lt;/table&gt;&lt;/html&gt;                 </pre>	
--	--

➤ **List tags**

Tag	Description
<ul></ul>	Specifies an unordered list.
<ol></ol>	Specifies an ordered list.
<li></li>	Specifies a list item.
<dl></dl>	Specifies a description list.
<dt></dt>	Specifies the term in a description list.
<dd></dd>	Specifies description of term in a description list.
<dir>	Not supported in HTML5. Use <ul> instead. Defines a directory list.
<menu>	Defines a list/menu of commands
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu.

**List tags**

<pre>&lt;html&gt; &lt;ol type="1" value="1"&gt; &lt;li&gt;Arabic Number&lt;/li&gt; &lt;li&gt;Arabic Number&lt;/li&gt;&lt;/ol&gt; &lt;ol type="a" value="1"&gt; &lt;li&gt;Lower Alphabet&lt;/li&gt; &lt;li&gt;Lower Alphabet&lt;/li&gt;&lt;/ol&gt; &lt;ol type="A" value="1"&gt; &lt;li&gt;Upper Alphabet&lt;/li&gt; &lt;li&gt;Upper Alphabet&lt;/li&gt;&lt;/ol&gt; &lt;ol type="i" value="1"&gt; &lt;li&gt;Lower Roman numeral&lt;/li&gt; &lt;li&gt;Lower Roman numeral&lt;/li&gt;&lt;/ol&gt; &lt;ol type="I" value="1"&gt; &lt;li&gt;Upper Roman numeral&lt;/li&gt; &lt;li&gt;Upper Roman numeral&lt;/li&gt; &lt;/ol&gt;&lt;/html&gt;</pre>	<pre>1. Arabic Number 2. Arabic Number a. Lower Alphabet b. Lower Alphabet A. Upper Alphabet B. Upper Alphabet i. Lower Roman numeral ii. Lower Roman numeral I. Upper Roman numeral II. Upper Roman numeral</pre>
--	--

➤ **Frames**

Frames help us to divide the browser’s window into multiple rectangular regions. Each region contains separate html web page and each of them work independently. A set of frames in the entire browser is known as frameset. It tells the browser how to divide browser window into frames and the web pages that each has to load.

Tag	Description
<frameset></frameset>	It is replacement of the <body> tag. It doesn’t contain the tags that are normally used in <body> element; instead it contains the <frame> element used to add each frame.
<frame></frame>	Specifies the content of different frames in a web page.

**Advantages of Frames**

- ❖ Frame provides technical sophisticated appearance to the web site.
- ❖ It facilitates to reduce downloading time and improves the usability of the website.
- ❖ Frames generally include navigation link, header or footers, which help user to find and navigate to required information.
- ❖ It separates content of website from navigation elements, which is useful for website maintenance and content modification.

**Disadvantages of Frames**

- ❖ The web developer must be track of more HTML documents linked with main frame.
- ❖ It is difficult to print the entire page, which is developed using frame.

**Frame tags**

<pre> &lt;html&gt;&lt;head&gt; &lt;title&gt;Frameset page&lt;/title&gt;&lt;/head&gt; &lt;frameset cols = "25%, *"&gt;&lt;noframes&gt; &lt;body&gt;Browser doesn't support frames. Therefore, this is the noframe version of the site.&lt;/body&gt;&lt;/noframes&gt; &lt;frame src ="/html/frame_example_left.html" /&gt;                 </pre>	
---	--



<pre>&lt;frame src ="/html/frame_example_right.html" /&gt; &lt;/frameset&gt;&lt;/html&gt;Leftframe.html&lt;html&gt; &lt;body style="background-color:green"&gt; &lt;p&gt;This is the left frame (frame_example_left.html).&lt;/p&gt;&lt;/body&gt;&lt;/html&gt; Rightframe.html &lt;html&gt;&lt;body style="background-color:yellow"&gt; &lt;p&gt;This is the right frame (frame_example_right.html).&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</pre>	
--	--

### ➤ Forms

Forms are used to input the values. These values are sent to the server for processing. Forms uses input elements such as text fields, check boxes, radio buttons, lists, submit buttons etc. to enter the data into it.

Tag	Description
<form></form>	It is used to create HTML form.
<input></input>	Specifies the input field.
<textarea></textarea>	Specifies a text area control that allows to enter multi-line text.
<label></label>	Specifies the label for an input element.
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls.

<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

### Form tags

<pre> &lt;form action="/html/tags/html_form_tag_action.cfm" method="get"&gt;&lt;fieldset&gt; &lt;legend&gt;Your Details&lt;/legend&gt; &lt;div&gt; &lt;label for="first_name"&gt;First Name:&lt;/label&gt;&lt;br&gt; &lt;input type="text" name="first_name" value="" maxlength="100" /&gt;&lt;br&gt;&lt;/div&gt; &lt;div&gt; &lt;label for="lunch"&gt;Lunch:&lt;/label&gt;&lt;br&gt; &lt;input type="radio" name="lunch" value="Meals" /&gt; meals &lt;input type="radio" name="lunch" value="Tiffin" /&gt;tiffin&lt;/div&gt; &lt;div&gt; &lt;label for="drinks"&gt;Drinks:&lt;/label&gt;&lt;br&gt; &lt;input type="checkbox" name="drinks" value="fresh juice" /&gt; fresh juice &lt;input type="checkbox" name="drinks" value="tea" /&gt; tea&lt;/div&gt; &lt;div&gt; &lt;label for="city"&gt;Preferred City:&lt;/label&gt;&lt;br&gt; &lt;select name="city"&gt; &lt;option value = "Coimbatore"&gt;coimbatore&lt;/option&gt; &lt;option value = "Chennai"&gt;chennai&lt;/option&gt; &lt;/select&gt;&lt;/div&gt; &lt;div&gt; </pre>	<div style="border: 1px solid gray; padding: 5px;"> <p><b>Your Details</b></p> <p>First Name:  <input type="text"/></p> <p>Lunch:  <input type="radio"/> meals <input type="radio"/> tiffin</p> <p>Drinks:  <input type="checkbox"/> fresh juice <input type="checkbox"/> tea</p> <p>Preferred City:  <input type="text" value="coimbatore"/> ▼</p> <p>Comments:  <input type="text"/></p> <p style="text-align: right;"><input type="button" value="Submit"/></p> </div>
--	---

<pre> &lt;label for="comments"&gt;Comments:&lt;/label&gt;&lt;br&gt; &lt;textarea      rows="3"      cols="20" name="comments"&gt;&lt;/textarea&gt;&lt;/div&gt;  &lt;div&gt; &lt;input type="submit" value="Submit" /&gt;&lt;/div&gt; &lt;/fieldset&gt;&lt;/form&gt; </pre>	
--	--

### ➤ Images

Tag	Description
<img>	Defines an image
<map>	Defines a client-side image-map
<area>	Defines an area inside an image-map
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content

### ➤ Audio /Video

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements (<video> and <audio>)
<track>	Defines text tracks for media elements (<video> and <audio>)
<video>	Defines a video or movie

### ➤ Links

Tag	Description
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

➤ **Styles and Semantics**

Tag	Description
<style>	Defines style information for a document
<div>	Defines a section in a document
<span>	Defines a section in a document
<header>	Defines a header for a document or section
<hgroup>	Defines a group of headings
<footer>	Defines a footer for a document or section
<main>	Specifies the main content of a document
<section>	Defines a section in a document
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element

➤ **Meta information**

Tag	Description
<head>	Defines information about the document
<meta>	Defines metadata about an HTML document
<base>	Specifies the base URL/target for all relative URLs in a document
<basefont>	Not supported in HTML5. Use CSS instead. Specifies a default color, size, and font for all text in a document

➤ **Programming**

Tag	Description
<script>	Defines a client-side script

<noscript>	Defines an alternate content for users that do not support client-side scripts
<applet>	Not supported in HTML5. Use <object> instead. Defines an embedded applet
<embed>	Defines a container for an external (non-HTML) application
<object>	Defines an embedded object
<param>	Defines a parameter for an object

### Drag and Drop

Drag and Drop is a very interactive and user-friendly concept which makes it easier to move an object to a different location by grabbing it. This allows the user to click and hold the mouse button over an element, drag it to another location, and release the mouse button to drop the element there. In HTML 5 Drag and Drop are much easier to code and any element in it is draggable.

Tag	Description
ondragstart	Calls a function, drag(event), that specifies what data to be dragged
ondragenter	To determine whether or not the drop target is to accept the drop. If the drop is to be accepted, then this event has to be canceled
ondragleave	Occurs when the mouse leaves an element before a valid drop target while the drag is occurring
ondragover	Specifies where the dragged data can be dropped
Ondrop	Specifies where the drop was occurred at the end of the drag operation
ondragend	Occurs when the user has finished dragging an element

### XHTML

XHTML stands for **Extensible Hyper Text Markup Language**. It is the next step in the evolution of the internet. The XHTML 1.0 is the first document type in the XHTML family.

XHTML is almost identical to HTML 4.01 with only few differences. XHTML was developed by W3C to help web developers make the transition from HTML to XML. By migrating to XHTML today, web developers can enter the XML world with all of its benefits.

XHTML has stricter syntax rules in comparison to HTML. XHTML gives you a more consistent, well-structured format so that the webpages can be easily parsed and processed by present and future web browsers.

### Advantages of XHTML

- ❖ XHTML documents are XML conforming as they are readily viewed, edited, and validated with standard XML tools.
- ❖ XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.
- ❖ XHTML documents can utilize applications such as scripts and applets that rely upon either the HTML Document Object Model or the XML Document Object Model.

### Migration from HTML to XHTML

#### ➤ DOCTYPE Declaration

All XHTML documents must have a DOCTYPE declaration at the start.

#### **Example:**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

#### ➤ Case sensitivity

XHTML is case sensitive markup language. All the XHTML tags and attributes need to be written in lower case only.

- `<!-- This is invalid in XHTML -->`
- `<A Href="/xhtml/xhtml_abc.html">XHTML </A>`
- `<!-- Correct XHTML way of writing this is as follows -->`
- `<ahref="/xhtml/xhtml_abc.html">XHTML</a>`: In the example, Href and anchor tag A are having characters which are not in lower case, so it is incorrect.

#### ➤ Closing tags

Each and every XHTML tag should have an equivalent closing tag, even empty elements should also have closing tags.

- `<!-- This is valid in XHTML -->`
- `<p>This paragraph is not written according to XHTML syntax.</p>`

- <!-- This is also valid now -->
- <imgsrc="/images/xhtml.gif" />

#### ➤ Attribute quotes

All the values of XHTML attributes must be quoted. Otherwise, the XHTML document is assumed as an invalid document.

<!-- Correct XHTML way of writing this is as follows -->

```
<imgsrc="/images/xhtml.gif" width="250" height="50" />
```

#### ➤ Attribute minimization

XHTML does not allow attribute minimization. It means attribute and its values must be explicitly stated.

<!-- Correct XHTML way of writing this is as follows -->

```
<option selected="selected">
```

HTML style	XHTML style	HTML style	XHTML style
compact	compact="compact"	ismap	ismap="ismap"
Checked	checked="checked"	nohref	nohref="nohref"
Declare	declare="declare"	noshade	noshade="noshade"
readonly	readonly="readonly"	nowrap	nowrap="nowrap"
disabled	disabled="disabled"	multiple	multiple="multiple"
Selected	selected="selected"	noresize	noresize="noresize"
Defer	defer="defer"		

#### ➤ The id attribute

The id attribute replaces the name attribute. Instead of using name="name", XHTML prefers to use id="id".

<!-- Correct XHTML way of writing this is as follows -->

```
<imgsrc="/images/xhtml.gif" id="xhtml_logo" />
```

#### ➤ The language attribute

The language attribute of the script tag is deprecated.

<!-- Correct XHTML way of writing this is as follows -->

```
<script type="text/JavaScript">
```

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

➤ **Nested tags**

All the XHTML tags must be nested properly otherwise the document will be assumed as an incorrect XHTML document.

<!-- Correct XHTML way of writing this is as follows -->

```
<b><i>This text is bold and italic</i></b>
```

➤ **Element prohibitions**

The following elements are not allowed to have any other element inside them. This prohibition applies to all depths of nesting, i.e. it includes all the descendant elements.

Element	Prohibition
<a>	Must not contain other <a> elements.
<pre>	Must not contain the <img>, <object>, <big>, <small>, <sub>, or <sup> elements.
<button>	Must not contain the <input>, <select>, <textarea>, <label>, <button>, <form>, <fieldset>, <iframe> or <isindex> elements.
<label>	Must not contain other <label> elements.
<form>	Must not contain other <form> elements.

**Differences between HTML and XHTML**

HTML	XHTML
HTML or Hyper Text Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.	HTML (Extensible HyperText Markup Language) is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written.
It has document file format.	It is a mark- up language.



It is extended from SGML.	It is extended from XML and HTML.
It has flexible framework requiring lenient HTML specific parser.	It is restrictive subset of XML and needs to be parsed with standard XML parsers.

### 1.3 CASCADING STYLE SHEETS (CSS)

CSS helps to define the presentation of HTML elements as a separate file known as CSS file having .css extension. CSS helps to change formatting of any HTML element by just making changes at one place. All changes made would be reflected automatically to all of the web pages of the website in which that element appeared.

#### CSS Rules

CSS Rules are the styles that we have to create in order to create style sheets. These rules define appearance of associated HTML element.

Selector {property: value;}

- ❖ Selector is HTML element to which CSS rule is applied.
- ❖ Property specifies the attribute that the user wants to change corresponding to the selector.
- ❖ Property can take specified value. Property and Value are separated by a colon (:). Each declaration is separated by semi colon (;).

**Examples:** i) P { color : red;} ii) h1 (color : green; font-style : italic } iii) body { color : cyan; font-family : Arial; font- style : 16pt}

#### Embedding CSS into HTML

Following are the four ways to add CSS to HTML documents:

##### ➤ Inline Style Sheets

Inline Style Sheets are included with HTML element i.e. they are placed inline with the element. To add inline CSS, we have to declare style attribute which can contain any CSS property.

<b>Syntax:</b> <Tagname STYLE = “ Declaration1 ; Declaration2 “> .... </Tagname>	<b>Example:</b> <p style="color: blue; text-align: left; font-size: 15pt">
---	---

##### ➤ Embedded Style Sheets

Embedded Style Sheets are used to apply same appearance to all occurrence of a specific element. These are defined in element by using the <style> element. The

<style> element must include type attribute. The value of type attribute specifies what type of syntax it includes when rendered by the browser.

|  |   |
|--|---|
| <p><b>Syntax:</b></p> <pre>&lt;head&gt;&lt;title&gt; .... &lt;/title&gt; &lt;style type =”text/css”&gt; .....CSS Rules/Styles....&lt;/head&gt;</pre> | <p><b>Example:</b></p> <pre>&lt;style type="text/css"&gt; p {color:green; text-align: left; font-size: 10pt} h1 { color: red; font-weight: bold} &lt;/style&gt;</pre> |
|--|---|

➤ **External Style Sheets**

External Style Sheets are the separate .css files that contain the CSS rules. These files can be linked to any HTML documents using <link> tag with rel attribute.

**Syntax:**

```
<head><link rel= “stylesheet” type=”text/css” href= “url of css file”> </head>
```

In order to create external css and link it to HTML document, follow the following steps:

- ❖ Create a CSS file and define all CSS rules for several HTML elements. Let’s name this file as external.css.

```
p { Color: orange; text-align: left; font-size: 10pt;}
h1 { Color: orange; font-weight: bold;}
```

- ❖ Now create HTML document and name it as externaldemo.html.

```
<html><head> <title> External Style Sheets Demo </title>
<link rel="stylesheet" type="text/css" href="external.css"></head>
<body> <h1> External Style Sheets</h1>
<p>External Style Sheets are the separate .css files that contain the CSS
rules.</p></body> </html>
```

➤ **Imported Style Sheets**

Imported Style Sheets allow us to import style rules from other style sheets. To import CSS rules we have to use @import before all the rules in a style sheet.

<p><b>Syntax:</b></p> <pre>&lt;head&gt;&lt;title&gt; Title Information &lt;/title&gt; &lt;style type=”text/css”&gt;</pre>	<p><b>Example:</b></p> <pre>&lt;html&gt;&lt;head&gt; &lt;title&gt; External Style Sheets Demo</pre>
---	---

<pre>@import URL (cssfilepath)     ... CSS rules... &lt;/style&gt; &lt;/head&gt;&lt;/style&gt;</pre>	<pre>&lt;/title&gt;&lt;style&gt; @import url(external.css); &lt;/style&gt; &lt;/head&gt; &lt;body&gt; &lt;h1&gt; External Style Sheets&lt;/h1&gt; &lt;p&gt;External Style Sheets.&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;</pre>
--	---

### CSS imported

```
<!DOCTYPE html><html><head><style>
p.ex1 { font: 15px arial, sans-serif;}
p.ex2 {font:italic bold 12px/30px Georgia, serif;}</style></head>
<body>
<p class="ex1">This is a paragraph. This is a paragraph. This is a paragraph. This is
a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a
paragraph.</p>
<p class="ex2">This is a paragraph. This is a paragraph. This is a paragraph. This is
a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This is a
paragraph.</p></body></html>
```

### Output:

```
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph. This
is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.

This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
This is a paragraph. This is a paragraph. This is a paragraph. This is a paragraph.
```

### CSS - Pseudo Classes

CSS pseudo-classes are used to add special effects to some selectors.

**Syntax:** selector:pseudo-class {property: value}

CSS classes can also be used with pseudo-classes:

**Syntax:** selector.class:pseudo-class {property: value}

There are following most commonly used pseudo-classes:

Value	Description
:link	Use this class to add special style to an unvisited link.
:visited	Use this class to add special style to a visited link.
:hover	Use this class to add special style to an element when you mouse over it.
:active	Use this class to add special style to an active element.
focus	Use this class to add special style to an element while the element has focus.
:first-child	Use this class to add special style to an element that is the first child of some other element.
:lang	Use this class to specify a language to use in a specified element.

While defining pseudo-classes in a `<style>...</style>` block, following points should be noted:

- `a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective.
- `a:active` MUST come after `a:hover` in the CSS definition in order to be effective.
- Pseudo-class names are not case-sensitive.
- Pseudo-class are different from CSS classes but they can be combined.

#### The `:link` pseudo-class

```
<style type="text/css">
a:link {color:#000000}
</style>
<a href="/html/index.htm">Black Link</a>
```

#### The `:visited` pseudo-class

```
<style type="text/css">
a:visited {color: #006600}
</style>
<a href="/html/index.htm">Click this link</a>
```

When the link is clicked, it will change its color to green.

**The :hover pseudo-class**

```
<style type="text/css">
a:hover {color: #FFCC00}
</style>
<a href="/html/index.htm">Bring Mouse Here</a>
```

When the mouse is moved over this link and it changes its color to yellow.

**The :active pseudo-class**

```
<style type="text/css">
a:active {color: #FF00CC}
</style>
<a href="/html/index.htm">Click This Link</a>
```

When this link is clicked the color will be changed to pink.

**Inheritance in CSS**

In CSS, inheritance controls what happens when no value is specified for a property on an element. The inherit keyword allows authors to explicitly specify inheritance. It works on both inherited and non-inherited properties.

**Inherited property:** When no value for an inherited property has been specified on an element, the element gets the computed value of that property on its parent element. Only the root element of the document gets the initial value given in the property's summary.

**Example:** color

**Non inherited property:** When no value for a non-inherited property has been specified on an element, the element gets the initial value of that property

**Example:** border

The following are the property values:

- **Inherit:** Sets the property value applied to a selected element to be the same as that of its parent element.
- **Initial:** Sets the property value applied to a selected element to be the same as the value set for that element in the browser's default style sheet. If no value is set by the browser's default style sheet and the property is naturally inherited, then the property value is set to inherit instead.

- **Unset:** Resets the property to its natural value, which means that if the property is naturally inherited it acts like inherit, otherwise it acts like initial.
- **Revert:** Reverts the property to the value it would have had if the current origin had not applied any styles to it. In other words, the property's value is set to the user stylesheet's value for the property (if one is set), otherwise, the property's value is taken from the user agent's default stylesheet.

<pre> &lt;ul&gt;&lt;li&gt;Default &lt;a href="#"&gt;link&lt;/a&gt; color&lt;/li&gt; &lt;li class="my-class-1"&gt;Inherit the &lt;a href="#"&gt;link&lt;/a&gt; color&lt;/li&gt; &lt;li class="my-class-2"&gt;Reset the &lt;a href="#"&gt;link&lt;/a&gt; color&lt;/li&gt; &lt;li class="my-class-3"&gt;Unset the &lt;a href="#"&gt;link&lt;/a&gt; color&lt;/li&gt;&lt;/ul&gt; body { color: green;} .my-class-1 a { color: inherit;} .my-class-2 a { color: initial;} .my-class-3 a { color: unset;    }         </pre>	<p>Result:</p> <ul style="list-style-type: none"> <li>• Default <a href="#">link</a> color</li> <li>• Inherit the <a href="#">link</a> color</li> <li>• Reset the <a href="#">link</a> color</li> <li>• Unset the <a href="#">link</a> color</li> </ul>
---	---

- Set the color of the <body> to green.
- As the color property is naturally inherited, all child elements of body will have the same green color. It's worth noting that browsers set the color of links to blue by default instead of allowing the natural inheritance of the color property, so the first link in our list is blue.
- The second rule sets links within an element with the class my-class-1 to inherit its color from its parent. In this case, it means that the link inherits its color from its <li> parent, which, by default inherits its color from its own <ul> parent, which ultimately inherits its color from the <body> element, which had its color set to green by the first rule.
- The third rule selects any links within an element with the class my-class-2 and sets their color to initial. Usually, the initial value set by browsers for the text color is black, so this link is set to black.
- The last rule selects all links within an element with the class my-class-3 and sets their color to unset — we unset the value. Because the color property is a naturally inherited property it acts exactly like setting the value to inherit. As a consequence, this link is set to the same color as the body — green.

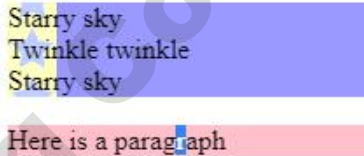
### CSS Background:

Defines how the background image will behave when scrolling the page. The background image will scroll with the page. It will also position and resize itself according to

the element it's applied to. The background property is specified as one or more background layers, separated by commas. Each layer may include zero or one occurrences of any of the following values: <attachment>, <bg-image>, <position>, <bg-size> and <repeat-style>


The <bg-size> value may only be included immediately after <position>, separated with the '/' character, like this: "center/80%".

The <box> value may be included zero, one, or two times. If included once, it sets both background-origin and background-clip. If it is included twice, the first occurrence sets background-origin, and the second sets background-clip. The <background-color> value may only be included in the last layer specified.

<pre>&lt;p class="topbanner"&gt; Starry sky&lt;br/&gt; Twinkle twinkle&lt;br/&gt; Starry sky&lt;/p&gt;  &lt;p class="warning"&gt;Here is a paragraph&lt;p&gt;  .warning { background: pink; }  .topbanner { background: url("https://mdn.mozillademos.org/files/11983/starsolid.gif ") #99f repeat-y fixed; }</pre>	
---	---

### Border:

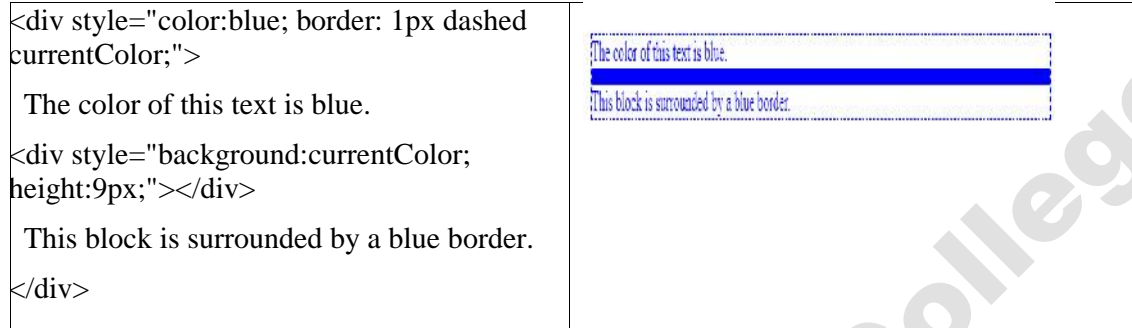
The border shorthand CSS property sets an element's border. It sets the values of border-width, border-style, and border-color.

<pre>&lt;div&gt;I have a border, an outline, AND a box shadow! Amazing, isn't it?&lt;/div&gt;  div { border: 0.5rem outset pink; outline: 0.5rem solid khaki; box-shadow: 0 0 2rem skyblue; border-radius: 12px; font: bold 1rem sans-serif; margin: 2rem; padding: 1rem; outline-offset: 0.5rem; }</pre>	
---	---

### CSS Colors:

CSS Color is a CSS module that deals with colors, color types, color blending, opacity, and how you can apply these colors and effects to HTML content. Not all CSS properties that take a <color> as a value are part of this module, but they do depend upon it. The <color> CSS data type represents a color in the RGB color space. A <color> may also include an alpha-channel transparency value, indicating how the color should composite with

its background. A <color> can be defined in any of the following ways: Using a keyword, Using the RGB cubic-coordinate system and Using the HSL cylindrical-coordinate system



### CSS Shadows:

CSS can add shadow to text and to elements.

**CSS Text Shadow:** The CSS text-shadow property applies shadow to text.

```
h1 { text-shadow: 2px 2px;}
```

To add more than one shadow to the text, add a comma-separated list of shadows.

```
h1 { text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;}
```

**Box Shadow:** The CSS box-shadow property applies shadow to elements.

```
div { box-shadow: 10px 10px grey;}
```

### CSS Text

**Text Color:** The color property is used to set the color of the text. The color is specified by: a color name - like "red", a HEX value - like "#ff0000" or an RGB value - like "rgb(255,0,0)"

- direction-Specifies the text direction/writing direction
- letter-spacing- Increases or decreases the space between characters in a text
- line-height-Sets the line height
- text-align-Specifies the horizontal alignment of text
- text-decoration-Specifies the decoration added to text
- text-indent-Specifies the indentation of the first line in a text-block
- text-shadow- Specifies the shadow effect added to text



- text-transform -Controls the capitalization of text
- text-overflow- Specifies how overflowed content that is not displayed should be signaled to the user
- vertical-align- Sets the vertical alignment of an element
- white-space- Specifies how white-space inside an element is handled
- word-spacing- Increases or decreases the space between words in a text

```
html { font-size: 10px;}
h1 { font-size: 2.6rem; text-transform: capitalize;}
h1 + p { font-weight: bold;}
p { font-size: 1.4rem; color: red; font-family: Helvetica, Arial, sans-serif;}
```

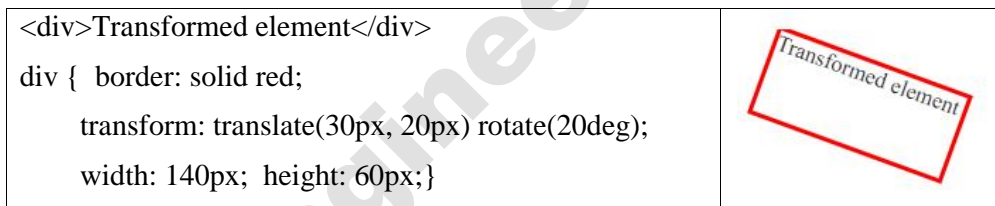
### CSS Transformation

The transform CSS property allows to rotate, scale, skew, or translate an element. It modifies the coordinate space of the CSS visual formatting model. Only transformable elements can be transformed. That is, all elements whose layout is governed by the CSS box model except for: non-replaced inline boxes, table-column boxes, and table-column-group boxes.

#### Properties:

- none -Defines that there should be no transformation
- matrix(n,n,n,n,n,n)-Defines a 2D transformation, using a matrix of six values
- matrix3d (n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)-Defines a 3D transformation, using a 4x4 matrix of 16 values
- translate(x,y)-Defines a 2D translation
- translate3d(x,y,z)-Defines a 3D translation
- translateX(x)- Defines a translation, using only the value for the X-axis
- translateY(y)- Defines a translation, using only the value for the Y-axis
- translateZ(z)- Defines a 3D translation, using only the value for the Z-axis
- scale(x,y)- Defines a 2D scale transformation
- scale3d(x,y,z)-Defines a 3D scale transformation
- scaleX(x)-Defines a scale transformation by giving a value for the X-axis

- scaleY(y)-Defines a scale transformation by giving a value for the Y-axis
- scaleZ(z)-Defines a 3D scale transformation by giving a value for the Z-axis
- rotate(angle)-Defines a 2D rotation, the angle is specified in the parameter
- rotate3d(x,y,z,angle)-Defines a 3D rotation
- rotateX(angle)-Defines a 3D rotation along the X-axis
- rotateY(angle)-Defines a 3D rotation along the Y-axis
- rotateZ(angle)-Defines a 3D rotation along the Z-axis
- skew(x-angle,y-angle)-Defines a 2D skew transformation along the X- and the Y-axis
- skewX(angle) Defines a 2D skew transformation along the X-axis
- skewY(angle) Defines a 2D skew transformation along the Y-axis
- perspective(n) Defines a perspective view for a 3D transformed element
- initial Sets this property to its default value.
- inherit Inherits this property from its parent element.



### CSS Transitions:

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, the transition cause the changes in a property to take place over a period of time. To create a transition effect, you must specify two things:

1. the CSS property you want to add an effect to
2. the duration of the effect. If the duration part is not specified, the transition will have no effect, because the default value is 0.

**transition-property:** Specifies the name or names of the CSS properties to which transitions should be applied. Only properties listed here are animated during transitions; changes to all other properties occur instantaneously as usual.

**transition-duration:** Specifies the duration over which transitions should occur. You can specify a single duration that applies to all properties during the transition, or multiple values to allow each property to transition over a different period of time.

**Example:**transition-duration: 1s

**transition-timing-function:** Specifies a function to define how intermediate values for properties are computed. Timing functions determine how intermediate values of the transition are calculated. Most timing functions can be specified by providing the graph of the corresponding function, as defined by four points defining a cubic bezier. Some of the functions are:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets to define our own values in a cubic-bezier function

**transition-delay:** Defines how long to wait between the time a property is changed and the transition actually begins.

### CSS Animations

CSS animations make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

#### Advantages of animations over transitions:

- They're easy to use for simple animations; you can create them without even having to know JavaScript.
- The animations run well, even under moderate system load.
- Letting the browser control the animation sequence lets the browser optimize performance and efficiency.

Keyframes are used to create animations. The @keyframes CSS at-rule controls the intermediate steps in a CSS animation sequence by defining styles for keyframes along the animation sequence. This gives more control over the intermediate steps of the animation sequence than transitions. The sub-properties of the animation property are:

- animation-name- Specifies the name of the @keyframes at-rule describing the animation's keyframes.

- animation-duration- Configures the length of time that an animation should take to complete one cycle.
- animation-timing-function- Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.
- animation-delay- Configures the delay between the time the element is loaded and the beginning of the animation sequence.
- animation-iteration-count- Configures the number of times the animation should repeat.
- animation-direction- Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.
- animation-fill-mode- Configures what values are applied by the animation before and after it is executing.
- animation-play-state- Lets to pause and resume the animation sequence.

```
p { animation-duration: 3s; animation-name: slidein;}
@keyframes slidein { from { margin-left: 100%; width: 300%; }
to { margin-left: 0%; width: 100%; }}
```

In this example the style for the <p> element specifies that the animation should take 3 seconds to execute from start to finish, using the animation-duration property, and that the name of the @keyframes at-rule defining the keyframes for the animation sequence is named “slidein”.

## REVIEW QUESTIONS

### PART-A

**1. Define website.**

A website is a set of related web pages typically served from a single web server.

**2. Define Internet.**

The Internet is a collection of computers around the world connected to each other via high speed series of networks.

**3. What is meant by World Wide Web (WWW)?**

The World Wide Web – or Web consists of a vast assortment of files and documents that are stored on the computers and written in some form of Hyper Text Markup Language (HTML).

**4. What are Servers?**

The computers that store the files are called servers because they can serve requests from many users at the same time.

**5. Define Browsers.**

A Web browser is a program that displays Web pages and other documents on the web. Examples: Internet explorer, Firefox, Google Chrome etc.

**6. What is HTML?**

Hyper Text Markup Language, is the authoring language that describes how a Web page should be displayed by a Web browser.

**7. Give the evolution of internet.**

- 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.
- 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.

**8. Define host.**

A computer connected to the Internet is commonly referred to as a host.

**9. What are Communication services?**

The data is passed back and forth between host computers using packets and protocols, such as electronic mail (e-mail) for messaging, file transfer protocol (FTP) for moving files, telnet for accessing information, hypertext transfer protocol (HTTP) for serving up Web sites, custom protocols, etc. They are called communication services.

**10. What do you mean by Internet Service Provider (ISP)?**

The Internet itself is decentralized-no one is completely responsible or has total control; however, the connection to the Internet is partly controlled by an Internet Service Provider (ISP). Example for ISP: Reliance, Airtel, Idea (IIN) etc.

**11. What is the use of Hyperlinks?**

Allow a user to quickly move from one web page to another, even if the pages are on different servers in different parts of the world.

**12. Give the use of TCP/IP?**

TCP is the protocol that establishes a virtual connection between a destination and a source. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. Internet Protocol (IP) is responsible for packaging the little packets of information and delivering them.

**13. What is Client/ Server model?**

TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network.

**14. Give the significance of IP address.**

It is the address of the machine. It is a four byte unique number that identifies a system on the Internet.

**15. What is Domain Name Services (DNS)?**

They link text to our numeric IP addresses, allowing users to use the DNS as a proxy for the IP address. The IP addresses are often provided by the ISP. Each site must register the name for a cost through a DNS hosting service. DNS host servers then are used to convert our text DNS address to its digital IP address equivalent.

**16. Define Universal Resource Locators.**

URL's are a way of identifying information on a server. A URL gives the protocol, the domain, the directory, and even the file. A URL consists of the following parts:protocol (such as http:// or ftp://), host name (the Web server's IP address or domain name), directory (i.e. folder) and file name

**17. Define protocol.**

Protocol is a set of mutually accepted and implemented rules at both ends of the communications channel for the proper exchange of information.

**18. Differentiate between TCP and UDP.**

TCP	UDP
TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
As a message makes its way across the internet from one computer to another. This is connection based.	UDP is also a protocol used in message transport or transfer. This is not connection based which means that one program can send a load of packets to another and that would be the end of the relationship.
TCP is suited for applications that require high reliability, and transmission time is relatively less critical	UDP is suitable for applications that need fast, efficient transmission, such as games. UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients.
TCP is used by HTTP, HTTPS, FTP, SMTP, Telnet	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, VOIP.
TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.
The speed for TCP is slower than UDP.	UDP is faster because there is no error-checking for packets.
There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
TCP header size is 20 bytes	UDP Header size is 8 bytes.
Data is read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries	Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent

TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP is lightweight. There is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP.
TCP does Flow Control. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.	UDP does not have an option for flow control
TCP does error checking	UDP does error checking, but no recovery options.
Acknowledge segments	No Acknowledgment
Handshaking is done	No handshake (connectionless protocol)

### 19. Brief the working of FTP.

FTP is used to copy files from one host to another. FTP offers the mechanism for the same in following manner:

- FTP creates two processes such as Control Process and Data Transfer Process at both ends i.e. at client as well as at server.
- FTP establishes two different connections: one is for data transfer and other is for control information.
- Control connection is made between control processes while Data Connection is made between data transfer process.
- FTP uses port 21 for the control connection and Port 20 for the data connection.

### 20. Distinguish between FTP and TFTP.

FTP	TFTP
Authentication is done before transferring files.	No authentication is done before transferring files
The underlying protocol employed is TCP.	The underlying protocol employed is UDP.
Port 20 is used as control port and 21 for data transfers.	Port numbers: 3214, 69, 4012 are used.
Reliable data transfer is provided.	Unreliable data transfer



**21. Define telnet.**

Telnet is a protocol used to log in to remote computer on the internet. There are a number of Telnet clients having user friendly user interface.

**22. What is HTTP?**

HTTP is a communication protocol. It defines mechanism for communication between browser and the web server. It is also called request and response protocol because the communication between browser and server takes place in request and response pairs. It is a stateless protocol (i.e.) the history of the communication between server and client is not stored in any form.

**23. What is DNS?**

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

**24. What are zones in DNS?**

Zone is collection of nodes (sub domains) under the main domain. The server maintains a database called zone file for every zone. If the domain is not further divided into sub domains then domain and zone refers to the same thing. The information about the nodes in the sub domain is stored in the servers at the lower levels however; the original server keeps reference to these lower levels of servers.

**25. Define web client.**

A web client is software that accesses a web server by sending an HTTP request message and processing the resulting HTTP response.

**26. What is URN?**

This identifies the resource using unique names. They do not signify the location of the resource. It consists of three parts: Scheme name, Namespace identifier and Namespace string

**27. Differentiate absolute and relative URL.**

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

**28. Distinguish between web servers and web sites.**

Web Sites	Web Servers
A website is a set of linked documents associated with a particular person, organization or topic that is held on a computer system and can be accessed as part of the world wide web.	The web server is a computer program, which delivers content, such as websites or web pages. It responds to the request for web pages.
All the web sites reside on the web server.	The web server is a computer with high configuration.
Web sites can contain text files, images, videos and audios.	Web servers are hardware or software unit.

**29. List the advantages of XHTML.**

- ❖ XHTML documents are XML conforming as they are readily viewed, edited, and validated with standard XML tools.
- ❖ XHTML documents can be written to operate better than they did before in existing browsers as well as in new browsers.
- ❖ XHTML documents can utilize applications such as scripts and applets that rely upon either the HTML Document Object Model or the XML Document Object Model.

**30. Distinguish between XHTML and HTML.**

HTML	XHTML
HTML or Hyper Text Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.	HTML (Extensible HyperText Markup Language) is a family of XML markup languages that mirror or extend versions of the widely used Hypertext Markup Language (HTML), the language in which web pages are written.
It has document file format.	It is a mark- up language.
It is extended from SGML.	It is extended from XML and HTML.
It has flexible framework requiring lenient HTML specific parser.	It is restrictive subset of XML and needs to be parsed with standard XML parsers.

**31. What is CSS?**

CSS helps to define the presentation of HTML elements as a separate file known as CSS file having .css extension. CSS helps to change formatting of any

HTML element by just making changes at one place. All changes made would be reflected automatically to all of the web pages of the website in which that element appeared.

### **32. What is transition in CSS?**

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, the transition cause the changes in a property to take place over a period of time. To create a transition effect, you must specify two things:

- (a) the CSS property you want to add an effect to
- (b) the duration of the effect. If the duration part is not specified, the transition will have no effect, because the default value is 0.

### **33. What are CSS Animations?**

CSS animations make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

### **34. Give the advantages of animations over transitions.**

- They're easy to use for simple animations; you can create them without even having to know JavaScript.
- The animations run well, even under moderate system load.
- Letting the browser control the animation sequence lets the browser optimize performance and efficiency.

## **PART-B**

1. Explain the web fundamentals.
2. Describe the evolution of internet.
3. Explain TCP/IP, FTP, Telnet.
4. Elaborate the working of DNS.
5. Explain HTTP request and response messages.
6. Describe in detail about web clients.
7. Explain about web servers.
8. Explain HTML tags in detail.

9. How is XHTML different from HTML?
10. Describe CSS.
11. Explain transitions and animations in CSS.

Arunai Engineering College

---

---

# 2

## CLIENT SIDE PROGRAMMING

---

---

### 2.1 SCRIPTING LANGUAGES

Scripting languages are becoming more popular due to the emergence of web-based applications. The new scripting languages allow users with little or no programming expertise to develop interactive web pages with minimum effort.

*A scripting language is a programming language that supports scripts and programs that are written for a special run-time environment. They are interpreted rather than compiled.*

An interpreter checks the syntax of the code and generates object code one source line at a time. The language is interpreted at run-time so that the instructions are executed immediately. There are two types of scripting languages: Server side scripting languages and Client side scripting languages

#### Server side scripting Languages

Server side scripting Languages are run on a web server (back end). This environment is known as **Server side scripting environment**. The user issues a request and it is fulfilled by running a script, directly on the web server. The web server will generate dynamic HTML pages as the output according to the script. This HTML is then sent to the client browser. These languages are mostly used in interactive web sites that are connected to databases.

**Examples:** PERL, ASP (Active Server Pages).

#### Advantages of Server side scripting Languages

These languages support high customization of the response based on the user's requirements.

#### Disadvantages of Server side scripting Languages

They impart more load to the web server. They can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and

## 2.2 Client Side Programming

---

recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

### Client side scripting Languages

They run on a browser (front end). This environment is known as **Client side scripting environment**. The processing of the scripts takes place on the end users computer. The source code of the requested service or web page is transferred from the web server to the end users computer over the internet and run directly in the browser. The scripting language must be enabled on the client computer. They make interactive and dynamic webpages. They also interact with temporary storage and local storage and provide remote services for client-side applications. **Examples:** VB script, JavaScript.

### Advantages of Client side scripting Languages

- No load on the server since all the processing is done in the browser.
- These languages are easier than server side scripting languages.

### Disadvantages of Client side scripting Languages

- Minimal customization of web pages.

### Difference between server side and client side scripting languages

Server side Scripting Languages	Client side Scripting Languages
The scripting code is run at the back end (i.e.) at the web server.	The scripting code is run in the back end (i.e) in the end user's browser.
They are less interactive.	They are more interactive.
Any change by these languages will be reflected on the database.	The changes are done only at the client side, so the database will not get affected.
More overhead on the server.	The overhead is on the local browser.
The server side scripts are not visible to the user.	The client side scripts are visible to the user.
They allow a level of privacy and personalization.	The security features are less efficient.

### Advantages of scripting languages:

- Any errors in the scripting language will terminate the execution of the source code. They have a simple syntax. They are easy to learn and use.

- This does not require programming expertise. It allows complex tasks to be performed in relatively few steps. It allows simple creation and editing. It could be done in a variety of text editors
- It facilitates the addition of dynamic and interactive activities to web pages. They are portable across various hardware and network platforms and scripts can be embedded in standard text document also.
- Instantaneous error reporting and error correction. The debugging process is also easy.

**Disadvantages of scripting languages**

- Dubious web sites or unauthorized programs are easily accessed without the user’s knowledge because the executable code is run on the end user’s browser.
- The above action may harm the end user’s system.

**Difference between programming languages and scripting languages**

Programming Languages	Scripting Languages
They are compiled.	They are interpreted.
They cannot be run directly without compilation.	They can be directly run. No explicit compilation is needed.
They have complete syntax and semantic rules.	They are unstructured subset of programming language.
They are used to build applications.	They are used to control the behavior of an application.

**2.2 INTRODUCTION TO JAVASCRIPT**

JavaScript is a client side scripting language developed by Netscape for use within HTML web pages. JavaScript is loosely based on Java and it is built into all the major modern browsers like Internet Explorer, Firefox, Chrome, Safari etc.

**Features of JavaScript**

- JavaScript is a lightweight, interpreted scripting language that is directly embedded into web pages.
- It is used for creating network-centric applications. It is complementary to and integrated with Java and HTML.

## 2.4 Client Side Programming

---

- It is an open and cross-platform scripting language. It adds interactivity to HTML pages.

### Capabilities of JavaScript

- JavaScript acts as a programming tool for web designers. They can add dynamic features into an HTML page.
- JavaScript can react to events. JavaScript can read and write HTML elements and validate input data. JavaScript can be used to create cookies and much more.

### Placement of JavaScript in a HTML File:

The following are the ways to include JavaScript in the HTML file:

- Script in `<head>...</head>` section.
- Script in `<body>...</body>` section.
- Script in `<body>...</body>` and `<head>...</head>` sections. Here JavaScript is included at both head and body of the HTML. The above three are **inline** JavaScripts
- Script in an external file and then include in `<head>...</head>` section. Here the JavaScript is an external file and the JavaScript file is linked with the HTML file in the header section. This is **external** JavaScript.

### Inline JavaScript

In Inline JavaScript, the scripts can be placed **anywhere** on the page. The output of a page will appear where the script block is in the HTML file. For instance if the JavaScript blocks are placed in the header region of the HTML document, then the dynamic content will appear in the header part of the web page and if the script blocks are at the body region of the HTML document, then the dynamic content will appear in the body part of the web page.

It is a good practice to place the scripts at the bottom of the HTML document. The reason is that each time the browser encounters a `<script>` tag it has to pause, compile the script, execute the script, then continue on generating the page. This takes time.

### External JavaScript

External JavaScript allows the **reuse of same block of code** on several different web pages. The JavaScript code will be written on a separate page and the web pages can make use of this code by including the page in the `src` attribute of the script tag.

The biggest advantage to have an external JavaScript file is that once the file has been loaded, the script will remain in the browser's cache area. So the next time the page will be loaded from the browser's cache instead of having to reload it over the Internet. This enables faster execution.

**Syntax:** `<script type='text/javascript' src='filename.js'>`  
`</script>`

When the browser encounters this block it will load `filename.js` and execute it.



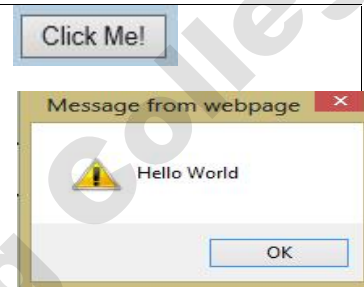
### Advantages of external JavaScript

- It separates HTML and code. It makes HTML and JavaScript easier to read and maintain.
- Cached JavaScript files can speed up page loads.

### External JavaScript

#### external.js

```
function popup()
{alert("Hello World")}
<html><head><script src="external.js"></script></head>
<body>
<input type="button" onclick="popup()" value="Click Me!">
</body></html>
```



The external.js is a JavaScript file. It cannot contain `<script></script>`.

### Differences between inline and external javascript

Inline JavaScript	External JavaScript
The JavaScript code will be embedded in the same html document.	The JavaScript code will be included in the src attribute of the <code>&lt;script&gt;</code> in the html document. The JavaScript code will not be a part of the html document.
Difficult to maintain and slow.	Easy to maintain and faster execution since the external file is stored in browser's cache.

### Creating a simple web page with JavaScript

JavaScript is embedded inside a HTML code. A JavaScript consists of set of JavaScript statements that are placed within the `<script>... </script>` HTML tags in a web page. The `<script>` tag alerts the browser program to begin interpreting all the text between these tags as a script.

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. Usage of semi colons is optional. But it is a good programming practice to use semi colons where ever necessary to enhance the readability of the code. JavaScript is a **case-sensitive** language. The identifiers "CAT" and "cat" are two different tokens in JavaScript because of their cases.

**Syntax:** `<script>` JavaScript code `</script>`

The attributes of the script tag are:

- **Language:**

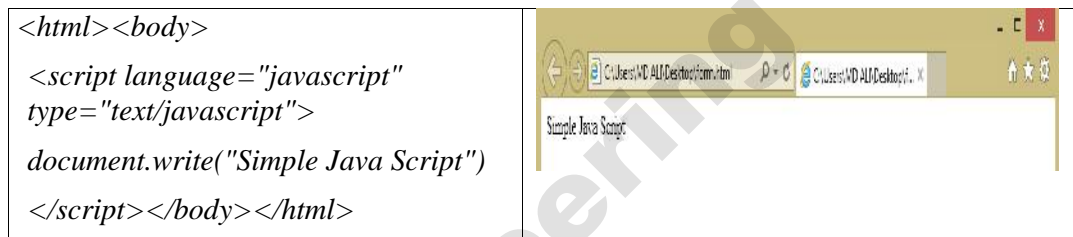
This specifies the scripting language used. In case of JavaScript, the value will be javascript. If other scripting languages are used, then this attribute will take the names of the language used. This is an optional attribute.

- **Type**

This attribute specifies the type of code. Its value should be set to text/javascript. **Example:** `<script language="JavaScript" type="text/javascript">`

JavaScript code `</script>`

### Simple JavaScript



- The first method to use is the `document.writeln(string)`. This is used while the web page is being constructed. After the page has finished loading a new `document.writeln(string)` command will delete the page in most browsers.
- As the page is loading, JavaScript will encounter this script and it will output " Simple Java Script " exactly where the script block appears on the page. The problem with `writeln` is that if this method is used after the page has loaded the browser will destroy the page and start constructing a new one.

### Comments in JavaScript

JavaScript supports both C-style and C++-style comments. The text between `//` and the end of a line is treated as a comment and is ignored by JavaScript. This is single line comment. The text between the characters `/*` and `*/` is treated as a comment. This is multi-line comment. JavaScript also recognizes the HTML comment opening sequence `<!--`. JavaScript treats this as a single-line comment. The closing of HTML comment should be written as `-->`.

### Data types

JavaScript allows three primitive data types: Numbers, Strings and Boolean. JavaScript also defines two trivial data types, null and undefined both define only a single value.

**Reserved words**

The reserved words or keywords cannot be used as JavaScript variables, functions, methods, loop labels, or any object names. The following are the keywords in JavaScript:

Abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

**JavaScript Variables**

Variables are named containers. JavaScript is not a strongly typed language. The programmer needs to care only what the variable is storing. In JavaScript the variables can store anything, even functions. Before using a variable in a JavaScript program, it must be declared.

**Syntax:** `var variable_name;`

Here `var` is the keyword and is optional. Any variable in JavaScript is declared without specifying its data type. The variable takes the type of the value it holds.

1. `var s = 'This is a string';` //now s is of string data type
2. `var s = 25;` //now s is of number or integer data type
3. `var s = true;` // now s is of Boolean data type.
4. `var s = [0, 'one', 2, 3, '4', 5];` // now s is of array data type
5. `var s = { 'color': 'red' }` //now s is of object data type. Color is a JavaScript object
6. `var s = function()`

```

{           return "example function"           }

```

// The compiler executes the function and stores the return value of the function which is// " example function" in the variable s.

### Naming variables

- Keywords in JavaScript cannot be used as a valid variable name. JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or the underscore character. JavaScript variable names are case sensitive.

### Scope of a variable

The lifetime of the JavaScript variables starts when they are declared, and ends when the page is closed. The **scope of a variable** is the region of the program in which it is defined. JavaScript variables will have only two scopes:

**Global Variables:** A global variable has global scope which means it can be accessed everywhere in the JavaScript code of the web page. If a function defines a new variable without using the var keyword, that variable will be a global variable.

**Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

### Special Keywords

JavaScript has a few pre-defined variables with special or fixed meaning. The following are those special keywords:

- NaN (Not a Number)-This is generated when an arithmetic operation returns an invalid result.
- Infinity is a keyword which is returned when an arithmetic operation overflows JavaScript's precision which is in the order of 300 digits.
- Null is a reserved word that means "empty". In boolean operations null evaluates as false. JavaScript supports true and false as boolean values.
- Undefined value-If a variable has not been declared or assigned yet then that variable will be given a special undefined value. In boolean operations undefined evaluates as false.

**Arithmetic Operators:** The following are the arithmetic operators supported by JavaScript: +, -, \*, /, % (modulus), ++ and \_\_

**Comparison Operators:** Javascript supports ==, !=, >, <, >= and <= operators.

**Logical Operators:** The following are the logical operators supported by JavaScript: &&, || and !.

**Bitwise Operators:** The following are the bitwise operators supported by JavaScript: &, | and ^.

**Assignment Operators:** The following are the assignment operator formats supported by JavaScript: =, +=, -=, \*=, /=, %=.

**Conditional Operator or ternary operator (? :):** This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.

if (operand1 conditional operator operand 2)? statement1 :statement2

**Example:**if(a= =b)?1:0

### typeof Operator

The typeof is a unary operator. This operator returns the data type of the operand. The typeof operator valuates to number, string, or boolean depending on the value taken by the operand.

<b>Syntax:</b> typeof(operand)	<b>Example:</b> typeof(1) //This returns number
--------------------------------	---

### JavaScript Statements

Statements define what the script will do and how it will be done. The end of a statement is indicated with a semicolon(;). The following are the types of statements in JavaScript:Conditional Statements, Loop Statements, Object Manipulation Statements, Comment Statements and Exception Handling Statements

#### Comment Statements

Comment statements are used to prevent the browser from executing certain parts of code that you designate as non-code. The single line comment is just two slashes (//) and the multiple line comment starts with (/\*) and ends with (\*).

#### Exception Handling Statements

These statements are safety mechanisms, so that the code handles common problems that may arise. The try...catch statement tries to execute a piece of code and if it fails, the catch should handle the error gracefully.

#### Conditional Statements

Java script supports the following conditional control statements:Simple if, If else, If else ladder and Switch

##### a) Simple if

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

##### b) if else statement

The if...else statement is a form of control statement that allows JavaScript to execute statements in more controlled way. If the condition evaluates to true then one block of statements will get executed and if the condition is false other block of statements will get executed.

**c) if-else ladder**

The if else ladder statement is an advanced form of control statement that allows JavaScript to make correct decision out of several conditions. A normal If Statement must be placed before the use the else If statement. This is because the else if statement is an add- on to the simple if Statement. Any number of else if statements can be included in a program.

**If-else ladder**

<pre>&lt;html&gt;&lt;script type="text/javascript"&gt; var a= "first letter"; if(a == "first number") {document.write("I is the first natural number");} else if(a == "first letter") {     document.write("a is the first letter");} else {     document.write("nothing");} &lt;/script&gt;&lt;/html&gt;</pre>	<b>Output:</b> a is the first letter
---	---

**d) Switch statement**

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

<pre>&lt;html&gt; &lt;script type="text/javascript"&gt; var grade='A'; switch (grade) { case 'A':document.write("Distinction&lt;br /&gt;"); break; case 'B': document.write("First Class&lt;br /&gt;"); break; case 'C': document.write("Second Class&lt;br /&gt;");break; case 'F':document.write("Failed&lt;br /&gt;"); break; default: document.write("Did not appear for exams &lt;br /&gt;") }&lt;/script&gt;&lt;/html&gt;</pre>	<b>Output:</b> Distinction
---	-------------------------------

## Looping Statements

The following are the looping statements in JavaScript: While loop, Do while loop and For loop

### a) While loop

The most basic loop in JavaScript is the while loop. There are two key parts to a JavaScript while loop: The conditional statement which must be true for the while loop's code to be executed. The while loop's code that is contained in curly braces "{ and }" will be executed if the condition is True.

When a while loop begins, the JavaScript interpreter checks if the condition statement is true. If it is, the code between the curly braces is executed. The same procedure is repeated until the condition stays true. If the condition statement is always True, then you will never exit the while loop.

### b) do-while loop

The do...while loop is similar to the while loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is false.

### Do-while

<pre> &lt;html&gt; &lt;script type="text/javascript"&gt; var loop= 0; varlinebreak = "&lt;br /&gt;"; do{     document.write("loop= " + loop);     document.write(linebreak);     loop++; }while(loop&lt;5); &lt;/script&gt;&lt;/html&gt; </pre>	<p><b>Output:</b></p> <pre> loop = 0 loop = 1 loop = 2 loop = 3 loop = 4 loop = 5 </pre>
---	--

### c) for loop

The for loop is the most compact form of looping and includes the following three important parts: The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins. The **test statement** which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out. The **iteration** statement where counter value is incremented or decremented.

**For loop**

<pre>&lt;html&gt;&lt;script type="text/javascript"&gt; varlinebreak = "&lt;br /&gt;"; loop=0; for(i = 0; i &lt; 4; i++) {     document.write("Counter = " + i);     document.write(linebreak); }&lt;/script&gt;&lt;/html&gt;</pre>	<p><b>Output:</b></p> <p><b>Output:</b></p> <p>Counter = 0</p> <p>Counter = 1</p> <p>Counter = 2</p> <p>Counter = 3</p>
--	---

**Break and Continue Statements**

The break statement is used to exit a loop early. It breaks the execution of the code from that block. The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block. When a continue statement is encountered, program flow will move to the loop check expression immediately and if condition remain true then it start next iteration otherwise control comes out of the loop.

**Functions in JavaScript**

A JavaScript function contains some code that will be executed only by an event or by a call to that function. The function can be called from anywhere within the page or from other external pages. Functions can be defined either <head> or <body>. The most common way to define a function (optional), and a statement block surrounded by curly braces. As a convention, they are typically defined in the <head> section.

**Syntax:** <script type="text/javascript">

```
Function functionname(parameter-list)
{ statements}
</script>
```

**Calling a Function:**

The syntax to invoke a function is:

```
<script type="text/javascript">functionname(parameter-list) </script>
```

**Functions**

<pre>&lt;html&gt;&lt;head&gt;&lt;script type="text/javascript"&gt; function firstfunction() {document.write("This is my first function")}</pre>	<p><b>Output:</b></p> <p><input type="button" value="Click Me!"/></p> <p>This is my first function</p>
---	--



```

</script></head>

<body><form>

<input type="button" value="Click me!"
onclick="firstfunction()" >

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

```

### Dialog boxes

JavaScript supports three types of dialog boxes: Alert dialog box, Prompt dialog box and Confirmation dialog box

#### Alert dialog box

An alert dialog box is used to give a warning message to the users. It pops up a message box displaying some contents with an OK button. JavaScript alerts are used in the following situations:

- To see a message before doing anything on the website.
- To warn the user about something.
- It can be used as an error indication.

#### Alert Dialog box

```

<head>
<script type="text/javascript">
  alert("Hello there");
</script></head>

```



#### Confirmation Dialog Box:

A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method confirm() will return true. If the user clicks on the Cancel button confirm() returns false.

#### Confirmation dialog box

```

<head><script type="text/javascript">
varretVal = confirm("Are you sure you want to delete this
record ?");
if( retVal == true ){
alert("User wants to delete!");return true; }

```



## 2.14 Client Side Programming

```
else{  
  alert("User does not want to delete!"); return false; }  
</script></head>
```

### Prompt Dialog Box:

The prompt dialog box is very useful when a pop-up text box to used to get user input. Thus it enables to interact with the user. The user needs to fill in the field and then click OK. This dialog box is displayed using a method called prompt() which takes two parameters:

- (i) A label which you want to display in the text box
- (ii) A default string to display in the text box.

This dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method prompt() will return entered value from the text box. If the user clicks on the Cancel button the window method prompt() returns null.

### Prompt dialog box

```
<head>  
<script type="text/javascript">  
varretVal = prompt("Enter your name :");  
</script></head>
```



## 2.3 JAVASCRIPT DOCUMENT OBJECT MODEL (DOM)

The Document Object Model (DOM) is a programming API for HTML and XML documents.

*DOM defines the logical structure of documents and the way a document is accessed and manipulated. It specifies how the relationships among objects used in the web page must be implemented.*

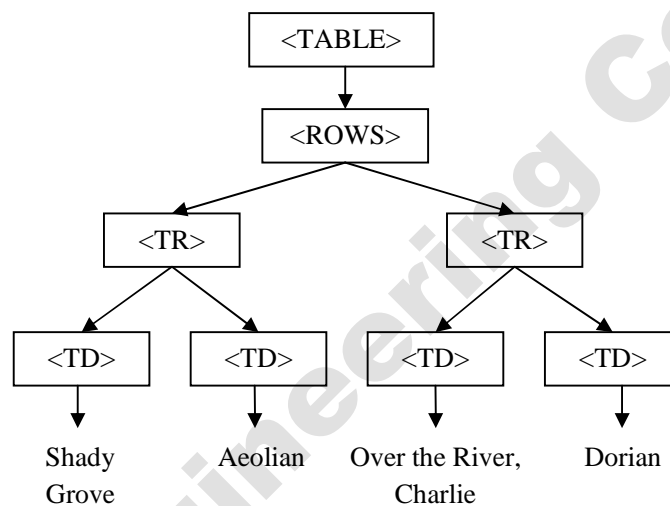
DOM is a standard way of referring to an XML and HTML documents. In the Document Object Model, documents (contents of the web page) have a logical structure which is very much like a tree. With the Document Object Model, programmers can create and build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model. The Document Object Model identifies:

- the interfaces and objects used to represent and manipulate a document
- the behavior and attributes of the interfaces and objects
- the relationships and collaborations among these interfaces and objects

Consider the following HTML document

```
<TABLE><ROWS><TR><TD>Shady Grove</TD>
<TD>Aeolian</TD></TR>
<TR><TD>Over the River, Charlie</TD>
<TD>Dorian</TD></TR>
</ROWS></TABLE>
```

The DOM representation of the above table is:



**Figure 2.1 DOM of a table**

HTML elements are represented by tree nodes and organized into a hierarchy. Every element in the DOM tree is called a node. Each Web page has a document node at the root of the tree. It is the top most node. The head and body nodes become child nodes of the document node.

Traversals to the nodes can be done from a node on the DOM tree, to any child node or go up and vice versa.

### Architecture of DOM

The DOM architecture consists of following modules:

- **DOM Core-** Specifies the DOM tree, tree nodes, its access, traversal, and manipulation. The DOM Range and DOM Traversal modules provide higher-level methods for manipulating the DOM tree defined by the Core.

## 2.16 Client Side Programming

---

- **DOM HTML**-inherits from the Core and provides specialized and convenient ways to access and manipulate HTML/XHTML documents.
- **DOM Events**-Specifies events and event handling for user interfaces and the DOM tree. With DOM Events, drag and drop programs, for example, can be standardized.
- **DOM CSS**- Specifies easy to use ways to manipulate Cascading Style Sheets for the formatting and presentation of documents.

### DOM Levels

DOM Levels are the versions of the specification for defining how the Document Object Model should work. The most recent spec is DOM Level 3, published in April 2004.

DOM Level	Description
Level 0	It offers access to a few HTML elements, most importantly forms and (later) images. The Level 0 DOM was invented by Netscape.
Level 1	W3C had developed the Level 1 DOM specification. In DOM 1 it is also possible to change the entire web page.
Level 2	DOM level 2 is a platform and language neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The DOM Level 2 is made of a set of core interfaces to create and manipulate the structure and contents of a document and a set of optional modules.  These modules contain specialized interfaces dedicated to XML, HTML, an abstract view, generic stylesheets, Cascading Style Sheets, Events, traversing the document structure, and a Range object.
Level 3	Level DOM is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. This has five specifications: Core, load and save, events, validation and Xpath.

### DOM Programming Interface

The HTML DOM can be accessed with JavaScript and with other programming languages. The following are some of the terms used in DOM programming:

- HTML DOM **methods** are actions performed on HTML elements. HTML DOM **properties** are values of HTML Elements. These values can be set and reset.
- In the DOM, all HTML elements are defined as **objects**. The **programming interface** is the properties and methods of each object.
- With the HTML DOM, the document object is the web page.

**DOM illustrating method and property**

<pre>&lt;html&gt;&lt;body&gt; &lt;p id="demo"&gt;&lt;/p&gt;&lt;script&gt; document.getElementById("demo").innerHTML = "hai!"; &lt;/script&gt;&lt;/body&gt;&lt;/html&gt;</pre>	
---	---

In the example above, `getElementById()` is a method, while `innerHTML` is a property.

- **getElementById()**

This is the most common way to access an HTML element is to use the id of the element. In the example above the `getElementById` method used `id="demo"` to find the element.

- **The innerHTML Property**

The easiest way to get the content of an element is by using the `innerHTML` property. The inner HTML property is useful for getting or replacing the content of HTML elements. Each HTML element has an inner HTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag.

**inner HTML**

<pre>&lt;script type="text/javascript"&gt; function changeText() {document.getElementById('boldStuff').innerHTML = 'Sharanya';} &lt;/script&gt; &lt;p&gt;Welcome to the site &lt;b id='boldStuff'&gt;dear &lt;/b&gt;&lt;/p&gt; &lt;input type='button' onclick='changeText()' value='Change Text'/&gt;</pre>	<p>Welcome to the site <b>dear</b></p> <p><input type="button" value="Change Text"/></p> <p>Welcome to the site <b>Sharanya</b></p> <p><input type="button" value="Change Text"/></p>
--	---

**The HTML DOM Document**

In the HTML DOM object model, the document object represents the web page. The document object is the owner of all the other objects in the web page and it is the root node of the DOM tree. To access any object in an HTML page, always start with accessing the document object.

### Finding HTML Elements

The following methods are used to find the HTML elements in any document:

- `document.getElementById()`-Finds an element by element id  
**Example:** `var x = document.getElementById("intro");`
- `document.getElementsByTagName()`-Find elements by tag name  
**Example:** `var y = x.getElementsByTagName("p");`
- `document.getElementsByClassName()`-Find elements by class name  
**Example:** `document.getElementsByClassName("intro");`
- Finding HTML Elements by HTML Object Collections

### Finding HTML elements

```

<html><head><script type="text/javascript">
function getGroup(name)
{
varelemArray = document.getElementsByName(name);
  for(var i = 0; i <elemArray.length; i++)
  {
varelem = document.getElementById(elemArray[i].id);
alert(elem.id+" | "+elem.checked+" | "+elem.value); }}
</script></head>
<body>
<input name="cbGroup1" type="checkbox" id="cb1"
value="Cat"> Cat <br />
<input name="cbGroup1" type="checkbox" id="cb2"
value="Dog"> Dog <br />
<input name="cbGroup1" type="checkbox" id="cb3"
value="Bird"> Bird <br />
<button onclick="getGroup('cbGroup1');">Evaluate Checkbox
Group</button>
</body></html>

```

 Cat  
 Dog  
 Bird

### Changing HTML Elements

- `element.innerHTML`-Changes the inner HTML of an element
- `element.attribute`-Changes the attribute of an HTML element
- `element.setAttribute(attribute,value)`-Changes the attribute of an HTML element
- `element.style.property`-Changes the style of an HTML element

### Adding and Deleting Elements

Methods	Description
<code>document.createElement()</code>	Create an HTML element.
<code>document.removeChild()</code>	Remove an HTML element.
<code>document.appendChild()</code>	Add an HTML element.
<code>document.replaceChild()</code>	Replace an HTML element.
<code>document.write(text)</code>	Write into the HTML output stream.

### Adding Events Handlers

Events are normally used in combination with functions, and the function will not be executed before the event occurs. The following method is used to add event handlers in any document:

```
document.getElementById(id).onclick=function { code }
```

Adding EventAdding event handler code to an onclick event. The event handler could be added to any event

#### **addEventListener()**

The `addEventListener()` method attaches an event handler to the specified element. The `addEventListener()` method attaches an event handler to an element without overwriting existing event handlers. The event listeners can be added to any DOM object not only HTML elements. i.e the window object. When using the `addEventListener()` method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

```
element.addEventListener(event, function, useCapture);
```

- The first parameter is the type of the event (like "click" or "mousedown").
- The second parameter is the function we want to call when the event occurs.
- The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

### removeEventListener() method

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

```
element.removeEventListener("mousemove", myFunction);
```

### Changing the objects of an HTML document

#### Changing HTML content

The easiest way to modify the content of an HTML element is by using the innerHTML property.

Syntax: document.getElementById(id).innerHTML = new HTML

#### Changing HTML content

<pre>&lt;html&gt;&lt;body&gt; &lt;p id="p1"&gt;Old content&lt;/p&gt; &lt;script&gt; document.getElementById("p1").innerHTML     = 'Modified content' &lt;/script&gt;&lt;/body&gt;&lt;/html&gt;&lt;/ht&gt;</pre>	Modified content
---	------------------

The HTML document above contains an <p> element. We use the HTML DOM to get the element with id="p1". A JavaScript changes the content (innerHTML) of that element.

#### Changing the Value of an Attribute

```
document.getElementById(id).attribute=new value
```

#### Changing the attribute's value

<pre>&lt;html&gt;&lt;body&gt; &lt;img id="Image1" src="cat.gif"&gt;&lt;script&gt; document.getElementById("Image1").src = "dog.jpg";/script&gt;&lt;/body&gt;</pre>
--

The HTML document above contains an <img> element with id="Image1". We use the HTML DOM to get the element with id="Image1". A JavaScript changes the src attribute of that element from "cat.gif" to "dog.jpg". Now the image id "Image1" will point to dog.jpg.

#### Changing HTML Style

The HTML DOM allows JavaScript to change the style of HTML elements.

```
document.getElementById(id).style.property=new style
```



**Changing HTML style**

<pre>&lt;html&gt;&lt;body&gt;  &lt;p id="p2"&gt;Hello &lt;p&gt;  &lt;script&gt;document.getElementById("p2").style.color "red";&lt;/script&gt;  &lt;p&gt;The paragraph above was changed by a script.&lt;/p&gt;  &lt;/body&gt;&lt;/html&gt;</pre>	<p>Hello</p> <p>The paragraph above was changed by a script.</p>
---	--

**Creating New HTML Elements (Nodes)**

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<div id="div1">
<p id="p1">This is first paragraph.</p>
<p id="p2">This is second paragraph.</p>
</div>
<script>
var para = document.createElement("p");
var node = document.createTextNode("This is third paragraph.");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

This code creates a new <p> element: `var para = document.createElement("p");` To add text to the <p> element, create a text node first. This code creates a text node:

```
var node = document.createTextNode("This is a new paragraph.");
```

Then append the text node to the <p> element: `para.appendChild(node);` Finally find the newly created element and append the to an existing element. This code finds an existing element:

```
var element = document.getElementById("div1");
element.appendChild(para);
```

**Removing Existing HTML Elements**

To remove an HTML element, find the parent of the element and then delete it.

<pre> &lt;div id="div1"&gt;   &lt;p id="p1"&gt;This is first paragraph.&lt;/p&gt;   &lt;p id="p2"&gt;This is second paragraph.&lt;/p&gt; &lt;/div&gt; &lt;script&gt;   Var parent =     document.getElementById("div1");   var child = document.getElementById("p1");   parent.removeChild(child); &lt;/script&gt; </pre>	<p>This is first paragraph.</p> <p>This is second paragraph.</p> <p>This is second paragraph.</p>
---	---

Find the element with id="div1":    var parent = document.getElementById("div1");

- Find the <p> element with id="p1":    var child = document.getElementById("p1");
- Remove the child from the parent:    parent.removeChild(child);
- Another way is to find the child to be removed, and use its parentNode property to find the parent:

```

var child = document.getElementById("p1");
child.parentNode.removeChild(child);

```

### Replacing HTML Elements

To replace an element to the HTML DOM, use the replaceChild() method:

```

<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para,child);
</script>

```

## 2.4 OBJECTS IN JAVASCRIPT

JavaScript is an Object Oriented Programming (OOP) language. A programming language is called object-oriented if it has the following four basic capabilities: encapsulation, aggregation, inheritance and polymorphism. All the objects will have properties and methods.

### Object Properties

Object properties can be any of the primitive data types or abstract data types. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that can be used throughout the page.

```
objectName.objectProperty = propertyValue;
```

**Example:** `varstr = document.title;`

### Object Methods

The **methods** are functions that let the object do something or let something be done to it. A function is a standalone unit of statements and a method is attached to an object and can be referenced by the keyword. Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

**Example:** `document.write("This is a method of the object document");`

### User-Defined Objects

Apart from built-in objects, the users can also create their own objects. All user-defined objects and built-in objects are descendants of an object called **Object**. The new operator is used to create a new object.

### The new Operator

The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

**Example:** `var books = new Array("Thirukural", "Geethai");`

In the above example `Array()` is a built-in object. `Books` is the instance of the object `Array()`.

### The Object() Constructor

A **constructor** is a function that creates and initializes an object. The `Object()` constructor is used to build an object. The return value of the `Object()` constructor is assigned to a variable. The variable contains a reference to the new object. The properties assigned to the object are not variables. These properties can be accessed only through objects.

```
Objectname.property
```

### Creating an object

<pre> &lt;html&gt;&lt;head&gt;&lt;script type="text/javascript"&gt; var book = new Object(); // Create the object     book.name = "PonniyinSelvan"; // Assign properties to the         object book.author = "Kalki"; &lt;/script&gt;&lt;/head&gt; &lt;body&gt;&lt;script type="text/javascript"&gt; document.write("Book name is : " + book.name + "&lt;br&gt;"); document.write("Book author is : " + book.author + "&lt;br&gt;"); &lt;/script&gt;&lt;/body&gt;&lt;/html&gt; </pre>	<pre> Book name is : Ponniyin Selvan Book author is : Kalki </pre>
---	--

### JavaScript Native Objects

JavaScript has several built-in or native objects. These objects are accessible anywhere in the program as the other objects. The following are some of the important JavaScript Native Objects: JavaScript Number Object, JavaScript Boolean Object, JavaScript String Object, JavaScript Array Object, JavaScript Date Object, JavaScript Math Object, JavaScript RegExp Object

#### 2.4.1 JavaScript Number Object

The Number object represents numerical date, either integers or floating-point numbers. The browser automatically converts number literals to instances of the number class.

```
var val = new Number(number);
```

If the argument cannot be converted into a number, it returns NaN (Not-a-Number).

#### Number Properties

Property	Description
MAX_VALUE	The largest possible value a number in JavaScript (1.7976931348623157E+308).
MIN_VALUE	The smallest possible value a number in JavaScript can have (5E-324)
NaN	Equal to a value that is not a number.
NEGATIVE_INFINITY	A value that is less than MIN_VALUE.

POSITIVE_INFINITY	A value that is greater than MAX_VALUE
Prototype	A static property of the Number object. This is used to assign new properties and methods to the Number object in the current document.

### Number Methods

Method	Description
Number()	Returns the number object.
toExponential()	Forces a number to display in exponential notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
toPrecision()	Defines how many total digits to display of a number (including digits to the before and after the decimal).
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

### 2.4.2 JavaScript String object

`var val = new String(string);` // The string parameter is series of characters.

#### String Properties

- length -Returns the length of the string.
- Prototype-The prototype property allows you to add properties and methods to an object.

#### String Methods

Method	Description
String()	Returns the string object.
charAt(i)	Returns the character at the specified index.

## 2.26 Client Side Programming

---

<code>charCodeAt(index)</code>	Returns a number indicating the Unicode value of the character at the given index.
<code>concat()</code>	Combines the text of two strings and returns the combined string.
<code>indexOf()</code>	Returns the index of the first occurrence of the String object or -1 if not found.
<code>lastIndexOf()</code>	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
<code>localeCompare()</code>	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
<code>match()</code>	Used to match a regular expression against a string.
<code>replace()</code>	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring
<code>search()</code>	Executes the search for a match between a regular expression and a specified string.
<code>slice()</code>	Extracts a section of a string and returns a new string.
<code>split()</code>	Splits a String object into an array of strings by separating the string into substrings.
<code>substr()</code>	Returns the characters in a string beginning at the specified location through the specified number of characters.
<code>substring()</code>	Returns the characters in a string between two indexes into the string.
<code>toLocaleLowerCase()</code>	The characters within a string are converted to lower case while respecting the current locale.
<code>toLocaleUpperCase()</code>	The characters within a string are converted to upper case while respecting the current locale.
<code>toLowerCase()</code>	Returns the calling string value converted to lower case.
<code>toString()</code>	Returns a string representing the specified object.
<code>toUpperCase()</code>	Returns the calling string value converted to uppercase.
<code>valueOf()</code>	Returns the primitive value of the specified object.

## String HTML wrappers

The functionalities of these wrappers are similar to the HTML tags.

Method	Description
anchor()	Creates an HTML anchor that is used as a hypertext target.
big()	Creates a string to be displayed in a big font as if it were in a <big> tag.
blink()	Creates a string to blink as if it were in a <blink> tag.
bold()	Creates a string to be displayed as bold as if it were in a <b> tag.
fixed()	Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag.
fontcolor()	Causes a string to be displayed in the specified color as if it were in a <font color="color"> tag.
fontsize()	Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
italics()	Causes a string to be italic, as if it were in an <i> tag.
link()	Creates an HTML hypertext link that requests another URL.
small()	Causes a string to be displayed in a small font, as if it were in a <small> tag.
strike()	Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
sub()	Causes a string to be displayed as a subscript, as if it were in a <sub> tag.
sup()	Causes a string to be displayed as a superscript, as if it were in a <sup> tag.

## JavaScript Array object

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

**Example:** `var fruits = new Array( "apple", "orange", "mango" );`

The Array parameter is a list of strings or integers. The maximum length allowed for an array is 4,294,967,295.

### Array Properties

- `index` - The property represents the zero-based index of the match in the string

## 2.28 Client Side Programming

---

- input - This property is only present in arrays created by regular expression matches.
- length - Reflects the number of elements in an array.
- Prototype - The prototype property allows you to add properties and methods to an object.

### Array Methods

Method	Description
Array()	Returns a reference to the array function that created the object.
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or
join()	Joins all elements of an array into a string.
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left
reduceRight()	Apply a function simultaneously against two values of the array (from right
reverse()	Reverses the order of the elements of an array



shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

### JavaScript Math Object

The math object provides the properties and methods for mathematical constants and functions. Unlike the other global objects, Math is not a constructor. All properties and methods of Math are static and can be called by using Math as an object without creating it.

```
var pi_val = Math.PI; //Math object need not be created
```

```
var sine_val = Math.sin(30);
```

### Math Properties

Property	Description
E	Euler's constant and the base of natural logarithms, approximately 2.718.
LN2	Natural logarithm of 2, approximately 0.693.
LN10	Natural logarithm of 10, approximately 2.302.
LOG2E	Base 2 logarithm of E, approximately 1.442.
LOG10E	Base 10 logarithm of E, approximately 0.434.
PI	Ratio of the circumference of a circle to its diameter, approximately 3.14159.
SQRT1_2	Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
SQRT2	Square root of 2, approximately 1.414.

**Math Methods**

Method	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
asin()	Returns the arcsine (in radians) of a number.
atan()	Returns the arctangent (in radians) of a number.
atan2()	Returns the arctangent of the quotient of its arguments.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns the cosine of a number.
exp()	Returns EN, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
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.
abs()	Returns the absolute value of a number.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo
round()	Returns the value of a number rounded to the nearest integer.
sin()	Returns the sine of a number.
sqrt()	Returns the square root of a number.
tan()	Returns the tangent of a number.
toSource()	Returns the string "Math".

**Regular expressions and regular objects**

A regular expression is an object that describes a pattern of characters. The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that

use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

```
var pattern = new RegExp(pattern, attributes);
```

```
var pattern = /pattern/attributes;
```

- **pattern:** A string that specifies the pattern of the regular expression or another regular expression.
- **attributes:** An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

### Brackets:

Brackets ([ ]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

- [...] - Any one character between the brackets.
- [^...] - Any one character not between the brackets.
- [0-9] - It matches any decimal digit from 0 through 9.
- [a-z] - It matches any character from lowercase a through lowercase z.
- [A-Z] - It matches any character from uppercase A through uppercase Z.
- [a-Z] - It matches any character from lowercase a through uppercase Z.

### Quantifiers:

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character have a specific connotation. The +, \*, ?, and \$ flags all follow a character sequence.

- p+ -It matches any string containing at least one p.
- p\*-It matches any string containing zero or more p's.
- p?-It matches any string containing one or more p's.
- p{N} -It matches any string containing a sequence of N p's
- p{2,3}-It matches any string containing a sequence of two or three p's.
- p{2, } -It matches any string containing a sequence of at least two p's.
- p\$-It matches any string with p at the end of it.
- ^p-It matches any string with p at the beginning of it.

**RegExp Properties**

Property	Description
Global	Specifies if the "g" modifier is set.
ignoreCase	Specifies if the "i" modifier is set.
lastIndex	The index at which to start the next match.
Multiline	Specifies if the "m" modifier is set.
Source	The text of the pattern.
Global	Specifies if the "g" modifier is set.

**RegExp Methods**

Method	Description
exec()	Executes a search for a match in its string parameter.
test()	Tests for a match in its string parameter.
toSource()	Returns an object literal representing the specified object; you can use this value to create a new object.
toString()	Returns a string representing the specified object.

**JavaScript Date Object**

The Date object is a data type built into the JavaScript language. Once a Date object is created, a number of methods are available to operate on it.

- `new Date()` - This constructor creates a Date object set to the current date and time.
- `new Date(milliseconds)`- When one numeric argument is passed, it is taken as the internal numeric representation of the date in milliseconds, as returned by the `getTime()` method. For example, passing the argument 5000 creates a date that represents five seconds past midnight on 1/1/70
- `new Date(datestring)` - When one string argument is passed, it is a string representation of a date, in the format accepted by the `Date.parse()` method.
- `new Date(year,month,date[,hour,minute,second,millisecond ])` -The parameters on square brackets are optional . The description of the arguments are listed below:

1. year: Integer value representing the year. For compatibility (in order to avoid the Y2K problem), you should always specify the year in full; use 1998, rather than 98.
2. month: Integer value representing the month, beginning with 0 for January to 11 for December.
3. date: Integer value representing the day of the month.
4. hour: Integer value representing the hour of the day (24-hour scale).
5. minute: Integer value representing the minute segment of a time reading.
6. second: Integer value representing the second segment of a time reading.
7. millisecond: Integer value representing the millisecond segment of a time reading.

**Date Properties:**

- constructor- Specifies the function that creates an object's prototype.
- Prototype- The prototype property allows you to add properties and methods to an object

**Date Methods:**

Method	Description
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
getMilliseconds()	Returns the milliseconds in the specified date according to local time.
getMinutes()	Returns the minutes in the specified date according to local time.
getMonth()	Returns the month in the specified date according to local time.

### 2.34 Client Side Programming

---

getSeconds()	Returns the seconds in the specified date according to local time.
getTime()	Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
Date()	Returns today's date and time
getDate()	Returns the day of the month for the specified date according to local time.
getDay()	Returns the day of the week for the specified date according to local time.
getFullYear()	Returns the year of the specified date according to local time.
getHours()	Returns the hour in the specified date according to local time.
getTimezoneOffset()	Returns the time
getUTCDate()	Returns the day (date) of the month in the specified date according to universal time.
getUTCDay()	Returns the day of the week in the specified date according to universal time.
getUTCFullYear()	Returns the year in the specified date according to universal time.
getUTCHours()	Returns the hours in the specified date according to universal time.
getUTCMilliseconds()	Returns the milliseconds in the specified date according to universal time.
getUTCMinutes()	Returns the minutes in the specified date according to universal time.
getUTCMonth()	Returns the month in the specified date according to universal time.
getUTCSeconds()	Returns the seconds in the specified date according to universal time.
getYear() Deprecated	Returns the year in the specified date according to local time.
getTimezoneOffset()	Returns the time

<code>getUTCDate()</code>	Returns the day (date) of the month in the specified date according to universal time.
<code>getUTCDay()</code>	Returns the day of the week in the specified date according to universal time.
<code>getUTCFullYear()</code>	Returns the year in the specified date according to universal time.
<code>setDate()</code>	Sets the day of the month for a specified date according to local time.
<code>setFullYear()</code>	Sets the full year for a specified date according to local time.
<code>setHours()</code>	Sets the hours for a specified date according to local time.
<code>setMilliseconds()</code>	Sets the milliseconds for a specified date according to local time.
<code>setMinutes()</code>	Sets the minutes for a specified date according to local time.
<code>setMonth()</code>	Sets the month for a specified date according to local time.
<code>setSeconds()</code>	Sets the seconds for a specified date according to local time.
<code>setTime()</code>	Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.
<code>setDate()</code>	Sets the day of the month for a specified date according to local time.
<code>setFullYear()</code>	Sets the full year for a specified date according to local time.
<code>setHours()</code>	Sets the hours for a specified date according to local time.
<code>setMilliseconds()</code>	Sets the milliseconds for a specified date according to local time.
<code>setUTCDate()</code>	Sets the day of the month for a specified date according to universal time. <code>setUTCFullYear()</code>
<code>toLocaleDateString()</code>	Returns the "date" portion of the Date as a string, using the current locale's conventions.
<code>toLocaleFormat()</code>	Converts a date to a string, using a format string.
<code>toLocaleString()</code>	Converts a date to a string, using the current locale's conventions.

## 2.36 Client Side Programming

---

<code>toLocaleTimeString()</code>	Returns the "time" portion of the Date as a string, using the current locale's conventions.
<code>toSource()</code>	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.
<code>toString()</code>	Returns a string representing the specified Date object.
<code>toTimeString()</code>	Returns the "time" portion of the Date as a human
<code>toUTCString()</code>	Converts a date to a string, using the universal time convention.
<code>valueOf()</code>	Returns the primitive value of a Date object.
<code>setUTCDate()</code>	Sets the day of the month for a specified date according to universal time. <code>setUTCFullYear()</code>
<code>toLocaleDateString()</code>	Returns the "date" portion of the Date as a string, using the current locale's conventions.
<code>toLocaleFormat()</code>	Converts a date to a string, using a format string.
<code>toLocaleString()</code>	Converts a date to a string, using the current locale's conventions.
<code>toLocaleTimeString()</code>	Returns the "time" portion of the Date as a string, using the current locale's conventions.
<code>toSource()</code>	Returns a string representing the source for an equivalent Date object; you can use this value to create a new object.

### Date objects

```
<html><body><script type="text/javascript">
var d = new Date()
document.write(d.getDate())document.write(".")
document.write(d.getMonth() + 1)document.write(".")
document.write(d.getFullYear())d.setFullYear("1990")document.write(".")
document.write(d.getUTCHours())document.write(".")
document.write(d.getUTCMinutes() + 1)document.write(".")
document.write(d.getUTCSeconds())
var weekday=new
```



```

Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
document.write("Today is " + weekday[d.getDay()])
varweekday=new
Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
varmonthname=new
Array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")
document.write(weekday[d.getDay()] + " ")
document.write(d.getDate() + ". ")
document.write(monthname[d.getMonth()] + " ")
document.write(d.getFullYear())
</body></html>

```

22.12.2014.10.9.11Today is SaturdaySaturday 22. Dec 1990

## 2.7 VALIDATION IN JAVASCRIPT

Form validation occurs at the server, after the client had entered all necessary data and then pressed the Submit button. If some of the data that had been entered by the client had been in the wrong form or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This is a lengthy process and over burdening server. JavaScript, provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions:

**Basic Validation** - Checking the form to make sure that the data entered is right.

**Data Format Validation** – Checking the data entered in the form for right value.

### Form validation

```

<html><head>
<title>Form Validation</title>
<script type="text/javascript">
</script></head><body>
<form action="/cgi-bin/test.cgi" name="myForm" onsubmit="return(validate());">
<table cellpadding="2" cellspacing="2" border="1">
<tr><td align="right">Name</td>
<td><input type="text" name="Name" /></td></tr>

```

```
<tr><td align="right">EMail</td>
<td><input type="text" name="EMail" /></td></tr>
<tr><td align="right">Zip Code</td>
<td><input type="text" name="Zip" /></td></tr>
<tr><td align="right">Country</td><td>
<select name="Country">
<option value="-1" selected>[choose yours]</option>
<option value="1">USA</option>
<option value="2">UK</option>
<option value="3">INDIA</option>
</select></td></tr>
<tr><td align="right"></td>
<td><input type="submit" value="Submit" /></td>
</tr></table></form>
<script type="text/javascript">
function validate()
{ if( document.myForm.Name.value == "" )
{alert( "Please provide your name!" );
document.myForm.Name.focus(); return false; }
  if( document.myForm.EMail.value == "" )
  {alert( "Please provide your Email!" );document.myForm.EMail.focus();
  return false; }
  if( document.myForm.Zip.value == "" ||isNaN( document.myForm.Zip.value ) ||
document.myForm.Zip.value.length != 5 )
{alert( "Please provide a zip in the format #####." );
document.myForm.Zip.focus(); return false; }
  if( document.myForm.Country.value == "-1" )
  {alert( "Please provide your country!" ); return false; }
  return( true );
}</script></body></html>
```

Name	Sharanya
EMail	
Zip Code	6543292
Country	INDIA
	Submit




The validate() in the above example checks whether the user has entered all the fields of the form. In the above case, e-mail field is left blank and when submit button is clicked, the error message is displayed.

## 2.8 EVENT HANDLING IN JAVASCRIPT

JavaScript's interacts with HTML is through events that occur when the user or browser manipulates a page. When the page loads, that is an event. When the user clicks a button, that click, too, is an event. Events are a part of the Document Object Model (DOM) Level 3 and every HTML element have a certain set of events which triggers the JavaScript Code.

### onclick Event Type:

This is the most frequently used event type which occurs when a user clicks mouse left button.

<pre>&lt;HTML&gt;&lt;TITLE&gt;Example of onClick Event Handler&lt;/TITLE&gt; &lt;HEAD&gt;&lt;SCRIPT LANGUAGE="JavaScript"&gt; function valid(form){var input=0;     input=document.myform.data.value;     alert("Hello " + input + " ! Welcome..."); }&lt;/SCRIPT&gt;&lt;/HEAD&gt; &lt;BODY&gt; &lt;H3&gt; Example of onClick Event Handler &lt;/H3&gt; Click on the button after inputting your name into the text box:&lt;br&gt; &lt;form name="myform"&gt; &lt;input type="text" name="data" value=""</pre>	<p><b>Example of onClick Event Handler</b></p> <p>Click on the button after inputting your name into the text box:</p> <p>Sharanya    Click Here</p> 
--	--

## 2.40 Client Side Programming

---

```
size=10>  
<INPUT TYPE="button" VALUE="Click  
Here" onClick="valid(this.form)">  
</form></BODY></HTML>
```

### Standard Events

Event	Description
onchange	Script runs when the element changes
onsubmit	Script runs when the form is submitted
onreset	Script runs when the form is reset
onselect	Script runs when the element is selected
onblur	Script runs when the element loses focus
onfocus	Script runs when the element gets focus
onkeydown	Script runs when key is pressed
onkeypress	Script runs when key is pressed and released
onclick	Script runs when a mouse click
ondblclick	Script runs when a mouse double
onmousedown	Script runs when mouse button is pressed
onmousemove	Script runs when mouse pointer moves
onmouseout	Script runs when mouse pointer moves out of an element
onmouseover	Script runs when mouse pointer moves over an element
onmouseup	Script runs when mouse button is released
onclick	Script runs when a mouse click
Ondblclick	Script runs when a mouse double

### Advantages of JavaScript

- **Speed:** Being client-side scripting, JavaScript is very fast because any code functions can be run immediately instead of having to contact the server and wait for an answer.
- **Simplicity:** JavaScript is relatively simple to learn and implement.
- **Versatility:** JavaScript plays nicely with other languages and can be used in a huge variety of applications. JavaScript can also be used inside scripts written in other languages such as Perl and PHP.
- **Server Load:** Being client-side reduces the demand on the website server.

### Disadvantages of JavaScript

- **Security:** Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable JavaScript.
- **Reliance on End User:** JavaScript is sometimes interpreted differently by different browsers. Whereas server-side scripts will always produce the same output, client-side scripts can be a little unpredictable.

## 2.9 DHTML WITH JAVASCRIPT

- One of the most popular uses of JavaScript is DHTML (Dynamic HyperText Markup Language).
- DHTML is the combination of HTML and JavaScript. DHTML is using JavaScript to modify the CSS styles of HTML elements.
- DHTML is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic.
- The HTML document acts as a reference to the DHTML. The DHTML can change the visibility, position, contents, background colour, z-index, clipping, size of the already positioned element. New elements can also be added to the HTML document.

### Differences between HTML and DHTML

HTML	DHTML
A plain page without any styles and Scripts called as HTML.	A page with HTML, CSS, DOM and Scripts called as DHTML.
HTML sites will be slow upon client-side technologies.	DHTML sites will be fast enough upon client-side technologies.

HTML stands for only static pages. It is referred as a static HTML and static in nature.	DHTML is Dynamic HTML means HTML+JavaScript. Hence it is referred as a dynamic HTML.
--	--

### Components of DHTML

Dynamic HTML includes the following components: HTML, Cascading Style Sheets, Scripting and the Document Object Model.

- **HTML:**
  - HTML defines the structure of a Web page, using such basic elements as headings, forms, tables, paragraphs and links.
- **Cascading Style Sheets (CSS):**
  - A style sheet controls the formatting of HTML elements.
  - Style sheets are used to specify page margins, point sizes and leading.
  - Cascading Style Sheets is a method to determine precedence and to resolve conflicts when multiple styles are used.
- **Scripting:**
  - Scripting provides the mechanisms to interpret user actions and produce client-side changes to a page.
  - DHTML can communicate with several scripting languages but JavaScript is widely used.
- **Document Object Model (DOM):**
  - The DOM outlines Web page content in a way that makes it possible for HTML elements, style sheets and scripting languages to interact with each other.
  - The W3C defines the DOM as a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented stage.

### Positioning elements in DHTML

- There are two types of positioning: absolute and relative.
- **Absolute positioning** allows to place an element anywhere in relation to the page.
- **Relative positions** the element based on offset values.
- Most DHTML is done with absolutely positioned elements. Relatively positioned elements do not accept the 'clip' style and do not allow their clipping to be changed.

## JavaScript and Cascading Style Sheets (CSS)

- Cascading Style Sheets are the standard way to define the presentation of the DHTML pages, from fonts and colors to the complete layout of a page. They are much more efficient than using HTML.
- CSS files are termed “cascading” stylesheets because of two reasons: one style sheet can cascade, or have influence over, multiple pages. Similarly, many CSS files can define a single page.
- CSS is the most important feature of DHTML. The CSS has various style properties. There are 3 ways to implement css commands into the site:
  1. Use one CSS file for all pages.
  2. Integrate CSS commands into the head of each of the documents.
  3. Use the style attribute to put CSS code directly into a HTML element.

### Common style properties of CSS

1. **Position** -Specifies how the block should be positioned on the page with respect to other page elements.
  - position:absolute- Block is positioned absolutely within the browser window, relative to <BODY> block.
  - position:relative- Block is positioned relative to its parent block, if any, or else normal flow of page.
  - position:static- Block is positioned according to standard HTML layout rules.
2. **width**-Specifies the width at which the block's contents should wrap. Width may be in measured units (50px), as a percentage of the parent block's width (50%), or auto which wraps the block according to its parent's width.

**Examples:** width:50px or width:50%
3. **height**-Specifies the height of the block, measured in units (50px), percentage of the parent block's height (50%), or auto. The height of the block will be forced to the minimum necessary to display its contents.

**Examples:** height:50px or height:50%
4. **left**-Specifies the offset of the left edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the right while negative measures (-5px) are offset towards the left.

**Examples:** left:5px or left:-5px

5. **top**- Specified the offset from the top edge of the block in accordance with the position attribute. Positive measures (5px) are offset towards the bottom of the page while negative measures (-5px) are offset towards the top of the page.

**Examples:** top:10px or top:-10px

6. **clip**-Specifies a rectangular portion of the block which is visible. The syntax of this property is different for different browsers.

1. MSIE: clip:rect(top right bottom left)

**Example:**clip:rect(0px 30px 50px 0px)

2. Netscape: clip:rect(left,top,right,bottom)

**Example:**clip:rect(0,0,30,50)

7. **visibility**-Specifies whether a block should be visible. If not visible, the block will not appear on the page, although you can make it visible later using JavaScript. The possible values for this property vary between browsers.

1. MSIE

visibility:inherit- Block inherits the visibility property of its parent.

visibility:visible- Block is visible.

visibility:hidden-Block is hidden or invisible

2. Netscape:

visibility:inherit- Block inherits the visibility property of its parent.

visibility:show- Block is visible.

visibility:hide- Block is invisible.

8. **z-index**- Specifies the "**stacking order**" of blocks, should they happen to overlap other positioned blocks. A block is assigned a z-index, which is any integer. When blocks overlap, that which has the greater positive z-index appears above a block with a lower z-index. Blocks with an equal z-index value are stacked according to the order in which they appear in the source code (bottom-to-top: first block defined appears on bottom, last block defined appears on top).

**Example:** z-index:3

9. **background-color**-Specifies the background color for the block.

**Example:**background-color:green or background-color:FF8F00

10. **background-image**-Specifies a background image for the block.

**Example:** background-image:url('images/cat.jpg')



## Advantages of CSS

- Pages download faster. Less code and the pages are shorter and neater.
- The look of the site is kept consistent throughout all the pages that work off the same stylesheet.
- Updating the design and general site maintenance are made much easier, and errors caused by editing multiple HTML pages occur far less often.
- The ID field is most important, because id will be used to reference the positioned element. Browsers call these positioned elements as **layers**.

## Grouping elements in CSS:

1. `<span>`: The element `<span>` is what you could call a neutral element which does not add anything to the document itself. But with CSS, `<span>` can be used to add visual features to specific parts of text in your documents.
2. `<div>`: Whereas `<span>` is used within a block-level element as seen in the previous example, `<div>` is used to group one or more block-level elements.

## Object Referencing

- The simplest way to reference an element in a DHTML document is by using the element's id attribute. The element is represented as an object, and its various XHTML attributes become properties that can be manipulated by scripting.

## Changing text using DHTML

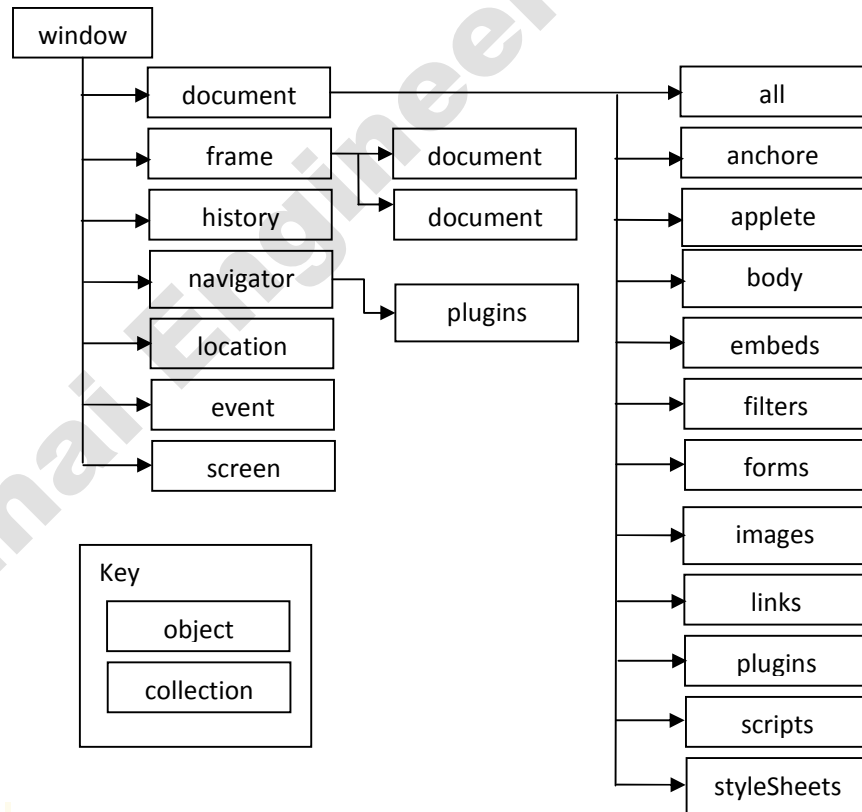
```
<html><head><title>Object Model</title>
<script type = "text/javascript">
function start()
{alert( pText.innerText );
Text.innerText = "Thanks for coming.";}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web page!</p>
</body></html>
```



- The onload calls JavaScript start function when document loading completes.Start() displays an alert box containing the value of pText.innerText.
- The object pText refers to the p element whose id is set to pText. The **innerText property** of the object refers to the text contained in that element in this example it is Welcome to our Web page!.
- The start() sets the innerText property of pText to a different value. Changing the text displayed on screen is a Dynamic HTML ability called **dynamic content**.

**Collections all and children**

- **Collections** are arrays of related objects on a page. Collections provide an easy way of referring to any specific element without an id.
- There are several special collections in the object model.The **all collection** contains all the XHTML elements in a document.
- The **length property** of the collection specifies the size of the collection. The **children** collection of a specific element contains that element’s child elements. For example, an html element has only two children—the head element and the body element.



**Figure 2.2 Object collections**

Object	Description
window	This object represents the browser window and provides access to the document object contained in the window. If the window contains frames, a separate window object is created automatically for each frame, to provide access to the document rendered in that frame. Frames are considered to be subwindows in the browser.
document	This object represents the XHTML document rendered in a window. The document object provides access to every element in the XHTML document and allows dynamic modification of the XHTML document. This object provides access to the body element of an XHTML document.
history	This object keeps track of the sites visited by the browser user. The object provides a script programmer with the ability to move forward and backward through the visited sites, but for security reasons does not allow the actual site URLs to be manipulated.
navigator	This object contains information about the Web browser, such as the name of the browser, the version of the browser etc.
location	This object contains the URL of the loaded document. When this object is set to a new URL, the browser immediately switches (navigates) to the new location.
Event	This object can be used in an event handler to obtain information about the event that occurred.
screen	The object contains information about the computer screen for the computer on which the browser is running. Information such as the width and height of the screen in pixels can be used to determine the size at which elements should be rendered in a Web page.

Collections	Descriptions
all	Many objects have an all collection that provides access to every element contained in the object. For example, the body object's all collection provides access to every element in the body element of an XHTML document.
anchors	This collection contains all anchor elements (a) that have a name or id attribute. The elements appear in the collection in the order they were defined in the XHTML document.
applets	This collection contains all the applet elements in the XHTML document. Currently, the most common applet elements are Java applets.

## 2.48 Client Side Programming

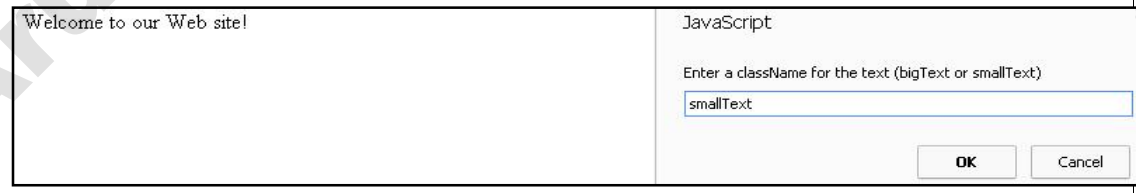
embeds	This collection contains all the embed elements in the XHTML document.
forms	This collection contains all the form elements in the XHTML document. The elements appear in the collection in the order they were defined in the XHTML document.
frames	This collection contains window objects that represent each frame in the browser window. Each frame is treated as its own subwindow.

### Dynamic Styles

An element's style can be changed dynamically. Often such a change is made in response to user events. The Dynamic HTML object model also allows to change the **class** attribute of an element—instead of changing many individual styles at a time.

### Dynamic styles

```
<html ><head><title>Object Model</title>
<style type = "text/css">
    .bigText { font-size: 3em; font-weight: bold }
    .smallText { font-size: .75em }
</style>
<script type = "text/javascript">
function start()
{
varinputClass = prompt("Enter a className for the text " + "(bigText or smallText)", "" );
pText.className = inputClass;}
</script></head>
<body onload = "start()">
<p id = "pText">Welcome to our Web site!</p>
</body></html>
```



The above example contains two classes: .bigText and .smallText. The **class** attribute applies a style class to an element. The class name always follows a period operator (.). Similar properties can be grouped into one class. The property name is followed by a colon (:) and the value of that property. Multiple properties are separated by semicolons (;).

### Dynamic Positioning

In dynamic positioning, the XHTML elements can be positioned with scripting. This is done by declaring an element's CSS position property to be either absolute or relative, and then moving the element by manipulating any of the top, left, right or bottom CSS properties.

```

<html><head><title>Dynamic Positioning</title>
<script type = "text/javascript">
var speed = 5;var count = 10; var direction = 1; varfirstLine = "Text growing";
varfontStyle = [ "serif", "sans-serif", "monospace" ];
varfontStylecount = 0;
function start()
{ window.setInterval( "run()", 100 ); }
function run()
{ count += speed;
if ( ( count % 200 ) == 0 )
{ speed *= -1;
direction = !direction;
pText.style.color =35 ( speed< 0 ) ? "red" : "blue" ;
firstLine = ( speed < 0 ) ? "Text shrinking" : "Text growing";
pText.style.fontFamily = fontStyle[ ++fontStylecount % 3 ];
}
pText.style.fontSize = count / 3;
pText.style.left = count;
pText.innerHTML = firstLine + "<br /> Font size: " + count + "px";
}
</script></head>
<body onload = "start()">
<p id = "pText" style = "position: absolute; left: 0; font-family: serif; color: blue">
Welcome!</p></body></html>

```

## 2.50 Client Side Programming



- In the above example the position of the element is varied on the page by accessing its CSS left attribute, scripting to vary the color, fontFamily and fontSize attributes, and the element's innerHTML property is used to alter the content of the element.
- This function set interval takes two parameters—a function name, and how often to run that function (in this case, every 100 milliseconds). The setTimeout() takes the same parameters but instead waits the specified amount of time before calling the named function only once.

### Frame collection

- Frames are seen as collection in DHTML. Usage of frames makes the web page more interactive.

### Cross frames

```
<html><head><title>Frames collection</title></head>
<frameset rows = "100, *">
<frame src = "frame.html" name = "upper" />
<frame src = "" name = "lower" /></frameset></html>
```

### Frame.html

```
<html ><head><title>The frames collection</title>
<script type = "text/javascript">
```

```
function start()  
{ var text = prompt( "What is your name?", "" );  
parent.frames( "lower" ).document.write( "<h1>Hello, " + text + "</h1>" );  
}}</script></head><body onload = "start()">  
<h1>Cross-frame scripting!</h1>  
</body></html>
```



### Filters and transitions

Applying filters to text and images causes changes that are persistent. **Transitions** are temporary phenomena. Applying a transition allows to transfer from one page to another with a pleasant visual effect. Filters and transitions do not add content to the pages. They just add

## 2.52 Client Side Programming

visual effects that work on some event. Filters and transitions are specified with the CSS **filter** property. Transitions give the same kind of graphics capabilities got from presentation software like Microsoft's PowerPoint. Filters are applied in the style attribute. The filter property's value is the name of the filter. Each filter has a property named `enabled`. If this property is set to `true`, the filter is applied. If it is set to `false`, the filter is not applied.

Filter	Description	Syntax
Flipv	Mirrors text vertically	<code>&lt; style = "filter: flipv "&gt;Text&lt;/style&gt;</code>
Fliph	Mirrors text horizontally	<code>&lt; style = "filter: fliph "&gt;Text&lt;/style&gt;</code>
Chroma	Applies transparency effects dynamically, without using a graphics editor to hard-code transparency into the image. This filter must be set and then enabled explicitly.	<code>chromaImg.filters( "chroma" ).color = theColor;</code> <code>chromaImg.filters( "chroma" ).enabled = true;</code>
mask	Allows to create an image mask, in which the background of an element is a solid color and the foreground of an element is transparent to the image or color behind it.	<code>&lt; style = "filter: mask( color = CCFFFF )" &gt;</code>
invert	Applies a negative image effect—dark areas become light, and light areas become dark.	<code>&lt; style = "filter: invert" &gt;</code>
Gray	The gray filter applies a grayscale image effect, in which all color is stripped from the image and all that remains is brightness data.	<code>&lt; style = "filter:gray" &gt;</code>
Xray	The xray filter applies an x-ray effect, which basically is an inversion of the grayscale effect.	<code>&lt; style = "filter:xray" &gt;</code>
Shadow	This filter creates a shadowing effect that gives the text a three-dimensional appearance. Property <code>direction</code> of the <code>shadowfilter</code> determines in which direction the shadow effect is applied and Property <code>color</code> specifies the color of the shadow that is applied to	<code>&lt;style = "filter: shadow( direction = 0, color = red ) &gt;</code>



	the text. The direction is given in angles.	
Alpha	The alpha filter also is used for transparency effects not achievable with the chroma filter. The style parameter takes value of 0 for uniform opacity, 1 for linear gradient, 2 for circular gradient and 3 for rectangular gradient. The opacity and finishopacity properties are both percentages that determine what percent opacity the specified gradient starts and finishes, respectively. Additional attributes are startX, startY, finishX and finishY specifies at what x-y coordinates the gradient starts and finishes in that element.	<code>&lt;style = "filter: alpha( style = 2, opacity = 100,finishopacity = 0 )"&gt;</code>
Glow	The glow filter adds an aura of color around text. The color and strength can both be specified as parameters.	<code>&lt; style = "filter: glow(color = red, strength = 5 )"&gt;</code>
Blur	The blur filter creates an illusion of motion by blurring text or images in a certain direction and can be applied in any of eight directions, and its strength can vary. The add property, when set to true, adds a copy of the original image over the blurred image, creating a more subtle blurring effect.	<code>&lt;style = "filter: blur( add = 0, direction = 0, strength = 0 )"&gt;</code>
Wave	The wave filter allows you to apply sine-wave distortions to text and images on your WebPages. The add property, like the blur filter, adds a copy of the text or image underneath the filtered effect. The freq and phase property determines the frequency and phase shift of the wave	<code>&lt;style = " filter: wave(add = 0, freq = 1, phase = 0,strength = 10)"&gt;</code>

## 2.54 Client Side Programming

	applied respectively. Strength is the amplitude of the sine wave that is applied.	
dropShadow	The dropShadowfilter drops shadow we applied to images.	<style=" filter: dropShadow( offx = 0, offy = 0, color = black ) ">
Light	The light filter simulates the effect of a light source shining on the page.	< style = "filter: light">

### Filters

```
<html><head><title> filters</title></head><body>
<div style="position:absolute; top:125; left:2; filter:mask">
</div><imgsrc="cat.gif" width=400 height=200 style="filter:invert">
</body></html>
```

Transitions	Description	Syntax
blendTrans	This creates a smooth fade-in/fade-out effect. The duration determines how long the transition takes.	<style=": blendTrans( duration = 3 )">
revealTrans	This allows the transition by using professional-style transitions, from box out to random dissolve. The transition parameter is the index of the element as specified in the transition array. There are 24 transitions.	<style=" filter: revealTrans( duration = 2, transition = 0 )">

### Transitions

```
<html><head><script language="text/javascript">
Function blendOut()
{ f.filters("blendTrans").apply();
f.style visibility="hidden";
f.filters("blendTrans").play();
}</script></head>
<body><div id="f" onClick="blendOut()" style="filter:blendTrans(duration=5)">
This is a nice effect</div>
</body></html>
```

### Data binding with tabular control

Before the advent of DHTML, the data manipulations were done on the server thus increasing the server load and the network load. With DHTML, these manipulations can be done directly on the client without involving the server and the network. **Data binding** is a process that allows an Internet user to manipulate Web page elements using a Web browser. It employs dynamic HTML and does not require complex scripting or programming. With data binding, data need no longer reside exclusively on the server. The data can be maintained on the client. The data storage is well distinguished from the XHTML markup on the page. In DHTML, larger amount of data will be sent to the client on the first request. Changes done to the data the client side do not propagate back to the server. To bind external data to XHTML elements, Internet Explorer employs software that is capable of connecting the browser to live data sources. These are known as **Data Source Objects (DSOs)**.

#### Tabular Data Control

The **Tabular Data Control (TDC)** is an ActiveX control that is added to a page with the object element. Data are stored in a separate file and will not be a part of the XHTML document. TDC is one way of accessing data from DSO.

- The object element will have the following properties :
  - Classid-specifies the TDC to be added to the page
  - Param tag-specifies the parameters for the object in the form of 'name-value' pairs.
- The following are the parameters taken by TDC:
  - Data URL: URL of data source.
  - UseHeader: If this value is true, then the first line of the data file has header field.
  - TextQualifier: Qualifiers are characters that are placed at both the ends of a field.
  - FieldDelim: Characters that act as separators between different data fields.
- **Record set** is the set of data from the data source. The following are the methods to access the record set:

Methods	Description
Move Next()	Moves the recordset to the next row in the data source.
Move First()	Moves to the first recordset in the file.
MoveLast()	Moves to the last recordset in the file.
Move Previous()	Moves to the previous recordset in the file.

**Tabular Data Control**

```
<html><head><title>Dynamic Recordset Viewing</title>
<object id = "Colors" classid = "CLSID:333C7BC4-460F-11D0-BC04-
0080C7055A83">
<param name = "DataURL" value = "HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" /></object>
<script type = "text/javascript">
varrecordSet = Colors.recordset;
function update()
{ h1Title.style.color = colorRGB.innerText; }
function move( whereTo )
{ switch ( whereTo )
{
case "first": recordSet.MoveFirst(); update(); break;
case "previous":recordSet.MovePrevious();
if ( recordSet.BOF )
recordSet.MoveLast(); update(); break;
case "next": recordSet.MoveNext();
if ( recordSet.EOF )
recordSet.MoveFirst(); update(); break;
case "last": recordSet.MoveLast();update(); break;
} }</script>
<style type = "text/css">
input { background-color: khaki; color: green; font-weight: bold }
</style></head>
<body style = "background-color: darkkhaki">
<h1 style = "color: black" id = "h1Title"> XHTML Color Table</h1>
<span style = "position: absolute; left: 200; width: 270; border-style: groove; text-align:
center;
background-color: cornsilk; padding: 10">
```

```

<strong>Color Name: </strong>
<span id = "colorName" style = "font-family: monospace"
datasrc = "#Colors" datafld = "ColorName">ABC</span><br />
<strong>Color RGB Value: </strong>
<span id = "colorRGB" style = "font-family: monospace"
datasrc = "#Colors" datafld = "ColorHexRGBValue">ABC
</span><br />
<input type = "button" value = "First" onclick = "move( 'first' );" />
<input type = "button" value = "Previous" onclick = "move( 'previous' );" />
<input type = "button" value = "Next" onclick = "move( 'next' );" />
<input type = "button" value = "Last" onclick = "move( 'last' );" />
</span></body></html>

```



Create a data source named HTMLStandardColors.txt that contains colors and its color values.

### Binding to an image

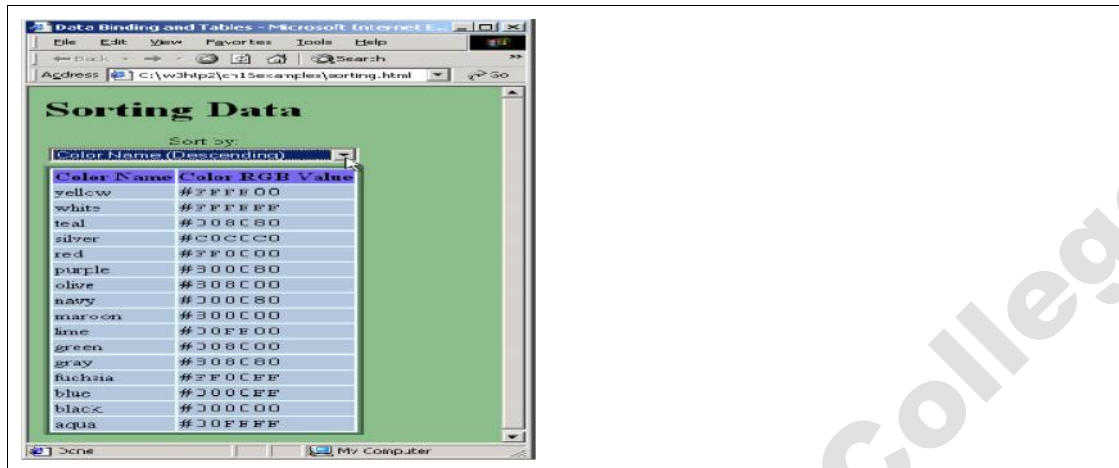
Many different types of XHTML elements for instance, image can be bound to data sources. Images are binded with data set by modifying the src attribute of the image.

```
<imgdatasrc="cat.jpeg" datafld="image">
```

The previous example, changes the color based on the recordset. Now by changing the datasrc and datafld attribute as above will make the recordset to traverse through various images (create an image data source). Omit the update();

**Binding to a table and sorting the table data:** Tables could also be binded to the data source and sorted.

```
<html ><head><title>Data Binding and Tables</title>
<object id="Colors" classid="CLSID:333C7BC4-460F-11D0-BC04-0080C7055A83">
<param name = "DataURL" value = "HTMLStandardColors.txt" />
<param name = "UseHeader" value = "TRUE" />
<param name = "TextQualifier" value = "@" />
<param name = "FieldDelim" value = "|" />
</object></head>
<body style = "background-color: darkseagreen">
<h1>Sorting Data</h1>
<table datasrc="#Colors" style="border-style: ridge; border-color: darkseagreen;
background-color: lightcyan">
<caption> Sort by: <select onchange = "Colors.Sort = this.value; Colors.Reset();">
<option value = "ColorName">Color Name (Ascending) </option>
<option value = "-ColorName">Color Name (Descending) </option>
<option value = "ColorHexRGBValue">Color RGB Value (Ascending)</option>
<option value = "-ColorHexRGBValue">Color RGB Value (Descending)</option>
</select></caption>
<thead><tr style = "background-color: mediumslateblue">
<th>Color Name</th><th>Color RGB Value</th></tr></thead>
<tbody><tr style = "background-color: lightsteelblue">
<td><span datafld = "ColorName"></span></td>
<td><span datafld = "ColorHexRGBValue" style = "font-family:
monospace"></span></td>
</tr></tbody></table></body></html>
```



Specify the column by which to sort in the Sort property of the TDC. This example sets property Sort to the value of the selected option tag (this.value) when the onchange event is fired. By default, a column is sorted in ascending order. To sort in descending order, the column name is preceded with a minus sign (-).

### Data binding elements

The following elements allow data binding:

Element	Bindable attribute
A	Href
Frame	Href
Iframe	Href
Img	Src
Div	Contained text
Input type="button"	value
Input type="checkbox"	Checked
Input type="hidden"	value
Input type="password"	Value
Input type="radio"	Checked
Input type="text"	Value
Marquee	Contained text

Param	Value
Select	select option
Span	Contained text
Table	Contained elements
Textarea	Contained text

### Structures graphics and ActiveX controls

The Structured Graphics Control is an ActiveX control that can be add to the page with an object element. It is easily accessible through scripting.

*The Structured Graphics Control is a web interface that is used for visual presentations. The Structured Graphics control facilitates the creation of simple shapes by using functions that can be called via scripting or through param tags inside object elements.*

The name attribute of the param tag method determines the order in which the function specified in the value attribute is called. The distortion of shapes like translation, rotation can also be done. To provide interaction with the user, the Structured Graphics Control can process the Dynamic HTML mouse events onmouseup, onmousedown, onmousemove, onmouseover, onmouseout, onclick and ondblclick. By default, the Structured Graphics Control does not capture mouse events, because doing so takes a small amount of processing power. The Structure Graphics Control allows you to keep a set of method calls in a separate source file and to invoke those methods by calling the SourceURL function. The following are the functions available for Structured Graphics Control:

Function	Description
SetLineColor(Rvalue, Gvalue, Bvalue)	sets the color of lines and borders of shapes that are drawn. It takes an RGB triplet in decimal notation as its three parameters.
SetLineStyle(line style, line width)	Draws a line. A value of 1 for line style creates a solid line (the default). A value of 0 does not draw any lines or borders, and a value of 2 creates a dashed line.
SetFillColor(color)	Sets the foreground color with which to fill shapes.
SetFillStyle(color style)	Sets the style in which a shape is filled with color; a value of 1 fills shapes with the solid color declared with the SetFillColor method.
Oval(x, y, height, width, direction)	Places the oval in specifies x-y location. The last parameter specifies the clockwise rotation of the oval relative to the x-axis, expressed in degrees.



Arc(x,y,height,width,starting angle, size, rotation).	Draws an arc with the given parameters.
Pie(x,y,height,width,starting angle, size, rotation)	It fills in the arc with the foreground color, thus creating a pie shape.
Polygon(no of vertices, (x,y) coordinates of the sides of the polygon)	Constructs a polygon. If the no of vertices of the polygon is 3, then 3 pairs of (x,y) co-ordinates must be specified.
Rect(x, y, height, weight, rotation)	Constructs a rectangle.
RoundRect(x, y, height, weight, rotation, height of rounded arc, width of rounded arc )	Constructs a rounded rectangle.
SetFont()	Sets the font style to use when placing text with the Text method.
PolyLine(no of points in the line, x, y of other vertex)	Draws a line with multiple segments
SetTextureFill(x, y, location of texture, value)	Fills a shape with a texture. A last parameter of 0 specifies that the texture should be stretched to fit inside the shape. A last parameter of 1 would instead tile the texture as many times as necessary inside the shape.
Translate(x, y, z)	Moves a shape in coordinate space without deforming it. Its three parameters determine the relative distance to move along the x-, y- and z-axes, respectively.
Rotate(x, y, z)	Rotates shapes in three-dimensional space. The three parameters of the Rotate function specify rotation in the x-, y- and z-coordinate planes, respectively.
MouseEventsEnabled()	Turn event capturing on

### Path, Sequencer and Sprite ActiveX Controls

The DirectAnimation Path Control allows to control the positions of elements on the page. The Path Control, the Sequencer Control and the Sprite Control allows a Web page designer to add certain multimedia effects to Web pages. This mechanism is more advanced than dynamic CSS positioning, because it allows to define paths that the targeted elements follow. This capacity to define paths gives the ability to create professional presentations, especially when integrated with other Dynamic HTML features such as filters and transitions.

## 2.62 Client Side Programming

---

Setting `AutoStart` attribute of the object to a nonzero value starts the element along a path as soon as the page loads. Setting a zero value prevents it from starting automatically, in which case a script would have to call the `Play` method to start the path.

- The **Path Control** also allows setting paths for multiple objects present on your page. To set paths for multiple objects, add a separate object tag for each object.
- The z-index of elements that overlap is determined by their order of declaration in the XHTML source (elements declared later in the XHTML file are displayed above elements declared earlier).
- A useful feature of the Path Control is the ability to execute certain actions at any point along an object's path. This capability is implemented with the `AddTimeMarker` method, which creates a time marker that can be handled with simple JavaScript event handling.
- The **Sequencer Control** provides a simpler interface for calling functions or performing actions at time intervals. The `oninit` event fires when the Sequencer Control has loaded.
- The Item object of the Sequencer Control creates a grouping of events using a common name. The **Sprite Control** allows the displaying animated images composed of individual frames.
- The object tag inserts the Sprite Control. The height and width CSS properties are needed to display the image correctly; they should be equal to the size of one frame in the file. Setting attribute
- Repeat to a nonzero value loops the animation indefinitely. **NumFrames** specifies how many frames are present in the animation source image.
- **Attributes NumFramesAcross** and **Num-FramesDown** specify how many rows and columns of frames there are in the animation file, respectively. **SourceURL** gives a path to the file containing the frames of the animation.
- The animated GIFs are most popular animation formats and are composed frames in the GIF image format. GIF images must be inserted into animated GIF files by using graphics applications such as Adobe PhotoShop elements.

Method	Description
<code>Repeat()</code>	Determines how many times the path will be traversed; setting the value to -1 specifies that the path should loop continuously
<code>Duration()</code>	Specifies the amount of time that it takes to traverse the path, in seconds.
<code>Bounce()</code>	When set to 1, reverses the element's direction on the path when it reaches the end. Setting the value to 0 returns the element to the beginning of the path when the path has been traversed.

PolyLine()	Creates a path with multiple line segments.
Target()	Specifies the id of the element that is targeted by the Path Control. Setting the CSS attribute position to absolute allows the Path Control to move an element around the screen. Otherwise, the element would be static, locked in the position determined by the browser when the page loads.
AddTimeMarker()	The first parameter determines the point at which our time marker is placed along the path, specified in seconds; when this point is reached, the onmarker event is fired. The second parameter gives an identifyingname to the event, which is later passed on to the event handler for the onmarker event. The last parameter specifies whether to fire the onmarker event every time the object's path loops pastthe time marker (value=0) or to fire the event just the first time that the time marker is passed (value= 1).
At(waiting time, action)	This method takes two parameters: How many seconds to wait, and what action to perform when that period of time has expired.
Play()	Starts the targeted element along the path.
Sprite Control()	Controls the rate at which frames are displayed
MouseEventsEnabled()	Enables or disables mouse events.
Stop()	Stops the animation in place

**Advantages of Activex Controls:**Platform Control, Active X Scripting and Easy To Use and Easy to Find

**Disadvantages of Activex Controls:**Requires User to Download Something, System Vulnerabilities, Built-in Malware and Spyware and Only Compatible with Microsoft Programs

## 2.10 JAVASCRIPT OBJECT NOTATION (JSON)

JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.

### JSON Syntax

JSON defines only two data structures: objects and arrays. An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null.

- Objects are enclosed in braces ({}), their name-value pairs are separated by a comma (,), and the name and value in a pair are separated by a colon (:). Names in an object are strings, whereas values may be of any of the seven value types, including another object or an array.
- Arrays are enclosed in brackets ([]), and their values are separated by a comma (,). Each value in an array may be of a different type, including another array or an object. When objects and arrays contain other objects or arrays, the data has a tree-like structure.

### Uses of JSON

JSON is often used as a common format to serialize and deserialize data in applications that communicate with each other over the Internet. These applications are created using different programming languages and run in very different environments. JSON is suited to this scenario because it is an open standard, it is easy to read and write, and it is more compact than other representations.

### JSON is built on two structures:

A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array. An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence

### JSON Syntax

JSON syntax is considered as a subset of JavaScript syntax:

- Data is in name/value pairs: JSON data is written as name/value pairs. A name/value pair consists of a field name (in double quotes), followed by a colon.

**Example:** "name": "Cat"

- Data is separated by commas: **Example:** "first\_name" : "Sun", "last\_name" : "moon",
- Curly braces hold objects. Square brackets hold arrays. JSON keys are on the left side of the colon. They need to be wrapped in double quotation marks,

### JSON Values:

In JSON, values must be one of the following data types: string, number, object (JSON object), array, Boolean and null

```
{ "firstName": "J", "lastName": "S", "age": 25, "children": [], "spouse": null,
  "address": {"street": "7504 TVS nagar", "city": "Tambaram", "state":
  "Tamilnadu", "postalCode": "603203" },
  "phoneNumbers": [ {"type": "mobile", "number": "212 555-3346"}],
```

```
{ "type": "fax", "number": "646 555-4567" } ]}
```

The first two name value pairs maps a string to another string. The third name value pair maps a string age with a number 25. The fourth pair maps a string children with an empty array []. The fifth pair maps a string spouse with null value. The sixth pair maps a string address with another JSON object. The seventh pair maps a string with array of JSON objects.

### Function Files

There is no native support for defining functions in JSON. Commonly used approach is to define function as string and use `eval()` or `new Function()` to construct the function. The basic difference between these two are:

- `eval()` works within the current execution scope. It can access or modify local variables.
- `new Function()` runs in a separate scope. It cannot access or modify local variables.

### HTTP Requests

JSON is most commonly used in asynchronous HTTP requests. This is where an application pulls data from another application via an HTTP request on the web. `XMLHttpRequest` is an API that provides scripted client functionality for transferring data between a client and a server. It enables to get data from an external URL without having to refresh the page. For example, a user could click a button that results in a small part of the page updating, rather than the whole page.

#### Artist.txt

```
{ "artists" : [
  { "artistname" : "Leonard Cohen", "born" : "1934" },
  { "artistname" : "Joe Satriani", "born" : "1956" },
  { "artistname" : "Snoop Dogg", "born" : "1971" } ]}
```

Below is a sample HTML page that retrieves that JSON data via HTTP, and uses JavaScript to wrap it in HTML tags and output it to the HTML document.

```
<!doctype html>
<title>Example</title><script>
// Store XMLHttpRequest and the JSON file location in variables
varxhr = new XMLHttpRequest();
varurl = "https://www.abc.com/json/Artists.txt";
```

```

// Called whenever the readyState attribute changes
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 && xhr.status == 200) { // Check if fetch request is done
    var jsonData = JSON.parse(xhr.responseText); // Parse the JSON string
    showArtists(jsonData); // Call the showArtists(), passing in the parsed JSON string
  }
};
// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();
// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
  var output = "<ul>"; // Open list
  var i; // Loop through the artists, and add them as list items
  for (var i in data.artists) {
    output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born +
    ")</li>"; }
  output += "</ul>"; // Close list. Output the data to the "artistlist" element
  document.getElementById("artistList").innerHTML = output;
}
</script><!-- The output appears here -->
<div id="artistList"></div>

```

## JSON-SQL

JSON functions in SQL Server enable to analyze and query JSON data, transform JSON to relational format, and export SQL query results as JSON text.

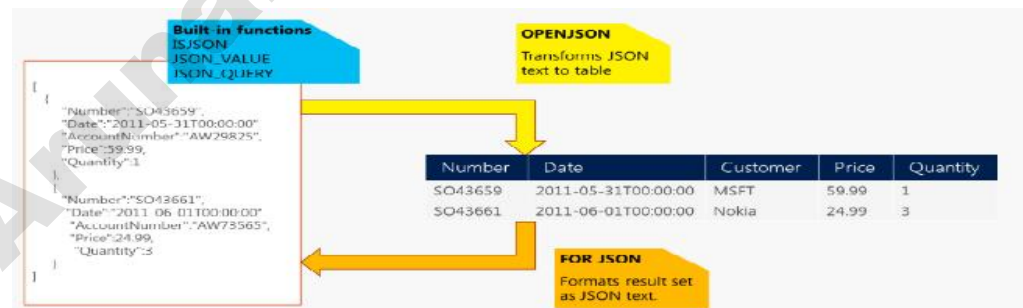


Fig 2.5: JSON with SQL

JSON text can be extracted from JSON or verify that JSON is properly formatted using built-in functions `JSON_VALUE`, `JSON_QUERY`, and `ISJSON`. For more advanced querying and analysis, the `OPENJSON` function can transform an array of JSON objects into a set of rows. Any SQL query can be executed on the returned result set. Finally, there is the `FOR JSON` clause that enables to format query results as JSON text.

**Transact-SQL code, we will define a text variable to put JSON text:**

```
DECLARE @json NVARCHAR(4000)
SET @json = N'{ "info":{ "type":1,
  "address":{ "town":"Bristol","county":"Avon", "country":"England" },
  "tags":["Sport", "Water polo"] },
  "type":"Basic"}
```

Extract values and objects from JSON text using the `JSON_VALUE` and `JSON_QUERY` functions:

```
SELECT
  JSON_VALUE(@json, '$.type') as type,
  JSON_VALUE(@json, '$.info.address.town') as town,
  JSON_QUERY(@json, '$.info.tags') as tags
```

This query will return "Basic", "Bristol", and ["Sport", "Water polo"] values. The `JSON_VALUE` function returns one scalar value from JSON text (e.g. strings, numbers, true/false) that is placed on a JSON path specified as the second parameter. `JSON_QUERY` returns an object or array on the JSON path. JSON built-in functions use JavaScript-like syntax to reference values and objects in JSON text via second parameter. The `OPENJSON` function enables to reference some array in JSON text and return elements from that array:

```
SELECT valueFROM OPENJSON(@json, '$.info.tags')
```

The string values from the tags array are returned. However, the `OPENJSON` function can return any complex object. Finally, there is a `FOR JSON` clause that can format any result set returned by SQL query as JSON text:

```
SELECT object_id, nameFROM sys.tablesFOR JSON PATH
```

## REVIEW QUESTIONS

### PART-A

#### 1. Define scripting languages.

A scripting language is a programming language that supports scripts and programs that are written for a special run-time environment. They are interpreted rather than compiled.

#### 2. List the advantages and disadvantages of server side scripting languages.

Advantages of Server side scripting Languages

These languages support high customization of the response based on the user's requirements.

Disadvantages of Server side scripting Languages

They impart more load to the web server. They can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

#### 3. List the advantages and disadvantages of client side scripting languages.

Advantages of Client side scripting Languages

- No load on the server since all the processing is done in the browser.
- These languages are easier than server side scripting languages.

Disadvantages of Client side scripting Languages

- Minimal customization of web pages.

#### 4. Differentiate between server side and client side scripting languages

Server side Scripting Languages	Client side Scripting Languages
The scripting code is run at the back end (i.e.) at the web server.	The scripting code is run in the back end (i.e) in the end user's browser.
They are less interactive.	They are more interactive.
Any change by these languages will be reflected on the database.	The changes are done only at the client side, so the database will not get affected.
More overhead on the server.	The overhead is on the local browser.



The server side scripts are not visible to the user.	The client side scripts are visible to the user.
They allow a level of privacy and personalization.	The security features are less efficient.

**5. List the advantages and disadvantages of scripting languages.**

Advantages of scripting languages:

- Any errors in the scripting language will terminate the execution of the source code.They have a simple syntax.They are easy to learn and use.
- This does not require programming expertise.It allows complex tasks to be performed in relatively few steps.It allows simple creation and editing. It could be done in a variety of text editors
- It facilitates the addition of dynamic and interactive activities to web pages. They are portable across various hardware and network platforms and scripts can be embedded in standard text document also.
- Instantaneous error reporting and error correction.The debugging process is also easy.

Disadvantages of scripting languages

- Dubious web sites or unauthorized programs are easily accessed without the user’s knowledge because the executable code is run on the end user’s browser.
- The above action may harm the end user’s system.

**6. Differentiate between programming languages and scripting languages**

<b>Programming Languages</b>	<b>Scripting Languages</b>
They are compiled.	They are interpreted.
They cannot be run directly without compilation.	They can be directly run. No explicit compilation is needed.
They have complete syntax and semantic rules.	They are unstructured subset of programming language.
They are used to build applications.	They are used to control the behavior of an application.

**7. What is javascript?**

JavaScript is a client side scripting language developed by Netscape for use within HTML web pages. JavaScript is loosely based on Java and it is built into all the major modern browsers like Internet Explorer, Firefox, Chrome, Safari etc.

**8. What do you mean by inline javascript?**

In Inline JavaScript, the scripts can be placed anywhere on the page. The output of a page will appear where the script block is in the HTML file. For instance if the JavaScript blocks are placed in the header region of the HTML document, then the dynamic content will appear in the header part of the web page and if the script blocks are at the body region of the HTML document, then the dynamic content will appear in the body part of the web page.

It is a good practice to place the scripts at the bottom of the HTML document. The reason is that each time the browser encounters a <script> tag it has to pause, compile the script, execute the script, then continue on generating the page. This takes time.

**9. What is External JavaScript?**

External JavaScript allows the reuse of same block of code on several different web pages. The JavaScript code will be written on a separate page and the web pages can make use of this code by including the page in the src attribute of the script tag.

The biggest advantage to have an external JavaScript file is that once the file has been loaded, the script will remain in the browser's cache area. So the next time the page will be loaded from the browser's cache instead of having to reload it over the Internet. This enables faster execution.

**10. Differentiate between inline and external javascript**

<b>Inline JavaScript</b>	<b>External JavaScript</b>
The JavaScript code will be embedded in the same html document.	The JavaScript code will be included in the src attribute of the <script> in the html document. The JavaScript code will not be a part of the html document.
Difficult to maintain and slow.	Easy to maintain and faster execution since the external file is stored in browser's cache.

**11. Give the Special Keywords in javascript.**

JavaScript has a few pre-defined variables with special or fixed meaning. The following are those special keywords:

- NaN (Not a Number)-This is generated when an arithmetic operation returns an invalid result.
- Infinity is a keyword which is returned when an arithmetic operation overflows JavaScript's precision which is in the order of 300 digits.
- Null is a reserved word that means "empty". In boolean operations null evaluates as false. JavaScript supports true and false as boolean values.
- Undefined value-If a variable has not been declared or assigned yet then that variable will be given a special undefined value. In boolean operations undefined evaluates as false.

### **12. Write about Alert dialog box.**

An alert dialog box is used to give a warning message to the users. It pops up a message box displaying some contents with a OK button. JavaScript alerts are used in the following situations:

- To see a message before doing anything on the website.
- To warn the user about something.
- It can be used as an error indication

### **13. What is the use of Confirmation Dialog Box?**

A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method confirm() will return true. If the user clicks on the Cancel button confirm() returns false.

### **14. What is Prompt Dialog Box?**

The prompt dialog box is very useful when a pop-up text box to used to get user input. Thus it enables to interact with the user. The user needs to fill in the field and then click OK. This dialog box is displayed using a method called prompt() which takes two parameters:

- (i) A label which you want to display in the text box
- (ii) A default string to display in the text box.

This dialog box with two buttons: OK and Cancel. If the user clicks on OK button the window method prompt() will return entered value from the text box. If the user clicks on the Cancel button the window method prompt() returns null.

### **15. Define DOM.**

DOM defines the logical structure of documents and the way a document is accessed and manipulated. It specifies how the relationships among objects used in the web page must be implemented.

### **16. What is the significance of DOM?**

The Document Object Model identifies:

- the interfaces and objects used to represent and manipulate a document
- the behavior and attributes of the interfaces and objects
- the relationships and collaborations among these interfaces and objects

### **17. What is innerHTML Property?**

The easiest way to get the content of an element is by using the innerHTML property. The inner HTML property is useful for getting or replacing the content of HTML elements. Each HTML element has an inner HTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag.

### **18. List the properties to change HTML elements.**

- element.innerHTML-Changes the inner HTML of an element
- element.attribute-Changes the attribute of an HTML element
- element.setAttribute(attribute,value)-Changes the attribute of an HTML element
- element.style.property-Changes the style of an HTML element

### **19. Define object properties.**

Object properties can be any of the primitive data types or abstract data types. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that can be used throughout the page.

```
objectName.objectProperty = propertyValue;
```

Example: varstr = document.title;

### **20. What are Object Methods?**

The methods are functions that let the object do something or let something be done to it. A function is a standalone unit of statements and a method is attached to an object and can be referenced by the keyword. Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

Example: document.write("This is a method of the object document");

### **21. Write about User-Defined Objects.**

Apart from built-in objects, the users can also create their own objects. All user-defined objects and built-in objects are descendants of an object called Object. The new operator is used to create a new object.

**22. Write about new Operator.**

The new operator is used to create an instance of an object. To create an object, the new operator is followed by the constructor method.

Example: `var books = new Array("Thirukural", "Geethai");`

In the above example `Array()` is a built-in object. `Books` is the instance of the object `Array()`.

**23. Give the importance of Object() Constructor**

A constructor is a function that creates and initializes an object. The `Object()` constructor is used to build an object. The return value of the `Object()` constructor is assigned to a variable. The variable contains a reference to the new object. The properties assigned to the object are not variables. These properties can be accessed only through objects.

`Objectname.property`

**24. What are patterns and attributes?**

- pattern: A string that specifies the pattern of the regular expression or another regular expression.
- attributes: An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

**25. What is DHTML?**

- One of the most popular uses of JavaScript is DHTML (Dynamic HyperText Markup Language).
- DHTML is the combination of HTML and JavaScript. DHTML is using JavaScript to modify the CSS styles of HTML elements.
- DHTML is the combination of several built-in browser features in fourth generation browsers that enable a web page to be more dynamic.
- The HTML document acts as a reference to the DHTML. The DHTML can change the visibility, position, contents, background colour, z-index, clipping, size of the already positioned element. New elements can also be added to the HTML document.

**26. Differentiate between HTML and DHTML**

HTML	DHTML
A plain page without any styles and Scripts called as HTML.	A page with HTML, CSS, DOM and Scripts called as DHTML.

HTML sites will be slow upon client-side technologies.	DHTML sites will be fast enough upon client-side technologies.
HTML stands for only static pages. It is referred as a static HTML and static in nature.	DHTML is Dynamic HTML means HTML+JavaScript. Hence it is referred as a dynamic HTML.

### 27. List the components of DHTML

Dynamic HTML includes the following components: HTML, Cascading Style Sheets, Scripting and the Document Object Model.

- HTML:
  - HTML defines the structure of a Web page, using such basic elements as headings, forms, tables, paragraphs and links.
- Cascading Style Sheets (CSS):
  - A style sheet controls the formatting of HTML elements.
  - Style sheets are used to specify page margins, point sizes and leading.
  - Cascading Style Sheets is a method to determine precedence and to resolve conflicts when multiple styles are used.
- Scripting:
  - Scripting provides the mechanisms to interpret user actions and produce client-side changes to a page.
  - DHTML can communicate with several scripting languages but JavaScript is widely used.
- Document Object Model (DOM):
  - The DOM outlines Web page content in a way that makes it possible for HTML elements, style sheets and scripting languages to interact with each other.
  - The W3C defines the DOM as a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented stage.

### 28. How to Position the elements in DHTML?

- There are two types of positioning: absolute and relative.
- Absolute positioning allows to place an element anywhere in relation to the page.

- Relative positions the element based on offset values.
- Most DHTML is done with absolutely positioned elements. Relatively positioned elements do not accept the 'clip' style and do not allow their clipping to be changed.

### **29. List the advantages of CSS**

- Pages download faster. Less code and the pages are shorter and neater.
- The look of the site is kept consistent throughout all the pages that work off the same stylesheet.
- Updating the design and general site maintenance are made much easier, and errors caused by editing multiple HTML pages occur far less often.
- The ID field is most important, because id will be used to reference the positioned element. Browsers call these positioned elements as layers.

### **30. What is Object Referencing?**

The simplest way to reference an element in a DHTML document is by using the element's id attribute. The element is represented as an object, and its various XHTML attributes become properties that can be manipulated by scripting

### **31. Define collections.**

- Collections are arrays of related objects on a page. Collections provide an easy way of referring to any specific element without an id.
- There are several special collections in the object model. The all collection contains all the XHTML elements in a document.

### **32. Write about Filters and transitions**

Applying filters to text and images causes changes that are persistent. Transitions are temporary phenomena. Applying a transition allows to transfer from one page to another with a pleasant visual effect. Filters and transitions do not add content to the pages. They just add visual effects that work on some event. Filters and transitions are specified with the CSS filter property. Transitions give the same kind of graphics capabilities got from presentation software like Microsoft's PowerPoint. Filters are applied in the style attribute. The filter property's value is the name of the filter. Each filter has a property named enabled. If this property is set to true, the filter is applied. If it is set to false, the filter is not applied.

### **33. Define Data binding.**

It is a process that allows an Internet user to manipulate Web page elements using a Web browser. It employs dynamic HTML and does not require complex scripting or programming. With data binding, data need no longer reside exclusively on the server. The data can be maintained on the client.

**34. What is Tabular Data Control (TDC)?**

This is an ActiveX control that is added to a page with the object element. Data are stored in a separate file and will not be a part of the XHTML document. TDC is one way of accessing data from DSO.

**35. Define structured graphics.**

The Structured Graphics Control is a web interface that is used for visual presentations. The Structured Graphics control facilitates the creation of simple shapes by using functions that can be called via scripting or through param tags inside object elements.

**36. Define path control.**

The Path Control also allows setting paths for multiple objects present on your page. To set paths for multiple objects, add a separate object tag for each object.

**37. Define sequencer control.**

The Sequencer Control provides a simpler interface for calling functions or performing actions at time intervals. The oninit event fires when the Sequencer Control has loaded.

**38. What is sprite control?**

The Item object of the Sequencer Control creates a grouping of events using a common name. The Sprite Control allows the displaying animated images composed of individual frames.

**39. Give the advantages and disadvantages of activex controls.**

Advantages of Activex Controls: Platform Control, Active X Scripting and Easy To Use and Easy to Find

Disadvantages of Activex Controls: Requires User to Download Something, System Vulnerabilities, Built-in Malware and Spyware and Only Compatible with Microsoft Programs

**40. What is JSON?**

JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.

**41. What is JSON Syntax?**

JSON defines only two data structures: objects and arrays. An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null.



**42. Give the uses of JSON.**

JSON is often used as a common format to serialize and deserialize data in applications that communicate with each other over the Internet. These applications are created using different programming languages and run in very different environments. JSON is suited to this scenario because it is an open standard, it is easy to read and write, and it is more compact than other representations.

**PART-B**

1. Brief about the fundamentals of scripting languages.
2. Explain server and client side scripting languages.
3. Describe the basics of javascript.
4. Elaborate the various control structures in javascript.
5. How to write user defined functions in javascript?
6. Brief about dialog boxes in javascript.
7. Explain DOM.
8. How to remove, modify the existing HTML elements in javascript?
9. Explain the methods in number object of javascript.
10. Explain the methods in date object of javascript.
11. Explain the methods in string object of javascript.
12. Explain the methods in array object of javascript.
13. Explain the methods in math object of javascript.
14. Explain the methods in regular expressions object of javascript.
15. Describe validation in javascript.
16. Write in detail about event handling in javascript.
17. Elaborate the usage of DHTML with javascript.
18. Brief about styling properties of CSS.
19. Explain positioning of elements in DHTML.
20. Describe frame collections.
21. Write about filters and transitions.

2.78 *Client Side Programming*

---

22. Brief about tabular data control.
23. Explain structures graphics and activex control.
24. Describe JSON.

Arunai Engineering College

---

---

# 3

## SERVER SIDE PROGRAMMING

---

---

### 3.1 SERVLETS

A Java servlet is a Java program that extends the capabilities of a server. Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

*Servlets are dynamically loaded Java code running on a JVM that services the requests from a web server.*

Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

Servlets are effective for developing Web-based solutions that help provide secure access to a Web site, interact with databases on behalf of a client, dynamically generate custom XHTML documents to be displayed by browsers and maintain unique session information for each client.

#### **CGI (Common Gateway Interface)**

The common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. CGI does not communicate directly with the browser. The browser communicates with the server, the server in turn talks with the CGI program and CGI answers to the server which is rendered to the browser.

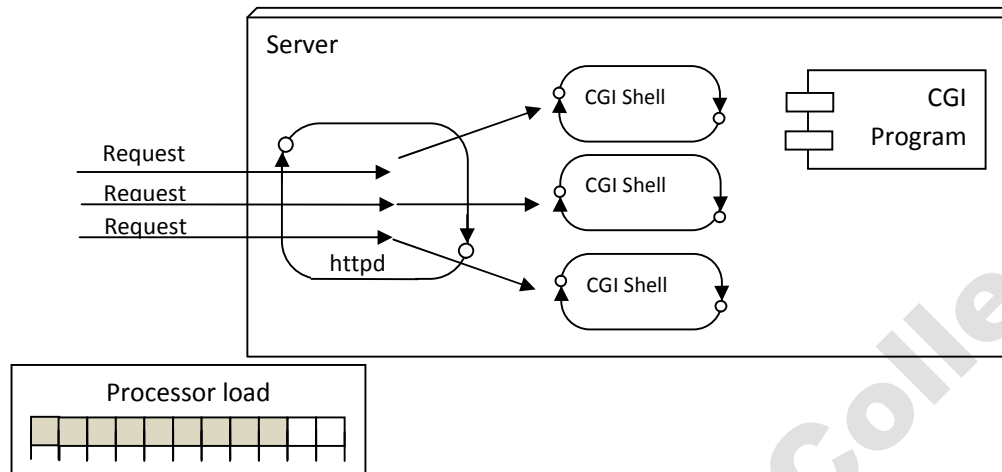


Figure 3.1 CGI

### Advantages of Servlets over CGI

There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the processes such as they share a common memory area, light weight, cost of communication between the threads are low. The basic benefits of servlet are as follows:

- Better performance: because it creates a thread for each request not process.
- Portability: because it uses java language.
- Robust: Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
- Secure: because it uses java language.

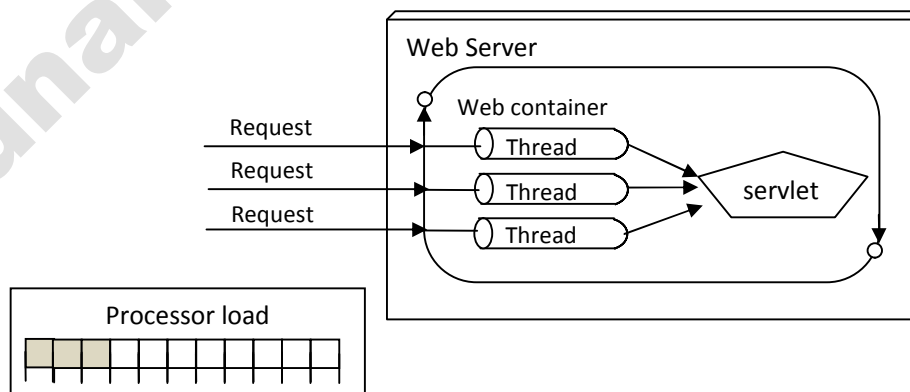


Figure 3.2 Servlets

## Capabilities of Servlet

Servlet is a technology that is used to create web application. Servlet is an API that provides many interfaces and classes including documentations. Servlet is an interface that must be implemented for creating any servlet. Servlet is a class that extends the capabilities of the servers and responds to the incoming request. It can respond to any type of requests. Servlet is a web component that is deployed on the server to create dynamic web page.

## Differences between process and threads

Process	Threads
Process run in separate address space	Threads of a process share address space.
Process have their own copy of data segment of parent process.	Threads have direct access to data segment of its process
Processes must use inter-process communication to communicate with sibling processes	Threads can directly communicate with other threads of that process.
Process carry more state information	Threads carry less state information
New process require duplication of parent process, and allocation of memory and resources for it are costly	New threads are easily created.

## Servlet Architecture

Servlets are used when a small portion of the content sent to the client is static text or in the form of markup languages. They perform a task on behalf of the client, and then invoke other servlets to provide a response. The server that executes a servlet is called as the **servlet container or servlet engine**. A client sends an HTTP request to the server or servlet container. The server or servlet container receives the request and directs it to be processed by the appropriate servlet.

The servlet does its processing, which may be interacting with a database or other server-side components. The servlet returns its results to the client. From the programming perspective, all servlets must implement the Servlet interface of the package javax.servlet. The methods of interface Servlet are invoked automatically by the servlet container. This interface defines the following five methods:

### 1. void init( ServletConfig config )

The servlet container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. The ServletConfig argument is supplied by the servlet container that executes the servlet.

## 2. ServletConfig getServletConfig()

This method returns a reference to an object that implements interface ServletConfig. This object provides access to the servlet's configuration information such as servlet initialization parameters and the servlet's ServletContext, which provides the servlet with access to its environment

## 3. String getServletInfo()

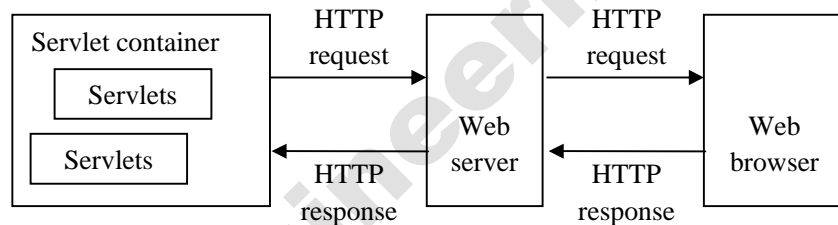
This method is defined by a servlet programmer to return a String containing servlet information such as the servlet's author and version.

## 4. void service( ServletRequest request, ServletResponse response )

The servlet container calls this method to respond to a client request to the servlet.

## 5. void destroy()

This "cleanup" method is called by the web container before removing the servlet instance from the service. Resources used by the servlet, such as an open file or an open database connection, should be deallocated here.

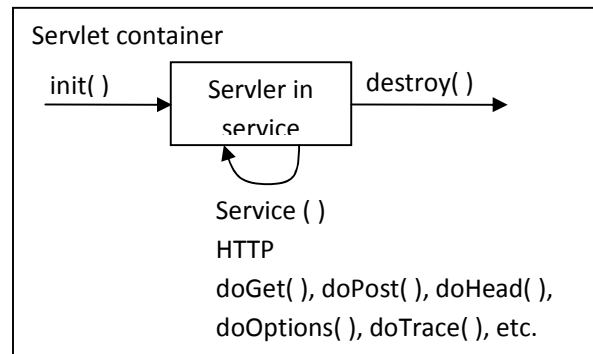


**Figure: 3.3 Servlet Architecture**

## Servlet lifecycle

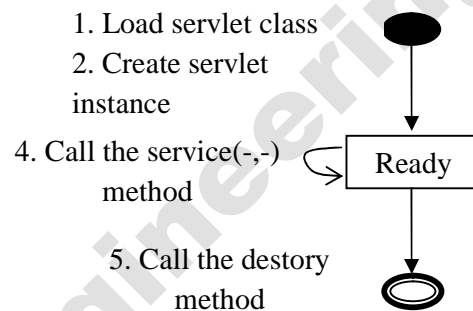
The web container maintains the life cycle of a servlet instance. The following are the phases in the life of a servlet:

- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.



**Figure 3.4 Servlet lifecycle**

There are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the `init()` method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the `destroy()` method, it shifts to the end state.



**Figure 3.5 State diagram of lifecycle of servlet**

### 1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

### 2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

### 3) `init` method is invoked

The web container calls the `init` method only once after creating the servlet instance. The `init` method is used to initialize the servlet. It is the life cycle method of the `javax.servlet.Servlet` interface.

```
public void init(ServletConfig config) throws ServletException
```

```
public void init(ServletConfig config) throws ServletException
```

#### 4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Remember that servlet is initialized only once.

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.The doGet() and doPost() are most frequently used methods with in each service request.

**Syntax:**public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException

#### 5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

```
public void destroy()
```

The servlet can be created by three ways:

1. By implementing Servlet interface
2. By inheriting GenericServlet class
3. By inheriting HttpServlet class

#### Servlet Interface

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

#### Methods in servlet interface

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



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

### Creating a servlet using servlet interface

```

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

public class First implements Servlet
{
    ServletConfig config=null;

    public void init(ServletConfig config)
    {
        this.config=config;
        System.out.println("servlet is initialized");
    }

    public void service(ServletRequest req,ServletResponse res)
        throws IOException,ServletException
    {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>This is a simple servlet</b>");
        out.print("</body></html>");
    }

    public void destroy()
    {
        System.out.println("servlet is destroyed");
    }

    public ServletConfig getServletConfig()
    {return config; }

    public String getServletInfo()
    {return "copyright 2010-2016"; } }

```

### 3.8 Server Side Programming

---

The above program creates a servlet by implementing the servlet interface. The servlet is initialized with its configuration in the `init()`. The `service()` does the real operation of the servlet. In this example, the servlet creates a web page that displays “This is a simple servlet”. The service method creates two objects in its parameter field: `req` and `res`. The `req` object is created for `ServletRequest` class. All the requests will be issues by this object. The `res` object is created for `ServletResponse` class. This is responsible for the output of the servlet.

`res.setContentType()` describes the content of the output. `PrintWriter` is the output stream with object `out`. The `print()` is called using the object of the `PrintWriter` class. The `print()` contains the HTML tags to create the web page.

The servlet is destroyed using `destroy()`. In this example, `getServletInfo ()` is optional and returns the copyrights information.

#### ServletConfig Interface

An object of `ServletConfig` is created by the web container for each servlet. This object can be used to get configuration information from `web.xml` file.

```
public ServletConfig getServletConfig(); //This returns the object of the
ServletConfig.
```

#### Methods of ServletConfig interface

Method	Description
<code>public String getInitParameter(String name)</code>	Returns the parameter value for the specified parameter name.
<code>public Enumeration getInitParameterNames()</code>	Returns an enumeration of all the initialization parameter names.
<code>public String getServletName()</code>	Returns the name of the servlet.
<code>public ServletContext getServletContext()</code>	Returns an object of <code>ServletContext</code> .

#### Servlet configuration

```
import java.io.*;import java.util.*;import javax.servlet.*;  
import javax.servlet.http.*;  
public class GetInitParameter extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```

throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("A Student have the following record : ");
    Enumeration enm = getServletConfig().getInitParameterNames();
    while (enm.hasMoreElements()) {
        out.print(enm.nextElement() + " ");
    }
    out.println("\nEnr. No. : " + getServletConfig().getInitParameter("enrNo"));
    out.println("Name : " + getServletConfig().getInitParameter("name"));
    out.println("Programme : " + getServletConfig().getInitParameter("prg"));
    out.println("Address : " + getServletConfig().getInitParameter("add"));
    out.println("Phone no : " + getServletConfig().getInitParameter("phNo"));
}

```

**web.xml**

```

<web-app><servlet><init-param>
<param-name>enrNo</param-name>
<param-value>102569638</param-value></init-param>
<init-param><param-name>name</param-name>
<param-value>Bipul</param-value></init-param>
<init-param><param-name>prg</param-name>
<param-value>MCA</param-value></init-param>
<init-param><param-name>add</param-name>
<param-value>Rohini</param-value></init-param>
<init-param><param-name>phNo</param-name>
<param-value>9013278579</param-value></init-param>
<servlet-name>GetInitParameter</servlet-name>
<servlet-class>GetInitParameter</servlet-class></servlet>
<servlet-mapping>
<servlet-name>GetInitParameter</servlet-name>
<url-pattern>/GetInitParameter</url-pattern>
</servlet-mapping>
</web-app>

```

### 3.10 Server Side Programming



A screenshot of a web browser window. The address bar shows 'localhost:8080/bipulz/GetIntParameter'. The main content area displays the following text:

```
A Student have the following record :
prg phNo name enrNo add

Enr. No.   : 102569638
Name      : Bipul
Programme : MCA
Address   : Rohini
Phone no  : 9013278579
```

#### GenericServlet class

GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method. GenericServlet class can handle any type of request so it is protocol-independent.

#### Methods of GenericServlet class

Method	Description
public void init(ServletConfig config)	To initialize the servlet.
public abstract void service(ServletRequest request, ServletResponse response)	Provides service for the incoming request. It is invoked at each time when user requests for a servlet.
public void destroy()	Invoked only once throughout the life cycle and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	Returns the object of ServletConfig.
public String getServletInfo()	Returns information about servlet such as writer, copyright, version etc.
public void init()	It is a convenient method for the servlet programmers, now there is no need to call super.init(config)
public ServletContext getServletContext()	Returns the object of ServletContext.
public String getInitParameter(String name)	Returns the parameter value for the given parameter name.
public Enumeration getInitParameterNames()	Returns all the parameters defined in the web.xml file.
public String getServletName()	Returns the name of the servlet object.

**Generic servlet**

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

public class First extends GenericServlet{

public void service(ServletRequest req,ServletResponse res) throws IOException,
ServletException{

res.setContentType("text/html");

PrintWriter out=res.getWriter();out.print("<html><body>");

out.print("<b>hello generic servlet</b>");out.print("</body></html>");}}
```

**HttpServlet Class**

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

**Methods of HttpServlet class**

Method	Description
public void service(ServletRequest req,ServletResponse res)	dispatches the request to the protected service method by converting the request and response object into http type.
protected void service(HttpServletRequest req, HttpServletResponse res)	receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
protected void doGet(HttpServletRequest req, HttpServletResponse res)	handles the GET request. It is invoked by the web container.
protected void doPost(HttpServletRequest req, HttpServletResponse res)	handles the POST request. It is invoked by the web container.
protected void doHead(HttpServletRequest req, HttpServletResponse res)	handles the HEAD request. It is invoked by the web container.
protected void doOptions(HttpServletRequest req, HttpServletResponse res)	handles the OPTIONS request. It is invoked by the web container.
protected void doPut(HttpServletRequest req, HttpServletResponse res)	handles the PUT request. It is invoked by the web container.

### 3.12 Server Side Programming

protected void doTrace(HttpServletRequest req, HttpServletResponse res)	handles the TRACE request. It is invoked by the web container.
protected void doDelete(HttpServletRequest req, HttpServletResponse res)	handles the DELETE request. It is invoked by the web container.
protected long getLastModified(HttpServletRequest req)	returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

#### HTTP servlet

```
import java.io.*;import javax.servlet.*;import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    private String message;
    public void init() throws ServletException
    {    message = "Hello World"; }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {    response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + message + "</h1>"); }
    public void destroy() { }}
```

#### ServletRequest Interface

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

#### Methods of ServletRequest interface

Method	Description
public String getParameter(String name)	Used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	Returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.

<code>java.util.Enumeration getParameterNames()</code>	Returns an enumeration of all of the request parameter names.
<code>public int getContentLength()</code>	Returns the size of the request entity data, or
<code>public String getCharacterEncoding()</code>	Returns the character set encoding for the input of this request.
<code>public String getContentType()</code>	Returns the Internet Media Type of the request entity data, or null if not known.
<code>public ServletInputStream getInputStream() throws IOException</code>	Returns an input stream for reading binary data in the request body.
<code>public String getParameter(String name)</code>	Used to obtain the value of a parameter by name.
<code>public String[] getParameterValues(String name)</code>	Returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
<code>public abstract String getServerName()</code>	Returns the host name of the server that received the request.
<code>public int getServerPort()</code>	Returns the port number on which this request was received.

### ServletRequest to display the name of the user

#### index.html

```
<form action="welcome" method="get">
Enter your name<input type="text" name="name"><br>
<input type="submit" value="login"></form>
<form action="welcome" method="get">
Enter your name<input type="text" name="name"><br>
<input type="submit" value="login"></form>
```

#### DemoServ.java

```
import javax.servlet.http.*; import javax.servlet.*; import java.io.*;
public class DemoServ extends HttpServlet{
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{ res.setContentType("text/html"); PrintWriter pw=res.getWriter();
String name=req.getParameter("name");//will return value
pw.println("Welcome "+name); pw.close(); }
```

## FORM GET AND POST ACTIONS

The browser uses two methods to pass this information to web server. These methods are GET () and POST ().

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

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

The GET method is the default method to pass information from browser to web server. GET method should be avoided while passing password or other sensitive information to the server. The GET method can hold only 1024 characters 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.

### Reading Form Data using Servlet

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

1. getParameter()-to get the value of a form parameter.
2. getParameterValues()-this method is used if the parameter appears more than once and returns multiple values(example checkbox).
3. getParameterNames()-this method is used when a complete list of all parameters is needed in the current request.

### Post()

Post() is a reliable method of passing information to a backend program. This handles the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? 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 can be parsed and processed. Servlet handles this type of requests using doPost() method.

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



```

public class ReadParams extends HttpServlet { // Method to handle GET method
request.
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");    PrintWriter out = response.getWriter();
        String title = "Reading All Form Parameters";
        out.println( "<html>\n" + "<head><title>" + title + "</title></head>\n" +
            "<body bgcolor=#f0f0f0>\n" + "<h1 align=center>" + title + "</h1>\n"
            + "<table width=100% border=1 align=center>\n" + "<tr
            bgcolor=#949494>\n" + "<th>Param Name</th><th>Param Value(s)</th>\n" +
            "</tr>\n");
        Enumeration paramNames = request.getParameterNames();
        while(paramNames.hasMoreElements())    {
            String paramName = (String)paramNames.nextElement();
            out.print("<tr><td>" + paramName + "</td>\n<td>");
            String[] paramValues = request.getParameterValues(paramName);
            if (paramValues.length == 1)    {
                String paramValue = paramValues[0];
                if (paramValue.length() == 0)    out.println("<i>No Value</i>");
                else    out.println(paramValue);    }
            else {    // Read multiple valued data
                out.println("<ul>");
                for(int i=0; i < paramValues.length; i++) {
                    out.println("<li>" + paramValues[i]);
                }    out.println("</ul>");    }    }
            out.println("</tr>\n</table>\n</body></html>");
        } // Method to handle POST method request.
        public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
        {    doGet(request, response);    }}

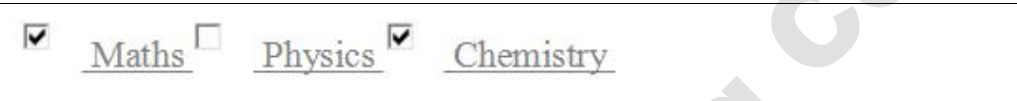
```

**Form.html**

```

<html><body>
<form action="ReadParams" method="POST" target="_blank">
<input type="checkbox" name="maths" checked="checked" /> Maths
<input type="checkbox" name="physics" /> Physics
<input type="checkbox" name="chemistry" checked="checked" /> Chem
<input type="submit" value="Select Subject" />
</form></body></html>

```


**Servlet output:**

Param Name	Param Value(s)
maths	on
Chemistry	on

In the above example, the subjects checked in form.html is retrieved in the servlet code using get() and post(). The getParameterValues() returns the values of more than one parameter and is a enumeration type. The hasMoreElements() checks whether there are more parameters to be passed and the nextElement() fetches the next parameter value.

**3.2 SESSION HANDLING**

Session simply means a particular interval of time. Session Tracking is a way to maintain state (data) of a user. It is also known as **session management** in servlet. Http is a stateless protocol that means each request is considered as the new request. So the states are maintained using various session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of a user to recognize to particular user.

**Session Tracking Techniques**

There are four techniques used in Session tracking: Cookies, Hidden Form Field, URL Rewriting and HttpSession Interface

## Cookies

A cookie is a small piece of information that persist between the multiple client requests. In HTTP each request is considered as a new request even if two requests are issued by the same user. A cookie is added with response from the servlet which serves as an identifier to the user. This is done when the first request is made. So cookie is stored in the cache of the browser. After that if any request is sent by the user, the stored cookie is added with request by default. Thus, the server recognizes the user as the old user.

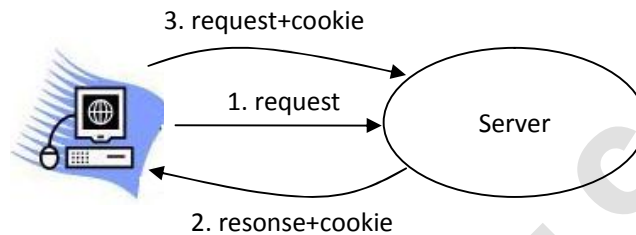


Figure 3.6 Cookies

## Differences between Cookie and Session

Session	Cookie
Sessions are <b>server-side</b> files that contain user information	Cookies are <b>client-side</b> files that contain user information
Any amount of data can be stored within sessions.	Official MAX Cookie size is 4KB
Session ends when user close his browser.	Cookie ends depends on the life time you set for it.

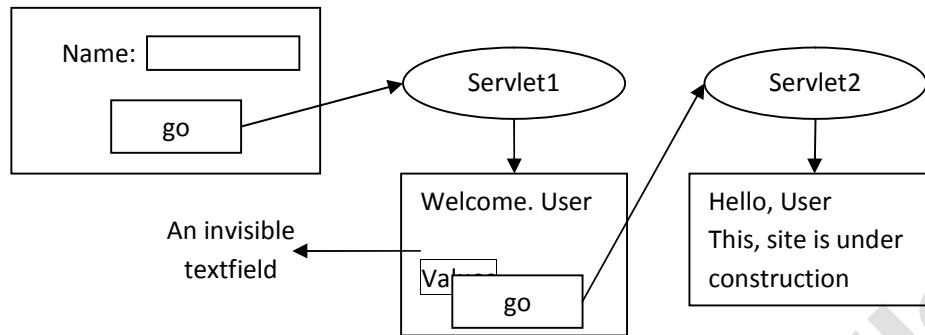
## Hidden Form field

Here, a hidden (invisible) textfield is used for maintaining the state of end user. In such case, the information is stored in the hidden field. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

```
<input type="hidden" name="Field name" value="value">
```

**Example:** `<INPUT TYPE="HIDDEN" NAME="session" VALUE="a1234">`

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data.



**Figure 3.7 Hidden Form field**

#### Advantage of Hidden Form Field

1. It will always work whether cookie is disabled or not.

#### Disadvantages of Hidden Form Field:

1. It is maintained at server side.
2. Extra form submission is required on each pages.
3. Only textual information can be used.

#### URL Rewriting

Appending the name of the user in the query string and getting the value from the query string in another page is called URL rewriting. The parameter **name-value pairs** are passed using the following format:

`url?name1=value1&name2=value2&??`

A name and a value is separated using an equal = sign, a parameter name-value pair is separated from another parameter using the ampersand(&). With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change.

Due to the limited space available in rewriting a URL, the extra information is usually limited to a unique session ID. When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a, the `getParameter()` method to obtain a parameter value.

<code>http://server:port/servlet/Rewritten</code>	Original URL
<code>http://server:port/servlet/Rewritten/123</code>	extra path information
<code>http://server:port/servlet/Rewritten?sessionid=123</code>	added parameter
<code>http://server:port/servlet/Rewritten;\$sessionid\$123</code>	custom change

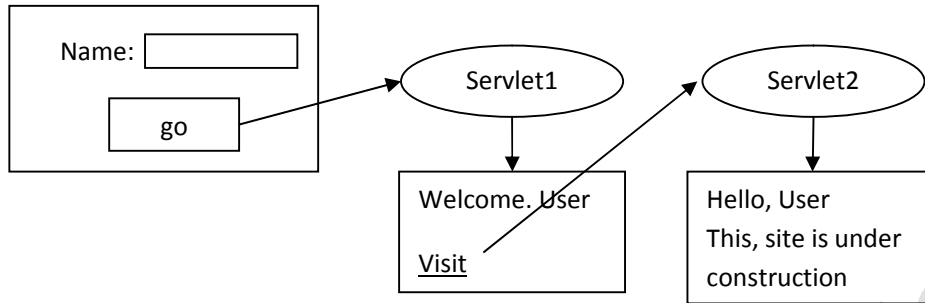


Figure 3.7(a) URL rewriting

**Advantage of URL Rewriting**

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

**Disadvantage of URL Rewriting**

1. It will work only with links.
2. It can send only textual information.

**HTTP Session Interface**

The web container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks: bind objects and view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.

The HttpSession interface provides two methods to get the object of HttpSession created by the container:

1. public HttpSession getSession()-Returns the current session associated with this request, or if the request does not have a session, creates one.
2. public HttpSession getSession(boolean create):-Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

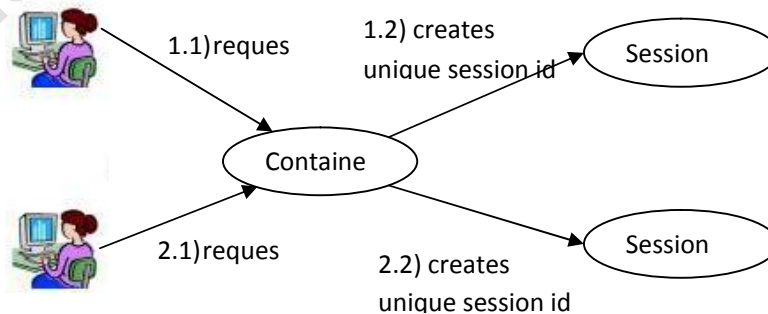


Figure 3.8 Creating session

**Methods of HttpSession interface**

Method	Description
public Object getAttribute(String name)	returns the object bound with the specified name in this session, or null if no object is bound under the name.
public Enumeration getAttributeNames()	returns an Enumeration of String objects containing the names of all the objects bound to this session.
public long getCreationTime()	returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
public String getId()	returns a string containing the unique identifier assigned to this session.
public long getLastAccessedTime()	returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
public int getMaxInactiveInterval()	returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
public void invalidate()	invalidates this session and unbinds any objects bound to it.
public boolean isNew()	returns true if the client does not yet know about the session or if the client chooses not to join the session.
public void removeAttribute(String name)	removes the object bound with the specified name from this session.
public void setAttribute(String name, Object value)	binds an object to this session, using the name specified.
public void setMaxInactiveInterval(int interval)	specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

**Session tracking**

```
import java.io.*;import javax.servlet.*;import javax.servlet.http.*;
import java.util.*;
public class SessionTrack extends HttpServlet {
```

```

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
{
    HttpSession session = request.getSession(true); //session id is created
    Date createTime = new Date(session.getCreationTime()); // Get session creation time.
    Date lastAccessTime = new Date(session.getLastAccessedTime());
    // Get last access time of this web page.
    String title = "Welcome Back to my website";
    Integer visitCount = new Integer(0);
    String visitCountKey = new String("visitCount");
    String userIDKey = new String("userID");
    String userID = new String("ABCD");
    // Check if this is new comer on your web page.
    if (session.isNew()){
        title = "Welcome to my website";
        session.setAttribute(userIDKey, userID);
    } else {
        visitCount = (Integer)session.getAttribute(visitCountKey);
        visitCount = visitCount + 1;
        userID = (String)session.getAttribute(userIDKey);
    }
    session.setAttribute(visitCountKey, visitCount);
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>\n" + "<head><title>" + title + "</title></head>\n" +
        "<body bgcolor=#f0f0f0>\n" + "<h1 align=center>" + title +
"</h1>\n" +
        "<h2 align=center>Session Infomation</h2>\n" +
"<table border=1 align=center>\n" + "<tr bgcolor=#949494>\n" +
" <th>Session info</th><th>value</th></tr>\n" + "<tr>\n" +
" <td>id</td>\n" + " <td>" + session.getId() + "</td></tr>\n" +
"<tr>\n" + " <td>Creation Time</td>\n" + " <td>" + createTime +
" </td></tr>\n" + "<tr>\n" + " <td>Time of Last Access</td>\n" +
" <td>" + lastAccessTime + " </td></tr>\n" + "<tr>\n" + " <td>User

```

### 3.22 Server Side Programming

```
ID</td>\n" +  
    " <td>" + userID + " </td></tr>\n" + "<tr>\n" + " <td>Number of  
visits</td>\n" +  
    " <td>" + visitCount + "</td></tr>\n" + "</table>\n" +  
"</body></html>");  
}}
```

Welcome to my website

Session Information

Session info	Value
id	0AE3EC93FF44E3C525B4351B77ABB2D5
Creation Time	Tue Jun 08 17:26:40 GMT+04:00 2010
Time of Last Access	Tue Jun 08 17:26:40 GMT+04:00 2010
User ID	ABCD
Number of visits	0

Now try to run the same servlet for second time, it would display following result.

Welcome Back to my website

Session Information

info type	Value
Id	0AE3EC93FF44E3C525B4351B77ABB2D5
Creation Time	Tue Jun 08 17:26:40 GMT+04:00 2010
Time of Last Access	Tue Jun 08 17:26:40 GMT+04:00 2010
User ID	ABCD
Number of visits	1



### 3.3 COOKIES

The cookie is stored on the user's machine, but it is not an executable program and cannot do anything to the stored machine.

*A cookie is a short piece of data, not code, which is sent from a web server to a web browser when that browser visits the server's site.*

Cookies are embedded in the HTML information flowing back and forth between the user's computer and the servers. Whenever a web browser requests a file from the web server it sends the request along with a cookie, the browser sends a copy of that cookie back to the server along with the request. Thus a server sends a cookie and the browser sends it back whenever it request another file from the same server. In this way, the server knows that the user have visited before and can coordinate the access to different pages on its web site.

**Example:** An Internet shopping site uses a cookie to keep track of which shopping basket belongs to a user.

#### Components of cookies

Cookies are a plain text data record of 5 variable-length fields:

- **Expiry time:** The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- **Domain:** The domain name of the site.
- **Path:** The path to the directory or web page that set the cookie.
- **Secure:** If this field contains the word "secure" then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **Name=Value:** Cookies are set and retrieved in the form of key and value pairs.

#### Uses of cookies

Identifying a user during an e-commerce session, Avoiding username and password, Customizing a site and Focusing advertising

#### Types of cookies

There are two types of cookies: Session Cookies and persistent cookies

##### Session Cookies

Session cookies are stored in memory during the applet or application session. Session cookies expire when the applet or application exits and are automatically deleted. These cookies usually store a session ID that is not personally identifiable to users, allowing the user

to move from page to page without having to log-in repeatedly. They are widely used by commercial web sites.

### Permanent or Persistent Cookies

Permanent cookies are stored in persistent storage and are not deleted when the application exits. They are deleted when they expire. They can retain user preferences for a particular web site, allowing those preferences to be used in future sessions. Permanent cookies can be used to identify individual users, so they may be used by web sites to analyze user's surfing behavior within the web site. These cookies can also be used to provide information about the numbers of visitors, the average time spent on a particular page, and the general performance of the web site. They are usually configured to keep track of users for a prolonged period of time, in some cases many years into the future.

### Methods in Servlet Cookies:

Method	Description
public void setDomain(String pattern)	sets the domain to which cookie applies.
public String getDomain()	gets the domain to which cookie applies.
public void setMaxAge(int expiry)	sets how much time (in seconds) should elapse before the cookie expires. By default, the cookie will last only for the current session.
public int getMaxAge()	returns the maximum age of the cookie, specified in seconds, By default,
public String getName()	returns the name of the cookie. The name cannot be changed after creation.
public void setValue(String newValue)	sets the value associated with the cookie.
public String getValue()	gets the value associated with the cookie.
public void setPath(String uri)	sets the path to which this cookie applies. By default, the cookie is returned for all URLs in the same directory as the current page as well as all subdirectories.
public String getPath()	gets the path to which this cookie applies.
public void setSecure(boolean flag)	sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e. SSL) connections.
public void setComment(String purpose)	specifies a comment that describes a cookie's purpose. The comment is useful if the browser presents the cookie to the user.

public String getComment()	returns the comment describing the purpose of this cookie, or null if the cookie has no comment.
public void setDomain(String pattern)	sets the domain to which cookie applies.

## Programming cookies

### Setting Cookies with Servlet

Setting cookies with servlet involves three steps:

- (1) **Creating a Cookie object:** Call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

```
Cookie cookie = new Cookie("key","value");
```

- (2) **Setting the maximum age:** Use setMaxAge() to specify how long (in seconds) the cookie should be valid. Following would set up a cookie for 20 hours.

```
cookie.setMaxAge(60*60*20);
```

- (3) **Sending the Cookie into the HTTP response headers:** Use response.addCookie() to add cookies in the HTTP response header as follows:

```
response.addCookie(cookie);
```

### Setting cookies

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

public class HelloForm extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    { Cookie firstName = new Cookie("first_name",
request.getParameter("first_name"));

        Cookie lastName = new Cookie("last_name",
request.getParameter("last_name"));

        firstName.setMaxAge(60*60*24);
        lastName.setMaxAge(60*60*24);

        response.addCookie( firstName );
        response.addCookie( lastName );

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
```

### 3.26 Server Side Programming

```
out.println( "<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>");
}}
```

#### Form.html

```
<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>
```

First Name:

Last Name:

#### Reading Cookies with Servlet:

To read cookies, create an array of `javax.servlet.http.Cookie` objects by calling the `getCookies()` method of `HttpServletRequest`. Then cycle through the array, and use `getName()` and `getValue()` methods to access each cookie and associated value.

#### Reading Cookies

```
import java.io.*;import javax.servlet.*;import javax.servlet.http.*;
public class ReadCookies extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    { Cookie cookie = null;
      Cookie[] cookies = null;
```

```

cookies = request.getCookies();
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println( "<html>\n" + "<head><title>" + title + "</title></head>\n" +
"<body bgcolor=\"#f0f0f0\">\n" );
if( cookies != null )
{ out.println("<h2> Found Cookies Name and Value</h2>");
for (int i = 0; i < cookies.length; i++)
{
    cookie = cookies[i];
    out.print("Name : " + cookie.getName( ) + ", ");
    out.print("Value: " + cookie.getValue( ) + " <br/>");
}
}
else{ out.println("<h2>No cookies found</h2>"); }
out.println("</body>"); out.println("</html>"); }}

```

Found Cookies Name and Value

Name : first\_name, Value: Priya

Name : last\_name, Value: Dharshini

### Delete Cookies with Servlet:

The following three steps will delete a cookie:

1. Read an already existing cookie and store it in Cookie object.
2. Set cookie age as zero using setMaxAge() method to delete an existing cookie.
3. Add this cookie back into response header.

```

import java.io.*;import javax.servlet.*;import javax.servlet.http.*;
public class DeleteCookies extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    Cookie cookie = null;
    Cookie[] cookies = null;
    cookies = request.getCookies();
    response.setContentType("text/html");

```

```

        PrintWriter out = response.getWriter();
        out.println("<html>\n" + "<head><title>" + title + "</title></head>\n" +
            "<body bgcolor=\"#f0f0f0\">\n" );
        if( cookies != null )    {
            out.println("<h2> Cookies Name and Value</h2>");
            for (int i = 0; i < cookies.length; i++)
                {
                    cookie = cookies[i];
                    if((cookie.getName( )).compareTo("first_name") == 0 )
                        {
                            cookie.setMaxAge(0);
                            response.addCookie(cookie);
                            out.print("Deleted cookie : " + cookie.getName( ) + "<br/>");
                        }
                    out.print("Name : " + cookie.getName( ) + ", ");
                    out.print("Value: " + cookie.getValue( )+" <br/>");    }    }
        else
            { out.println("<h2>No cookies founds</h2>");    }
            out.println("</body>");    out.println("</html>");    }}

```

```

Cookies Name and Value
Deleted cookie : first_name
Name : first_name, Value: Priya
Name : last_name, Value: Dharshini

```

To delete the cookies in Internet Explorer manually. Start at the Tools menu and select Internet Options. To delete all cookies, press Delete Cookies.

#### Advantages of cookies:

- Cookies are simple to use and implement.
- Occupies less memory as they do not require any server resources and are stored on the user's computer so no extra burden on server.
- Cookies can be configured to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client's computer (persistent cookies).
- Cookies persist a much longer period of time than Session state.

**Disadvantages of cookies:**

- Cookies are not secure as they are stored in clear text they may pose a possible security risk as anyone can open and tamper with cookies.
- Several limitations exist on the size of the cookie text (4kb in general), number of cookies(20 per site in general), etc.
- User has the option of disabling cookies on his computer from browser's settings.
- Cookies will not work if the security level is set to high in the browser.
- Users can delete cookies.
- Users browser can refuse cookies.
- Complex type of data not allowed (e.g. dataset etc). It allows only plain text (i.e. cookie allows only string content).

**3.4 CONFIGURING AND INSTALLING APACHE TOMCAT SERVER**

*Apache Tomcat server is an open source Java-capable HTTP server and servlet container developed by Apache Software Foundation(ASF).*

This could execute special Java programs known as Java Servlet and Java Server Pages (JSP). The sites for Tomcat are <http://tomcat.apache.org> or <http://www.apache.org>.

Tomcat was originally written by James Duncan Davison , based on an earlier Sun's server called Java Web Server (JWS). Tomcat is an HTTP application runs over TCP/IP. In other words, the Tomcat server runs on a specific TCP port in a specific IP address. The default TCP port number for HTTP protocol is 80, which is used for the production HTTP server. For test HTTP server, any unused port number between 1024 and 65535 can be chosen.

**Installing Apache Tomcat Server**

The basic environment required for installing Apache Tomcat Server is given below:

- JDK 6 (Java SE 6) (Tomcat 6 requires any installed Java 5 or later JRE (32-bit or 64-bit). We have used JRE 6.)
- Apache Tomcat 6.x
- Windows OS

**Step 1: Downloading Apache Tomcat**

**Download the Apache Tomcat Server from <http://tomcat.apache.org/download-60.cgi>.** Here we have used “Apache Tomcat 6.0.35 version.

### 3.30 Server Side Programming

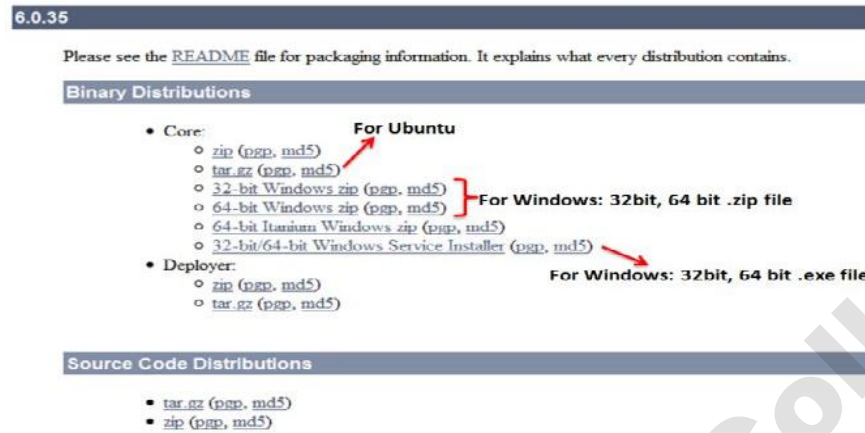


Figure 3.9 Tomcat for various OS

#### Step 2: Installing Apache Tomcat

Tomcat can be installed on any operating system that supports the zip or tar formats. To install Apache Tomcat, unzip the downloaded (.zip) file to a safe location. For simplicity and easy access, unzip Tomcat in “C:\Tomcat6\” directory.

#### Step 3: Click on the .exe file

After downloading windows installer file(.exe file), double click on it and follow the steps given below:

#### Welcome screen

Simply click on the ‘Next’ button to continue installation process.

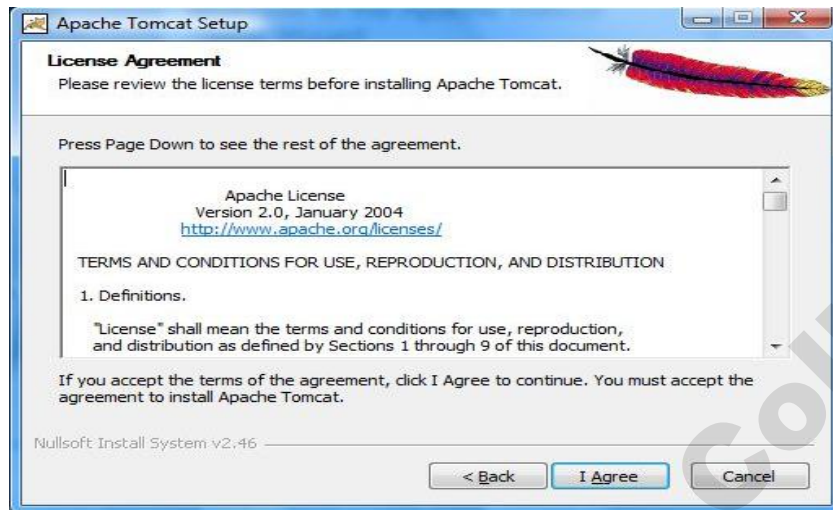


Figure 3.10 Welcome Screen

#### 3.2 License Agreement screen

Accept the terms of the agreement by clicking on ‘I Agree’ button.

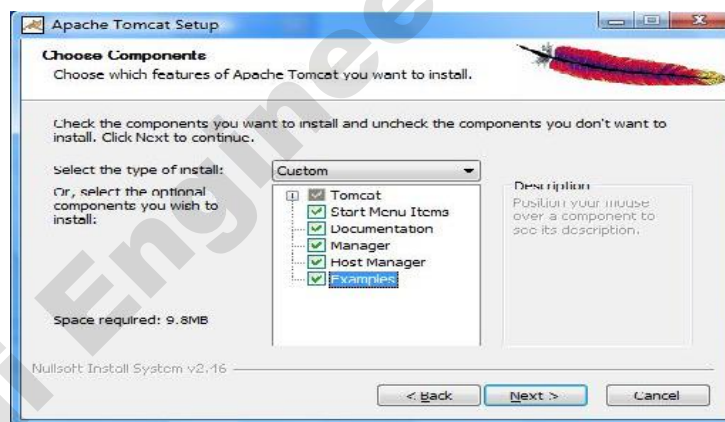




**Figure 3.11 License Agreement screen**

### 3.3 Choose the components

Choose the features of Apache Tomcat you want to install by checking the components and click 'Next'.



**Figure 3.12 Choose the components**

### 3.4 Tomcat Configuration options

The default port number for Tomcat to process HTTP requests is 8080. The port numbers can be changed after the installation in server.xml which is located in/conf/server.xml. Provide the username and password for Administrator login and click on the 'Next' button.

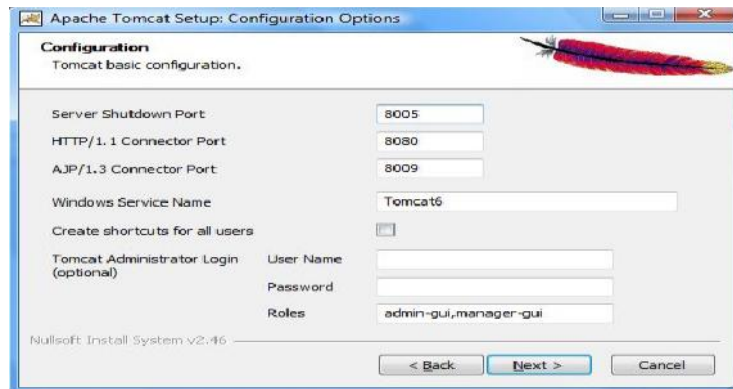


Figure 3.13 Tomcat Configuration options

### 3.5 Installed JRE path

The installer uses the registry to determine the base path of a Java 5 or later JRE, including the JRE installed as part of the full JDK. When running on a 64-bit operating system, the installer will first look for a 64-bit JRE and only look for a 32-bit JRE if a 64-bit JRE is not found. It is not mandatory to use the default JRE detected by the installer. Any installed Java 5 or later JRE (32-bit or 64-bit) may be used by clicking on the browse button and click 'Next'.

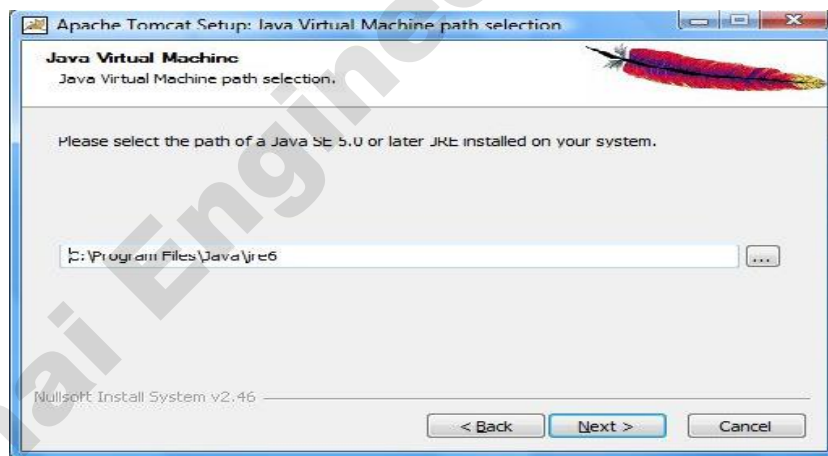
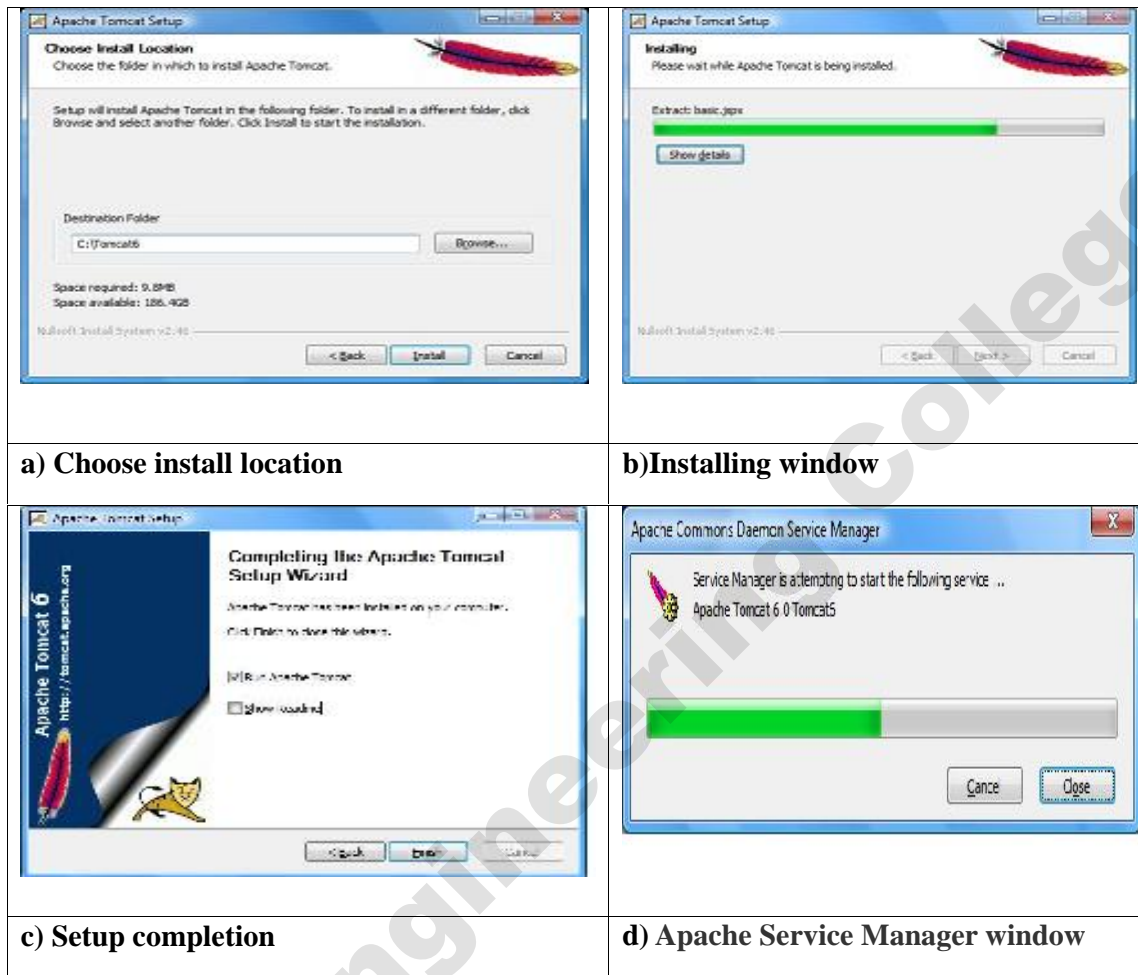


Figure 3.14 Installed JRE path

### 3.6 Choose Installation Location

In Windows, by default the location will be provided as 'C:\Program Files\Apache Software Foundation\Tomcat 6.0'. But for simplicity, we recommend you to use 'C:\Tomcat6' as shown below and click 'Install'.



**Figure 3.15 Choose Installation Location**

Once the service has been started, an **Apache Tomcat icon appears on Windows Taskbar** (bottom right).

### 3.7 Test the Installation

Open browser and type <http://localhost:8080> and see the Apache Tomcat home page as shown below.

### 3.34 Server Side Programming

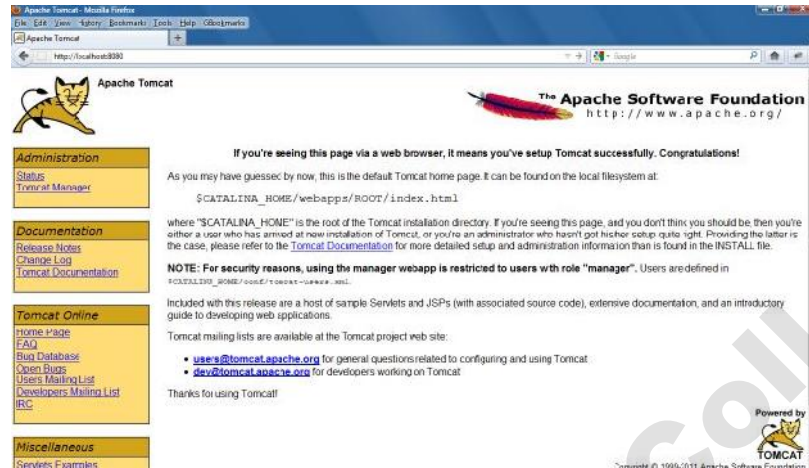


Figure 3.16 Testing the installation

### Configuring Apache Tomcat

To manually configure the server, double-click on the Tomcat icon in Taskbar to open the Apache Tomcat properties dialog. Select Manual as startup type and start the server or stop the server and click OK.

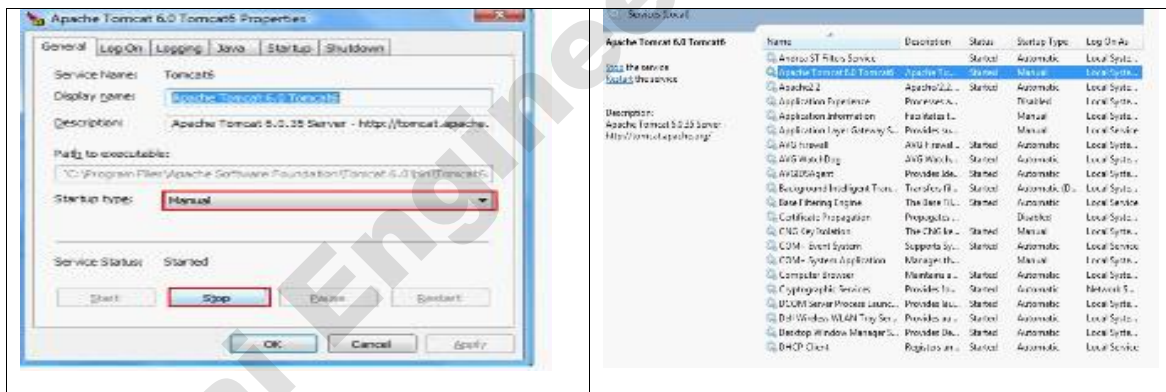


Figure 3.17 Manual configuration

Double-click on /bin/Tomcat6w.exe (in our case, it is C:\Tomcat6\bin) and follow the step as above. If it did not open the Apache Tomcat properties window then try to run it as administrator (Right click on exe file->Run as administrator). Using Windows Services: Open Control Panel\Administrative Tools and double-click on Services. Double-click on Apache Tomcat service and do as above.

### 3.5 JDBC CONNECTIVITY

JDBC provides framework to connect to relational databases from java programs.

*JDBC API provides industry-standard and database-independent connectivity between java applications and database servers.*

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

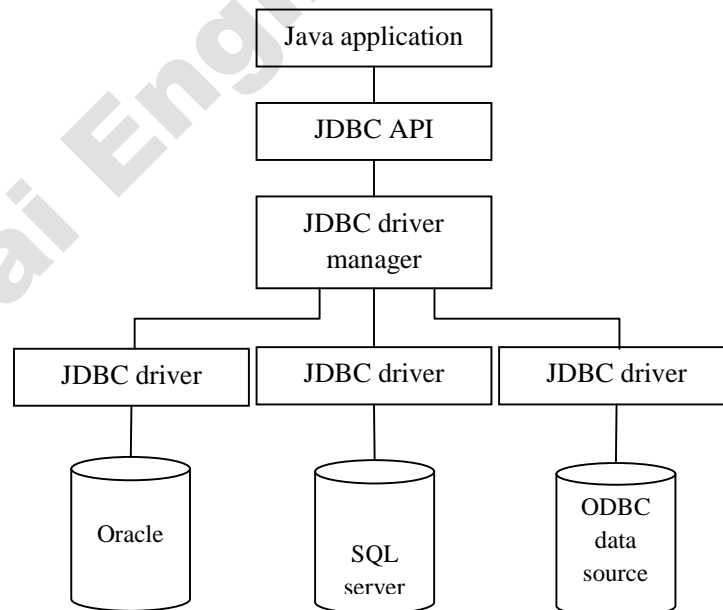
- Making a connection to a database
- Creating SQL or MySQL statements
- Executing that SQL or MySQL queries in the database
- Viewing & Modifying the resulting records

JDBC can be used with Java Applications, Java Applets, Java Servlets, Java ServerPages (JSPs) and Enterprise JavaBeans (EJBs)

#### JDBC Architecture

JDBC Architecture consists of two layers:

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.



**Figure 3.18 JDBC Architecture**

The JDBC API supports both two-tier and three-tier processing models for database access. The JDBC API uses a driver manager and database-specific drivers to provide connectivity to heterogeneous databases.

The **JDBC driver manager** ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

### **Components of JDBC**

The following are some of the important components of JDBC:

#### **DriverManager:**

This class manages a list of database drivers. It is responsible for matching the connection requests from the java application with the proper database driver using communication sub protocol.

#### **Driver:**

This interface handles the communications with the database server. The DriverManager manages these objects.

#### **Connection:**

This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

#### **Statement:**

The objects created from this interface are used to submit the SQL statements to the database.

#### **ResultSet:**

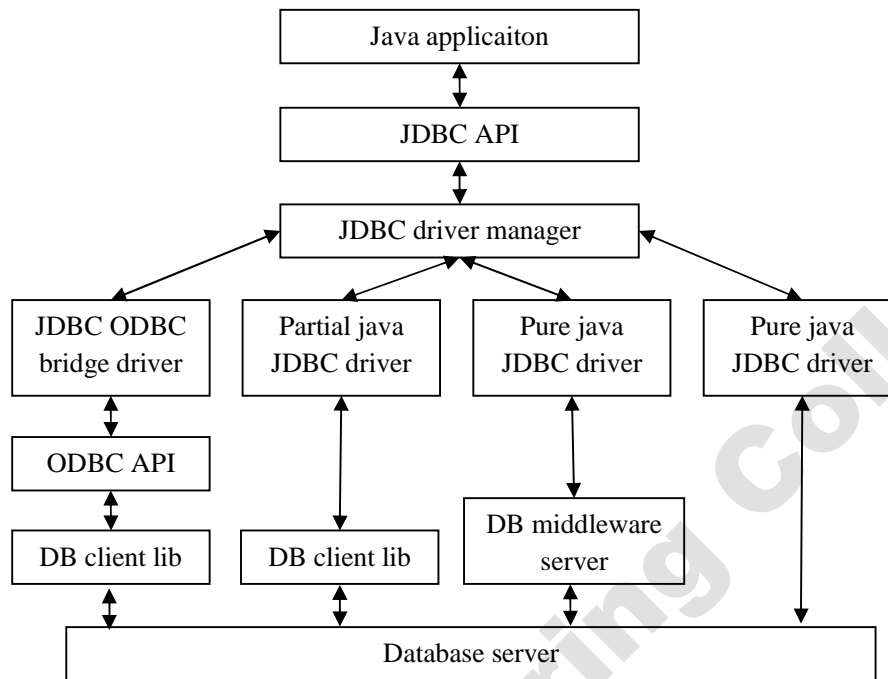
These objects hold data retrieved from a database after executing an SQL query using Statement objects. It acts as an iterator that moves through its data.

#### **SQLException:**

This class handles any errors that occur in a database application.

#### **JDBC Driver**

JDBC drivers implement the defined interfaces in the JDBC API for interacting with the database server. The JDBC drivers open database connections and interact with it by sending SQL or database commands then receiving results with Java.



**Figure 3.19 JDBC Drivers**

**1. JDBC-ODBCBridge plus ODBC Driver (Type 1):**

This driver uses ODBC driver to connect to database servers. The ODBC drivers must be installed in the machines from where the connection is established to JDBC drivers. This driver is almost obsolete and should be used only when other options are not available.

**2. Native API partly Java technology-enabled driver (Type 2):**

This type of driver converts JDBC class to the client API for the RDBMS servers. The database client API should be installed at the machine from which we want to make database connection. Because of extra dependency on database client API drivers, this is also not preferred driver.

**3. Pure Java Driver for Database Middleware (Type 3):**

This type of driver sends the JDBC calls to a middleware server that can connect to different type of databases. A middleware server must be installed to work with this kind of driver. This adds to extra network calls and slow performance. Hence this is also not widely used JDBC driver.

**4. Direct-to-Database Pure Java Driver (Type 4):**

This is the preferred driver because it converts the JDBC calls to the network protocol understood by the database server. This solution doesn't require any extra APIs at the client side and suitable for database connectivity over the network. However for this solution, we should use database specific drivers, for example OJDBC jars provided by Oracle for Oracle DB and MySQL Connector/J for MySQL databases.

### Creating a JDBC application

There are following six steps involved in building a JDBC application:

1. **Import the packages** - Include the packages containing the JDBC classes needed for database programming.
2. **Register the JDBC driver** - Initialize a driver so to open a communications channel with the database.
3. **Open a connection** - Using DriverManager.getConnection() method create a Connection object, which represents a physical connection with the database.
4. **Execute a query** - Statement for building and submitting an SQL statement to the database.
5. **Extract data from result set** - Use ResultSet.getXXX() method to retrieve the data from the result set.
6. **Clean up the environment** - Explicitly closes all database resources.

### Connections using JDBC

The JDBC connections can be established in two ways:

- By using JDBC-ODBC bridge
- By using vendor specific JDBC driver

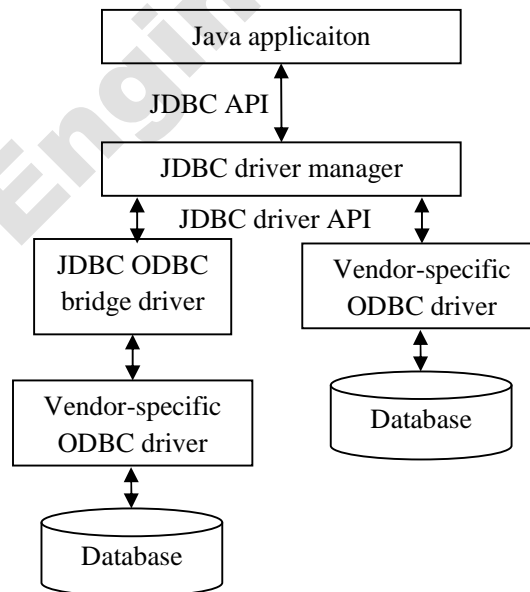


Figure 3.20 JDBC drivers



### Define the Connection URL

URLs referring to databases use the jdbc: protocol and embed the server host, port, and database name (or reference) within the URL.

### Establish the Connection

To make the actual network connection, pass the URL, database username, and database password to the getConnection method of the DriverManager class. The getConnection throws an SQLException.

```
Connection connection = DriverManager.getConnection(oracleURL, username, password);
```

RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	jdbc:db2:hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	jdbc:sybase:Tds:hostname: port Number/databaseName

### Methods of connection class:

1. preparedStatement- Creates precompiled queries for submission to the database.
2. prepareCall- Accesses stored procedures in the database.
3. rollback/commit- Controls transaction management.
4. close- Terminates the open connection.
5. isClosed- Determines whether the connection timed out or was explicitly closed.

### JDBC - Statements, PreparedStatement and CallableStatement

Once a connection is obtained we can interact with the database. The JDBC Statement, CallableStatement, and PreparedStatement interfaces define the methods and properties that enable to send SQL or PL/SQL commands and receive data from the database. They also define methods that help bridge data type differences between Java and SQL data types used in a database

Interfaces	Recommended Use
Statement	Use for general-purpose access to the database. Useful when using static SQL statements at runtime. The Statement interface cannot accept parameters.
PreparedStatement	Used when the SQL statements are to be executed many times. The PreparedStatement interface accepts input parameters at runtime.
CallableStatement	Use when database stored procedures are to be executed. The CallableStatement interface can also accept runtime input parameters.

### Create a Statement Object

A Statement object is used to send queries and commands to the database. It is created from the Connection using `createStatement`.

```
Statement statement = connection.createStatement();
```

Most, but not all, database drivers permit multiple concurrent Statement objects to be open on the same connection.

### Execute a Query or Update

The Statement object can be used to send SQL queries by using the `executeQuery` method, which returns an object of type `ResultSet`.

**Example:** `String query = "SELECT col1, col2, col3 FROM sometable";`

```
ResultSet resultSet = statement.executeQuery(query);
```

The methods in the Statement class are:

Methods	Description
<code>executeQuery</code>	Executes an SQL query and returns the data in a <code>ResultSet</code> . The <code>ResultSet</code> may be empty, but never null.
<code>executeUpdate</code>	Used for UPDATE, INSERT, or DELETE commands. Returns the number of rows affected, which could be zero. Also provides support for Data Definition Language (DDL) commands, for example, CREATE TABLE, DROP TABLE, and ALTER TABLE.

executeBatch	Executes a group of commands as a unit, returning an array with the update counts for each command. Use addBatch to add a command to the batch group.
setQueryTimeout	Specifies the amount of time a driver waits for the result before throwing a SQLException.
getMaxRows/setMaxRows	Determines the number of rows a ResultSet may contain. Excess rows are silently dropped. The default is zero for no limit.

### Process the Results

The simplest way to handle the results is to use the next method of ResultSet to move through the table a row at a time. Within a row, ResultSet provides various getXxx methods that take a column name or column index as an argument and return the result in a variety of different Java types. For instance, use getInt if the value should be an integer, getString for a String, and so on for most other data types. To just display the results, use getString for most of the column types.

```
while(resultSet.next())
{
    System.out.println(resultSet.getString(1) + " " + resultSet.getString(2) + " " +
        resultSet.getString("firstname") + " " + resultSet.getString("lastname"));
}
```

Methods	Description
next/previous	Moves the cursor to the next (any JDBC version) or previous (JDBC version 2.0 or later) row in the ResultSet, respectively.
relative/absolute	The relative method moves the cursor a relative number of rows, either positive (up) or negative (down). The absolute method moves the cursor to the given row number. If the absolute value is negative, the cursor is positioned relative to the end of the ResultSet (JDBC 2.0).
getXxx	Returns the value from the column specified by the column name or column index as an Xxx Java type (see java.sql.Types). Can return 0 or null if the value is an SQL NULL.
wasNull	Checks whether the last getXxx read was an SQL NULL.
findColumn	Returns the index in the ResultSet corresponding to the specified column name.
getRow	Returns the current row number, with the first row starting at 1 (JDBC 2.0).

getMetaData	Returns a ResultSetMetaData object describing the ResultSet.
ResultSetMetaData	Gives the number of columns and the column names.

### Close the Connection

To close the connection explicitly close() is used.

```
connection.close();
```

Closing the connection also closes the corresponding Statement and ResultSet objects.

### The PreparedStatement Objects

The PreparedStatement interface extends the Statement interface which gives added functionality with a couple of advantages over a generic Statement object. This statement gives the flexibility of supplying arguments dynamically.

#### Example:

```
PreparedStatement pstmt = null;
```

```
Try{ String SQL = "Update Employees SET age = ? WHERE id = ?";
```

```
pstmt = conn.prepareStatement(SQL);}
```

### The CallableStatement Objects

A Connection object creates the CallableStatement object which would be used to execute a call to a database stored procedure.

### Simple JDBC connection in a servlet

```
import java.io.IOException;import java.io.PrintWriter;import java.sql.Connection;
import java.sql.DriverManager;import java.sql.ResultSet;import java.sql.SQLException;
import java.sql.Statement;import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class JDBCServlet extends HttpServlet
{ public void doGet(HttpServletRequest inRequest, HttpServletResponse outResponse)
throws ServletException, IOException
{ PrintWriter out = null;
Connection connection = null;
Statement statement;
ResultSet rs;
```

```

try {
    Class.forName("com.mysql.jdbc.Driver");
    connection =
DriverManager.getConnection("jdbc:mysql://localhost/products");
    statement = connection.createStatement();
    outResponse.setContentType("text/html");
    out = outResponse.getWriter();
    rs = statement.executeQuery("SELECT ID, title, price FROM product");
    out.println("<HTML><HEAD><TITLE>Products</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<UL>");
    while (rs.next()) {
        out.println("<LI>" + rs.getString("ID") + " " + rs.getString("title")
+ " " + rs.getString("price"));
    }
    out.println("</UL>");
    out.println("</BODY></HTML>"); }
catch (ClassNotFoundException e)
{ out.println("Driver Error"); }
catch (SQLException e)
{ out.println("SQLException: " + e.getMessage()); } }
public void doPost(HttpServletRequest inRequest, HttpServletResponse outResponse)
throws ServletException, IOException
{ doGet(inRequest, outResponse); }}

```

ID	Title	Price
56	Soap	40
98	Shampoo	15

### Batch Processing

Batch Processing allows grouping related SQL statements into a batch and submitting them with one call to the database. When several SQL statements are sent to the database at once, they can be clubbed together to reduce the amount of communication overhead. This is called **batch processing**. The following methods are used in batch processing:

### 3.44 Server Side Programming

S.No	Method	Description
1.	DatabaseMetaData.supportsBatchUpdates()	This method determines if the target database supports batch update processing.
2.	addBatch()	This method is used to add individual statements to the batch.
3.	executeBatch()	This method is used to start the execution of all the statements grouped together.
4.	clearBatch()	This method removes all the statements added with the addBatch() method. The statements cannot be selectively chosen.

The following code provides an example of a batch update using Statement object:

```
Statement stmt = conn.createStatement();
conn.setAutoCommit(false); // Set auto-commit to false
String SQL = "INSERT INTO Employees (id, first, last, age) " +
"VALUES(200,'Zia', 'Ali', 30)";
stmt.addBatch(SQL); // Add above SQL statement in the batch.
String SQL = "INSERT INTO Employees (id, first, last, age) " +
"VALUES(201,'Raj', 'Kumar', 35)"; // Create one more SQL statement
stmt.addBatch(SQL); // Add above SQL statement in the batch.
String SQL = "UPDATE Employees SET age = 35 " + "WHERE id = 100";
stmt.addBatch(SQL);
int[] count = stmt.executeBatch();// Create an int[] to hold returned values
conn.commit();//Explicitly commit statements to apply changes
```

## 3.6 JAVA SERVER PAGES (JSP)

JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

*JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.*

The tags in JSP are enclosed within `<%` and `%>`. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

#### **Advantages of JSP over CGI**

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.
- JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

#### **Advantages of JSP over Active Server Pages (ASP):**

- The dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use.
- It is portable to other operating systems and non-Microsoft Web servers.

#### **Advantages of JSP over Pure Servlets:**

- It is an extension to the servlets.
- Easy to maintain than servlets
- No need for recompilation and redeployment (If JSP page is modified, we don't need to recompile and redeploy the project.).
- Less code.

#### **Advantages of JSP over JavaScript:**

- JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

#### **Advantages of JSP over Static HTML:**

- Regular HTML cannot contain dynamic information.

### Architecture of JSP:

The web server needs a JSP engine i.e. container to process JSP pages. The **JSP container** is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

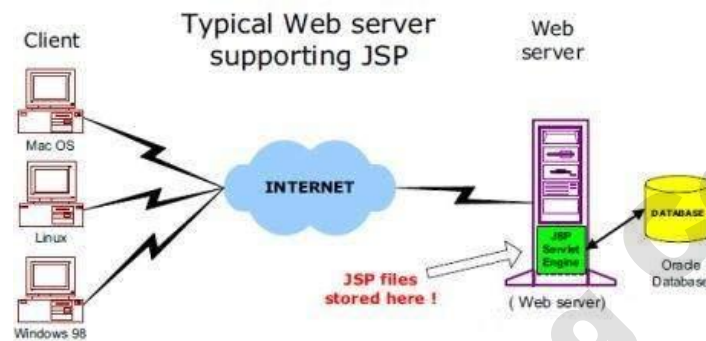


Figure 3.21 JSP-architecture

### Processing of JSP

The browser sends an HTTP request to the web server. The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**. The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to `println()` statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.

The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine. A part of the web server called the **servlet engine** loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response. The web server forwards the HTTP response to your browser in terms of static HTML content. Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.

A JSP page is really just another way to write a servlet without having to be a Java programming. Except for the translation phase, a JSP page is handled exactly like a regular servlet.



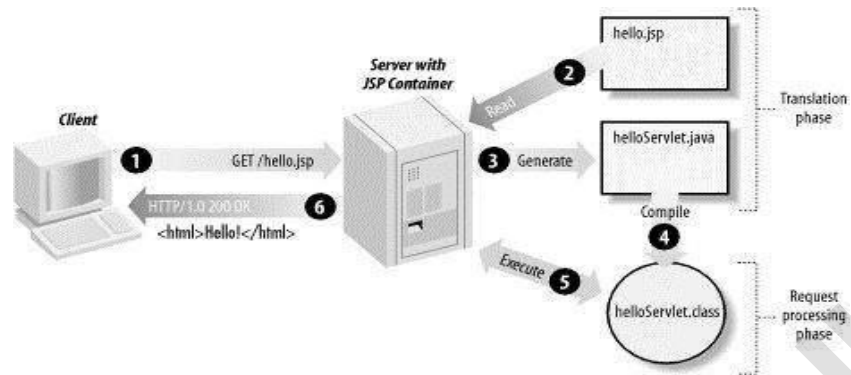


Figure 3.22 Processing of JSP

**Difference between JSP and Servlets**

JSP	Servlets
JSP is a webpage scripting language that can generate dynamic content.	Servlets are Java programs that are already compiled which also creates dynamic web content.
In MVC, JSP act as a view.	In MVC, servlet act as a controller.
It's easier to code in JSP than in Java Servlets.	More code is needed here.
JSP are generally preferred when there is not much processing of data required.	Servlets are used when there is more processing and manipulation involved.
JSP run slower compared to Servlet as it takes compilation time to convert into Java Servlets.	Servlets run faster compared to JSP.
The advantage of JSP programming over servlets is that we can build custom tags which can directly call Java beans.	There is no such custom tag facility in servlets.
We can achieve functionality of JSP at client side by runningJavaScript at client side.	There are no such methods for servlets.

**Lifecycle of JSP**

A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet. The following are the phases followed by a JSP: Compilation, Initialization, Execution and Cleanup

- **JSP Compilation:**

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page. The compilation process involves three steps: Parsing the JSP, Turning the JSP into a servlet and Compiling the servlet.

- **JSP Initialization:**

When a container loads a JSP it invokes the `jspInit()` method before servicing any requests. In case of perform JSP-specific initialization, override the `jspInit()` method:

```
public void jspInit()
{ // Initialization code...}
```

Typically initialization is performed only once and as with the servlet `init` method. The `jspInit()` initialize database connections, open files, and create lookup tables.

- **JSP Execution:**

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed. Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the `_jspService()` method in the JSP.

```
void _jspService(HttpServletRequest request, HttpServletResponse response)
{ // Service handling code...}
```

The `_jspService()` method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.

- **JSP Cleanup:**

The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container. Override `jspDestroy` to perform any cleanup, such as releasing database connections or closing open files.

```
public void jspDestroy()
{ //cleanup code }
```

### JSP Syntax

A JSP document contains the following tags: scriptlet, expressions and declaration tags.

- **Scriptlet:**

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language. Any text, HTML tags, or JSP elements must be outside the scriptlet.

1. `<% code fragment %>` OR
2. `<jsp:scriptlet> code fragment </jsp:scriptlet>` (XML format)

### Scriptlet

```

<html><head><title>Hello </title></head>
<body>
Hello <br/>
<%out.println("The IP address is " + request.getRemoteAddr());
%>
</body></html>

```

### Declaration

This part declares one or more variables or methods that used in Java code later in the JSP file.

1. `<%! declaration; [ declaration; ]+ ... %>` OR
2. `<jsp:declaration> code fragment </jsp:declaration>` (XML format)

Example: `<%! int i = 0; %> ;<%! int a, b, c; %> ;<%! Circle a = new Circle(2.0); %>`

- **JSP Expression:**

A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file. Because the value of an expression is converted to a String, the expression could be used within a line of text, whether or not it is tagged with HTML, in a JSP file. The expression element can contain any expression that is valid according to the Java Language Specification but semicolon to end an expression is invalid.

1. `<%= expression %>` OR
2. `<jsp:expression> expression</jsp:expression>` (XML format)

### Expression tags

#### Index.html

```

<html><body>
<form action="welcome.jsp">
<input type="text" name="uname"><br/>

```

```
<input type="submit" value="go">
</form></body></html>
```

**Hai.jsp**

```
<html>
<body>
<%= "Hai"+request.getParameter("uname") %>
</form></body></html>
```

In this example, the username is displayed using the expression tag. The index.html file gets the username and sends the request to the hai.jsp file, which displays the username.

- **JSP Comments:**

- JSP comment marks text or statements that the JSP container should ignore.

```
<%-- This is JSP comment --%>
```

**Declarations and Comments:**

```
<html><head><title>A Comment Test</title></head>
<body><h2> Comments</h2><p>
  Today's date: <%= (new java.util.Date()).toLocaleString()%>
<%-- Displays today's date --%>
</p></body></html>
```

```
A Comment Test
Today's date: 01-JAN-2015 21:24:25
```

**JSP Directives:**

A JSP directive affects the overall structure of the servlet class.

```
<%@ directive attribute="value" %>
```

Directive	Description
<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
<%@ include ... %>	Includes a file during the translation phase.
<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page.

**1. The page Directive:**

The page directive is used to provide instructions to the container that pertain to the current JSP page. Page directives can be used anywhere in the JSP page. By convention, page directives are coded at the top of the JSP page.

1. `<%@ page attribute="value" %>` OR
2. `<jsp:directive.page attribute="value" />` (XML format)

**2. The include Directive:**

The include directive is used to includes a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase.

1. `<%@ include file="relative url" >` OR
2. `<jsp:directive.include file="relative url" />` (XML Format)

**3. The taglib Directive:**

The JavaServer Pages API allows to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior. The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in a JSP page.

1. `<%@ taglib uri="uri" prefix="prefixOfTag" >` OR
2. `<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />` (XML Format)

Where the uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.

**JSP Actions:**

JSP actions use constructs in XML syntax to control the behavior of the servlet engine.

`<jsp:action_name attribute="value" />`

Syntax	Purpose
jsp:include	Includes a file at the time the page is requested
jsp:useBean	Finds or instantiates a JavaBean
jsp:setProperty	Sets the property of a JavaBean

jsp:getProperty	Inserts the property of a JavaBean into the output
jsp:forward	Forwards the requester to a new page
jsp:plugin	Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin
jsp:element	Defines XML elements dynamically.
jsp:attribute	Defines dynamically defined XML element's attribute.
jsp:body	Defines dynamically defined XML element's body.
jsp:text	Use to write template text in JSP pages and documents.

**Attributes:**

There are two attributes that are common to all Action elements:

- **Id attribute:** The id attribute uniquely identifies the Action element, and allows the action to be referenced inside the JSP page. If the Action creates an instance of an object the id value can be used to reference it through the implicit object PageContext
- **Scope attribute:** This attribute identifies the lifecycle of the Action element. The id attribute and the scope attribute are directly related, as the scope attribute determines the lifespan of the object associated with the id. The scope attribute has four possible values: (a) page, (b)request, (c)session, and (d) application.

**The <jsp:include> Action:**

Let us define following two files (a)date.jps and (b) main.jsp as follows:

**date.jsp file:**

```
<p>
  Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

**main.jsp file:**

```
<html><head><title>The include Action Example</title>
</head>
<body><center><h2>The include action Example</h2>
```

```
<jsp:include page="date.jsp" flush="true" />
</center></body></html>
```

The include action Example

Today's date: 12-Sep-2010 14:54:22

### JSP Implicit Objects:

JSP Implicit Objects are the Java objects that the JSP Container makes available to developers in each page and developer can call them directly without being explicitly declared. JSP supports nine automatically defined variables, which are also called implicit objects.

Objects	Description
request	This is the HttpServletRequest object associated with the request.
response	This is the HttpServletResponse object associated with the response to the client.
out	This is the PrintWriter object used to send output to the client.
session	This is the HttpSession object associated with the request.
application	This is the ServletContext object associated with application context.
config	This is the ServletConfig object associated with the page.
pageContext	This encapsulates use of server-specific features like higher performance JspWriters.
page	This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.
Exception	The Exception object allows the exception data to be accessed by designated JSP.

1. out.print(dataType dt)- Print a data type value
2. config.getServletName()-gets the name of the servlet using config object.
3. pageContext.removeAttribute("attrName", PAGE\_SCOPE) -removes the attribute from page scope.

### Control-Flow Statements

JSP provides full power of Java to be embedded in the web application. All the APIs and building blocks of Java can be used in JSP programming including decision making statements, loops etc.

### Decision-Making Statements

The if...else block starts out like an ordinary Scriptlet, but the Scriptlet is closed at each line with HTML text included between Scriptlet tags.

#### If.else

<pre>&lt;%! int day = 3; %&gt;&lt;html&gt; &lt;head&gt;&lt;title&gt;IF...ELSE Example&lt;/title&gt;&lt;/head&gt; &lt;body&gt;&lt;% if (day == 1   day == 7) { %&gt;&lt;p&gt; Today is weekend&lt;/p&gt; &lt;% } else { %&gt;&lt;p&gt; Today is not weekend&lt;/p&gt; &lt;% } %&gt;&lt;/body&gt;&lt;/html&gt;</pre>
Today is not weekend

#### Switch:

<pre>&lt;%! int day = 3; %&gt; &lt;html&gt;&lt;head&gt;&lt;title&gt;SWITCH...CASE Example&lt;/title&gt;&lt;/head&gt; &lt;body&gt; &lt;% switch(day) { case 0:out.println("It\'s Sunday."); break; case 1:out.println("It\'s Monday."); break; case 2:out.println("It\'s Tuesday."); break; case 3:out.println("It\'s Wednesday."); break; case 4:out.println("It\'s Thursday."); break; case 5:out.println("It\'s Friday."); break; default:out.println("It\'s Saturday.");} %&gt; &lt;/body&gt;&lt;/html&gt;</pre>
It's Wednesday.



**Loop Statements:**

All three basic types of looping blocks in Java can be used in JSP: for, while, and do...while blocks

**For loop**

```

<%! int fontSize; %>
<html>
<head><title>FOR LOOP Example</title></head>
<body><%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
<font color="green" size="<%= fontSize %>">
    JSP </font><br /><%}%> </body></html>

```

JSP
JSP
JSP

**JSP Operators:**JSP supports all the logical and arithmetic operators supported by Java.

**JSP Literals:**The JSP expression language defines the following literals: Boolean: true and false, Integer: as in Java, Floating point: as in Java, String: with single and double quotes; "is escaped as \", ' is escaped as \', and \ is escaped as \\ and null.

**JSP Standard Tag Library (JSTL)**

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

**Advantage of JSTL**

- Fast development: JSTL provides many tags that simplifies the JSP.
- Code reusability: We can use the JSTL tags in various pages.
- It avoids the use of scriptlet tag.

There JSTL mainly provides 5 types of tags:

Tag Name	Description
core tags	The JSTL core tag provide variable support, URL management, flow control etc. The url for the core tag is <a href="http://java.sun.com/jsp/jstl/core">http://java.sun.com/jsp/jstl/core</a> . The prefix of core tag is c.

sql tags	The JSTL sql tags provide SQL support. The url for the sql tags is <a href="http://java.sun.com/jsp/jstl/sql">http://java.sun.com/jsp/jstl/sql</a> and prefix is sql.
xml tags	The xml sql tags provide flow control, transformation etc. The url for the xml tags is <a href="http://java.sun.com/jsp/jstl/xml">http://java.sun.com/jsp/jstl/xml</a> and prefix is x.
internationalization tags	The internationalization tags provide support for message formatting, number and date formatting etc. The url for the internationalization tags is <a href="http://java.sun.com/jsp/jstl/fmt">http://java.sun.com/jsp/jstl/fmt</a> and prefix is fmt.
functions tags	The functions tags provide support for string manipulation and string length. The url for the functions tags is <a href="http://java.sun.com/jsp/jstl/functions">http://java.sun.com/jsp/jstl/functions</a> and prefix is fn.

### JSTL Core Tags

The JSTL core tags mainly provides 4 types of tags:

- miscellaneous tags: catch and out.
- url management tags: import, redirect and url.
- variable support tags: remove and set.
- flow control tags: forEach, forTokens, if and choose.

### Syntax for core tags

s% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

Tag	Description
<b>c:catch</b>	It handles the exception and doesn't propagate the exception to error page. The exception object thrown at runtime is stored in a variable named var.
<b>c:out</b>	It is just like JSP expression tag but it is used for expression. It renders data to the page.
<b>c:import</b>	It is just like jsp include but it can include the content of any resource either within server or outside the server.
<b>c:forEach</b>	It repeats the nested body content for fixed number of times or over collection.
<b>c:if</b>	It tests the condition.
<b>c:redirect</b>	It redirects the request to the given url.

## HTML forms with JSP tags

The browser uses two methods to pass some information to web server: GET Method and POST Method. The data entered in the forms reach the server through these methods.

### GET method:

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

**Example:** `http://www.abc.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 the browser's Location:box. Never use the GET method if password or other sensitive information is passed to the server.

The GET method has size limitation: only 1024 characters can be in a request string. This information is passed using QUERY\_STRING header and will be accessible through QUERY\_STRING environment variable which can be handled using `getQueryString()` and `getParameter()` methods of request object.

### POST method:

A more reliable method of passing information to a backend program is the POST method. This method packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? 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 can be parsed and use for processing. JSP handles this type of requests using `getParameter()` method to read simple parameters and `getInputStream()` method to read binary data stream coming from the client.

### Reading Form Data using JSP

JSP handles form data parsing automatically using the following methods depending on the situation:

Methods	Description
<code>getParameter()</code>	to get the value of a form parameter.
<code>getParameterValues()</code>	Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
<code>getParameterNames()</code>	Call this method to get a complete list of all parameters in the current request.
<code>getInputStream()</code>	Call this method to read binary data stream coming from the client.

### Get():

#### Form.html

```
<html><body><form action="main.jsp" 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>
```

#### Main.jsp

```
<html><head><title>Using GET Method to Read Form Data</title>  
</head><body><center>  
<h1>Using GET Method to Read Form Data</h1>  
<ul><li><p><b>First Name:</b>  
<%= request.getParameter("first_name")%></p></li>  
<li><p><b>Last Name:</b>  
<%= request.getParameter("last_name")%>  
</p></li>  
</ul></body></html>
```



First Name: SONA  
Last Name: RAJ Submit

### POST Method Using Form:

#### Form.html

```
<html><body><form action="main.jsp" 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>
```

**Main.jsp**

```

<html><head>
<title>Using GET and POST Method to Read Form Data</title></head>
<body><center>
<h1>Using GET Method to Read Form Data</h1>
<ul><li><p><b>First Name:</b>
<%= request.getParameter("first_name")%></p></li>
<li><p><b>Last Name:</b>
<%= request.getParameter("last_name")%></p></li>
</ul></body></html>

```

First Name:   
 Last Name:

**Reading All Form Parameters:****Form.html**

```

<html><body>
<form action="main.jsp" method="POST" target="_blank">
<input type="checkbox" name="maths" checked="checked" /> Maths
<input type="checkbox" name="physics" /> Physics
<input type="checkbox" name="chemistry" checked="checked" /> Chem
<input type="submit" value="Select Subject" />
</form></body></html>

```

**Main.jsp**

```

<%@ page import="java.io.*,java.util.*" %>
<html><head><title>HTTP Header Request Example</title>
</head>
<body><center><h2>HTTP Header Request Example</h2>
<table width="100%" border="1" align="center">

```

### 3.60 Server Side Programming

```
<tr bgcolor="#949494">
<th>Param Name</th><th>Param Value(s)</th></tr>
<%
Enumeration paramNames = request.getParameterNames();
while(paramNames.hasMoreElements()) {
String paramName = (String)paramNames.nextElement();
out.print("<tr><td>" + paramName + "</td>\n");
String paramValue = request.getHeader(paramName);
out.println("<td>" + paramValue + "</td></tr>\n");
}%></table></center></body></html>
```

Maths  Physics  Chem

Param Name	Param Value(s)
maths	on
chemistry	on

## REVIEW QUESTIONS

### PART-A

#### 1. Define servlets.

Servlets are dynamically loaded Java code running on a JVM that services the requests from a web server.

#### 2. What is CGI (Common Gateway Interface)?

The common gateway interface (CGI) is a standard way for a Web server to pass a Web user's request to an application program and to receive data back to forward to the user. CGI does not communicate directly with the browser. The browser communicates with the server, the server in turn talks with the CGI program and CGI answers to the server which is rendered to the browser.

#### 3. List the advantages of Servlets over CGI

There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the processes such as they share a common memory area, light weight, cost of communication between the threads are low. The basic benefits of servlet are as follows:

- Better performance: because it creates a thread for each request not process.
- Portability: because it uses java language.
- Robust: Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
- Secure: because it uses java language.

#### 4. Differentiate between process and threads

Process	Threads
Process run in separate address space	Threads of a process share address space.
Process have their own copy of data segment of parent process.	Threads have direct access to data segment of its process
Processes must use inter-process communication to communicate with sibling processes	Threads can directly communicate with other threads of that process.
Process carry more state information	Threads carry less state information
New process require duplication of parent process, and allocation of memory and resources for it are costly	New threads are easily created.

**5. Give the functionality of void destroy().**

This “cleanup” method is called by the web container before removing the servlet instance from the service. Resources used by the servlet, such as an open file or an open database connection, should be deallocated here.

**6. What are the events in servlet lifecycle?**

The following are the phases in the life of a servlet:

- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.

**7. What is Servlet Interface?**

Servlet interface provides common behaviour to all the servlets. Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).

**8. Write about ServletConfig Interface.**

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

```
publicServletConfiggetServletConfig(); //This returns the object of the ServletConfig.
```

**9. What is the use of GenericServlet class?**

GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method. GenericServlet class can handle any type of request so it is protocol-independent.

**10. What is ServletRequest Interface?**

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

**11. List the methods to Read Form Data using Servlet.**

Servlets handle form data parsing automatically using the following methods depending on the situation:



1. `getParameter()`-to get the value of a form parameter.
2. `getParameterValues()`-this method is used if the parameter appears more than once and returns multiple values(example checkbox).
3. `getParameterNames()`-this method is used when a complete list of all parameters is needed in the current request.

## 12. Define session handling.

Tracking is a way to maintain state (data) of a user. It is also known as session management in servlet.Http is a stateless protocol that means each request is considered as the new request. So the states are maintained using various session tracking techniques.Eachtime user requests to the server, server treats the request as the new request. So we need to maintain the state of a user to recognize to particular user.

## 13. Mention the Session Tracking Techniques

There are four techniques used in Session tracking:Cookies, Hidden Form Field, URL Rewriting and HttpSession Interface

## 14. Define cookies.

A cookie is a small piece of information that persist between the multiple client requests.In HTTP each request is considered as a new request even if two requests are issued by the same user.A cookie is added with response from the servlet which serves as an identifier to the user. This is done when the first request is made.So cookie is stored in the cache of the browser. After that if any request is sent by the user, the stored cookie is added with request by default. Thus, the server recognizes the user as the old user.

## 15. Differentiate between Cookie and Session

Session	Cookie
Sessions are server-side files that contain user information	Cookies are client-side files that contain user information
Any amount of data can be stored within sessions.	Official MAX Cookie size is 4KB
Session ends when user close his browser.	Cookie ends depends on the life time you set for it.

## 16. What areHidden Form fields?

Here, a hidden (invisible) textfield is used for maintaining the state of end user. In such case, the information is stored in the hidden field. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

<input type="hidden" name="Field name" value="value">

Example: <INPUT TYPE="HIDDEN" NAME="session" VALUE="a1234">

### **17. List the advantages and disadvantages of hidden form fields.**

Advantage of Hidden Form Field

1. It will always work whether cookie is disabled or not.

Disadvantages of Hidden Form Field:

1. It is maintained at server side.
2. Extra form submission is required on each pages.
3. Only textual information can be used.

### **18. What is URL Rewriting?**

Appending the name of the user in the query string and getting the value from the query string in another page is called URL rewriting. The parameter name-value pairs are passed using the following format:

url?name1=value1&name2=value2&??

### **19. List the advantages and disadvantages of URL rewriting.**

Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send only textual information.

### **20. What is HTTP Session Interface?**

The web container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks: bind objects and view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.

### **21. List the components of cookies.**

Cookies are a plain text data record of 5 variable-length fields:

- Expiry time: The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- Domain: The domain name of the site.
- Path: The path to the directory or web page that set the cookie.
- Secure: If this field contains the word "secure" then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- Name=Value: Cookies are set and retrieved in the form of key and value pairs.

**22. Give the uses of cookies.**

Identifying a user during an e-commerce session, Avoiding username and password, Customizing a site and Focusing advertising

**23. List the types of cookies**

There are two types of cookies: Session Cookies and persistent cookies

**24. Define Session Cookies.**

Session cookies are stored in memory during the applet or application session. Session cookies expire when the applet or application exits and are automatically deleted. These cookies usually store a session ID that is not personally identifiable to users, allowing the user to move from page to page without having to log-in repeatedly. They are widely used by commercial web sites.

**25. What are Permanent or Persistent Cookies?**

Permanent cookies are stored in persistent storage and are not deleted when the application exits. They are deleted when they expire. They can retain user preferences for a particular web site, allowing those preferences to be used in future sessions. Permanent cookies can be used to identify individual users, so they may be used by web sites to analyze user's surfing behavior within the web site. These cookies can also be used to provide information about the numbers of visitors, the average time spent on a particular page, and the general performance of the web site. They are usually configured to keep track of users for a prolonged period of time, in some cases many years into the future.

**26. How to delete cookies?**

The following three steps will delete a cookie:

1. Read an already existing cookie and store it in Cookie object.
2. Set cookie age as zero using setMaxAge() method to delete an existing cookie.
3. Add this cookie back into response header.

**27. List the advantages of cookies.**

- Cookies are simple to use and implement.
- Occupies less memory as they do not require any server resources and are stored on the user's computer so no extra burden on server.
- Cookies can be configured to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client's computer (persistent cookies).
- Cookies persist a much longer period of time than Session state.

**28. Give the disadvantages of cookies.**

- Cookies are not secure as they are stored in clear text they may pose a possible security risk as anyone can open and tamper with cookies.
- Several limitations exist on the size of the cookie text (4kb in general), number of cookies(20 per site in general), etc.
- User has the option of disabling cookies on his computer from browser's settings.
- Cookies will not work if the security level is set to high in the browser.
- Users can delete cookies.
- Users browser can refuse cookies.
- Complex type of data not allowed (e.g. dataset etc). It allows only plain text (i.e. cookie allows only string content).

**29. What is Apache Tomcat?**

Apache Tomcat server is an open source Java-capable HTTP server and servlet container developed by Apache Software Foundation(ASF).

**30. What is JDBC?**

JDBC API provides industry-standard and database-independent connectivity between javaapplications and database servers.

**31. Define JDBC driver manager.**

The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases

**32. What is DriverManager?**

This class manages a list of database drivers. It is responsible for matching the connection requests from the java application with the proper database driver using communication sub protocol.

**33. Define Driver software.**

This interface handles the communications with the database server. The DriverManager manages these objects.

**34. What is Statement object?**

The objects created from this interface are used to submit the SQL statements to the database.

**35. What is ResultSet object?**

These objects hold data retrieved from a database after executing an SQL query using Statement objects. It acts as an iterator that moves through its data.

**36. What is JDBC Driver?**

JDBC drivers implement the defined interfaces in the JDBC API for interacting with the database server. The JDBC drivers open database connections and interact with it by sending SQL or database commands then receiving results with Java.

**37. What is batch processing in JDBC?**

Batch Processing allows grouping related SQL statements into a batch and submitting them with one call to the database. When several SQL statements are sent to the database at once, they can be clubbed together to reduce the amount of communication overhead. This is called batch processing

**38. Define JSP.**

JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.

**39. List the advantages of JSP over Active Server Pages (ASP).**

- The dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use.
- It is portable to other operating systems and non-Microsoft Web servers.

**40. Give the advantages of JSP over Pure Servlets.**

- It is an extension to the servlets.
- Easy to maintain than servlets

- No need for recompilation and redeployment (If JSP page is modified, we don't need to recompile and redeploy the project.).
- Less code.

**41. List the advantages of JSP over JavaScript.**

- JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

**42. Give the advantages of JSP over Static HTML.**

- Regular HTML cannot contain dynamic information.

**43. What is JSP Implicit Objects?**

JSP Implicit Objects are the Java objects that the JSP Container makes available to developers in each page and developer can call them directly without being explicitly declared. JSP supports nine automatically defined variables, which are also called implicit objects.

**44. List the advantage of JSTL.**

- Fast development: JSTL provides many tags that simplifies the JSP.
- Code reusability: We can use the JSTL tags in various pages.
- It avoids the use of scriptlet tag.

**45. List the core tages of JSTL.**

The JSTL core tags mainly provides 4 types of tags:

- miscellaneous tags: catch and out.
- url management tags: import, redirect and url.
- variable support tags: remove and set.
- flow control tags: forEach, forTokens, if and choose.

**PART-B**

1. Describe servlet lifecycle.
2. Give the servlet architecture.
3. Explain all the methods in servlets.

4. Elaborate the form and post actions.
5. Brief about session handling.
6. Explain cookies.
7. How to configure and installing Apache tomcat server?
8. Explain in detail about JDBC drivers.
9. Describe the various methods in establishing servlet connection.
10. Describe the result processing methods.
11. Explain in detail about JSP.
12. Brief about various syntactical structures in JSP.
13. Write about JSP tags.

Arunai Engineering College

---

---

# 4

## PHP and XML 8

---

---

### 4.1 INTRODUCTION TO PHP

*The PHP Hypertext Pre-processor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases*

PHP is basically used for developing web based software applications. PHP is probably the most popular scripting language on the web. It is used to enhance web pages. PHP is known as a **server-sided language**. That is because the PHP doesn't get executed on the client's computer, but on the computer the user had requested the page from. The results are then handed over to client, and then displayed in the browser.

#### Features of PHP:

- ❖ PHP is a server side scripting language that is embedded in HTML
- ❖ PHP was originally developed by the **Danish Greenlander Rasmus Lerdorf**, and was subsequently developed as open source.
- ❖ It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- ❖ It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- ❖ PHP supports a large number of protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- ❖ PHP language tries to be as forgiving as possible.
- ❖ PHP syntax is C-Like.

#### Common uses of PHP

- ❖ PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.



- ❖ PHP can handle forms, i.e. gather data from files, save data to a file, through email the user can send data, return data to the user.
- ❖ The user can add, delete, and modify elements within the database through PHP.
- ❖ They can access cookies variables and set cookies.
- ❖ Using PHP, the user can restrict users to access some pages of the website.
- ❖ It can encrypt data.

### Working of PHP

When the client requests a PHP page residing on the server, the server first performs the operations mentioned by the PHP code of the page. Then it sends the output of the PHP page in HTML format. So when the user views the source code of the page, it will be full of HTML tags. All the work is done at the server side.

## 4.2 PROGRAMMING WITH PHP

### ➤ Comments

A comment is the portion of a program that exists only for the human reader and is stripped out before displaying the program's result. There are two commenting formats in PHP:

- **Single-line comments:** They are generally used for short explanations or notes relevant to the local code.
- **Multi-line comments:** They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C.

### Rules of PHP

- ❖ PHP is white space insensitive
- ❖ PHP is case sensitive
- ❖ Statements are expressions terminated by semicolons
- ❖ Expressions are combinations of tokens
- ❖ Braces make blocks

### ➤ PHP Variable Types

All variables in PHP are denoted with a leading dollar sign (\$). The value of a variable is the value of its most recent assignment. Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.

Variables in PHP do not have **intrinsic types (data types)** - a variable does not know in advance whether it will be used to store a number or a string of characters.

Variables used before they are assigned have default values. PHP automatically converts types from one to another when necessary. PHP has a total of eight data types:

- Integers are whole numbers, without a decimal point. They can be in decimal, octal or hexadecimal. Eg: 87.
- Doubles are floating-point numbers. Eg: 3.87
- Booleans have only two possible values either true or false.
- NULL is a special type that only has one value: NULL
- Strings are sequences of characters
- Arrays are named and indexed collections of other values.
- Objects are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources are special variables that hold references to resources external to PHP (such as database connections).
- The first five are simple types, and the arrays and objects are compound types.
- The compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

### **Variable Scope**

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types: Local variables, Function parameters, Global variables and Static variables.

### **Variable Naming**

Rules for naming a variable are:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but the user cannot use characters like + , - , % , ( , ) . & , etc

### **PHP Constants**

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. To define a constant ,

use `define()` function and retrieve the value of a constant. The function `constant()` is used to read a constant's value .

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

The name specifies the name of the constant, value: Specifies the value of the constant and case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

### Differences between constants and variables in PHP

Constants in PHP	Variables in PHP
No \$ sign before constants.	\$ sign is present before variables.
Constants are defined using <code>define()</code> .	Variables are defined using assignment statement.
Constants may be defined and accessed anywhere without any regard.	Variables follow certain scope.
Constants cannot be redefined or undefined.	They can be redefined.

### Pre-defined constants

Name	Description
<code>__LINE__</code>	The current line number of the file.
<code>__FILE__</code>	The full path and filename of the file. If used inside an include, the name of the included file is returned. Since PHP 4.0.2, <code>__FILE__</code> always contains an absolute path whereas in older versions it contained relative path under some circumstances
<code>__FUNCTION__</code>	The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
<code>__CLASS__</code>	The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
<code>__METHOD__</code>	The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive).

### ➤ Echo and Print statements

#### Differences between echo and print statement

echo	print
This does not have return value.	This has a return value of 1.
This cannot be used in expressions.	This can be used in expressions.
This can take multiple parameters.	This can take only one parameter.
This is slightly faster than print.	This is comparatively slower.

### ➤ 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 and Array operators.

## PHP CONTROL STATEMENTS

### Decision Making Statements

The if, else if ...else and switch statements are used to take decision based on the different condition.

- if statement - executes some code only if a specified condition is true.
- if...else statement - executes some code if a condition is true and another code if the condition is false.
- if...else if...else statement - specifies a new condition to test, if the first condition is false
- switch statement - selects one of many blocks of code to be executed

### ➤ if statement

#### if statement

```
<?php
$t = date("H");
if ($t < "20") //This program outputs the echo statement if the hour is <20
{
    echo "Have a good day!";
}
?>
```

➤ **if-else statement**

Use the if...else statement to execute some code if a condition is true and another code if the condition is false.

**if else statement**

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

➤ **if-else ladder**

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

**if-else ladder**

```
<?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!";
}
?>
```

➤ **switch statement**

To select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code. The switch statement works in an unusual way. First it evaluates given expression then seeks a label to match the resulting value. If a matching value is found then the code associated with the matching

label will be executed or if none of the labels match then statement will execute any specified default code.

### Switch statement

```
<html>
<body>
<?php $d=date("D");
switch ($d)
{
case "Mon":
    echo "Today is Monday";
    break;
case "Tue":
    echo "Today is Tuesday";
    break;
case "Wed":
    echo "Today is Wednesday";
    break;
case "Thu":
    echo "Today is Thursday";
    break;
case "Fri":
    echo "Today is Friday";
    break;
case "Sat":
    echo "Today is Saturday";
    break;
case "Sun":
    echo "Today is Sunday";
    break;
default:
    echo "Wonder which day is this ?";
```

```
}  
?>  
</body></html>
```

### Looping Statements

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types

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

#### ➤ For loop

##### for loop

```
<html>  
<body>  
<?php  
$a = 0;  
$b = 0;  
for( $i=0; $i<5; $i++ )  
{  
    $a += 10;  
    $b += 5;  
}  
echo ("At the end of the loop a=$a and b=$b" );  
?>  
</body></html>
```

At the end of the loop a=50 and b=25

#### ➤ While loop

The while statement will execute a block of code as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

**While loop**

```

<html> <body>
<?php
$i = 0;
$num = 50;
while( $i < 10)
{
$num--;
$i++;
}
echo ("Loop stopped at i = $i and num = $num" );
?> </body></html>

```

Loop stopped at i = 1 and num = 40

**➤ Do...while loop**

```

<html> <body>
<?php
$i = 0;
$num = 0;
do {
    $i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i" );
?></body></html>

```

Loop stopped at i = 10

**➤ For Each loop**

The for each statement is used to loop through arrays. For each pass the value of the current array element is assigned to \$value and the array pointer is moved by one and in the next pass next element will be processed.

```

<html> <body>
<?
php $array = array( 11, 12, 13,14, 15);

```



```
foreach( $array as $value )  
{  
    echo "Value is $value <br />";  
} ?> </body></html>
```

```
Value is 11  
Value is 12  
Value is 13  
Value is 14  
Value is 15
```

➤ **Break statement**

The PHP break keyword is used to terminate the execution of a loop prematurely. The break statement is situated inside the statement block. After coming out of a loop immediate statement to the loop will be executed.

➤ **Continue statement**

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop. Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

## FUNCTIONS

A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page loads. A function will be executed by a call to the function. There are two types of functions: Built-in functions and User defined functions

### User defined Functions

A user defined function declaration starts with the word function. 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.

```
<?php  
function sum($x, $y) {  
    $z = $x + $y;  
    return $z; }  
echo "5 + 10 = " . sum(5, 10) . "<br>";
```

```
echo "7 + 13 = " . sum(7, 13) . "<br>";
```

```
echo "2 + 4 = " . sum(2, 4); ?>
```

```
5 + 10 = 15
```

```
7 + 13 = 20
```

```
2 + 4 = 6
```

## Built-in functions

### Array functions

Function	Description
array()	Creates an array
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values() )	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array
array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_multisort()	Sorts multiple or multi-dimensional arrays

4.12 PHP and XML 8

array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values

count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Returns the current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array
key()	Fetches a key from an array
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
prev()	Rewinds the internal array pointer
range()	Creates an array containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

### ➤ Calendar Functions

The calendar extension contains functions that simplify converting between different calendar formats. It is based on the Julian Day Count, which is a count of days starting from January 1st, 4713 B.C. To convert between calendar formats, first convert to Julian Day Count, then to the calendar of the user's choice.

Function	Description
cal_days_in_month()	Returns the number of days in a month for a specified year and calendar
cal_from_jd()	Converts a Julian Day Count into a date of a specified calendar

cal_info()	Returns information about a specified calendar
cal_to_jd()	Converts a date in a specified calendar to Julian Day Count
easter_date()	Returns the Unix timestamp for midnight on Easter of a specified year
easter_days()	Returns the number of days after March 21, that the Easter Day is in a specified year
gregoriantojd()	Converts a Gregorian date to a Julian Day Count
jddayofweek()	Returns the day of the week
jdmonthname()	Returns a month name
jdtogregorian()	Converts a Julian Day Count to a Gregorian date
jdtonix()	Converts Julian Day Count to Unix timestamp
jewishtojd()	Converts a Jewish date to a Julian Day Count
juliantojd()	Converts a Julian date to a Julian Day Count
unixtojd()	Converts Unix timestamp to Julian Day Count

#### ➤ Date Functions

The date/time functions allow to get the date and time from the server on which PHP script runs. These functions depend on the locale settings of the server. Remember to take daylight saving time and leap years into consideration when working with these functions.

Function	Description
checkdate()	Validates a Gregorian date
date_add()	Adds days, months, years, hours, minutes, and seconds to a date
date_create_from_format()	Returns a new DateTime object formatted according to a specified format
date_create()	Returns a new DateTime object
date_date_set()	Sets a new date
date_default_timezone_get()	Returns the default timezone used by all date/time functions
date_default_timezone_set()	Sets the default timezone used by all date/time functions

date_diff()	Returns the difference between two dates
date_format()	Returns a date formatted according to a specified format
date_interval_format()	Formats the interval
date_isodate_set()	Sets the ISO date
date_modify()	Modifies the timestamp
date_parse()	Returns an associative array with detailed info about a specified date
date_sub()	Subtracts days, months, years, hours, minutes, and seconds from a date
date_sun_info()	Returns an array containing info about sunset/sunrise and twilight begin/end, for a specified day and location
date_sunrise()	Returns the sunrise time for a specified day and location
date_sunset()	Returns the sunset time for a specified day and location
date_time_set()	Sets the time
date()	Formats a local date and time
getdate()	Returns date/time information of a timestamp or the current local date/time
gettimeofday()	Returns the current time
gmdate()	Formats a GMT/UTC date and time
gmmktime()	Returns the Unix timestamp for a GMT date
gmstrftime()	Formats a GMT/UTC date and time according to locale settings
idate()	Formats a local time/date as integer
localtime()	Returns the local time
microtime()	Returns the current Unix timestamp with microseconds
mktime()	Returns the Unix timestamp for a date

strftime()	Formats a local time and/or date according to locale settings
time()	Returns the current time as a Unix timestamp
timezone_name_get()	Returns the name of the timezone
timezone_offset_get()	Returns the timezone offset from GMT
timezone_open()	Creates new DateTimeZone object
timezone_version_get()	Returns the version of the timezone db

### ➤ Directory functions

The directory function allows retrieving information about directories and their contents.

Function	Description
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes a directory handle
dir()	Returns an instance of the Directory class
getcwd()	Returns the current working directory
opendir()	Opens a directory handle
readdir()	Returns an entry from a directory handle
rewinddir()	Resets a directory handle
scandir()	Returns an array of files and directories of a specified directory

### ➤ Error handling functions

The error functions are used to deal with error handling and logging. The error functions allow us to define own error handling rules, and modify the way the errors can be logged. The logging functions allow us to send messages directly to other machines, emails, or system logs. The error reporting functions allow us to customize what level and kind of error feedback is given.

Function	Description
debug_backtrace()	Generates a backtrace
debug_print_backtrace()	Prints a backtrace
error_get_last()	Returns the last error that occurred
error_log()	Sends an error message to a log, to a file, or to a mail account
error_reporting()	Specifies which errors are reported
restore_error_handler()	Restores the previous error handler
restore_exception_handler()	Restores the previous exception handler
set_error_handler()	Sets a user-defined error handler function
set_exception_handler()	Sets a user-defined exception handler function
trigger_error()	Creates a user-level error message
user_error()	Alias of trigger_error()
debug_backtrace()	Generates a backtrace

### ➤ File system Functions

The file system functions allow the user to access and manipulate the file system.

Function	Description
basename()	Returns the filename component of a path
chgrp()	Changes the file group
chmod()	Changes the file mode
chown()	Changes the file owner
clearstatcache()	Clears the file status cache
copy()	Copies a file
dirname()	Returns the directory name component of a path
disk_free_space()	Returns the free space of a directory
disk_total_space()	Returns the total size of a directory



fclose()	Closes an open file
feof()	Tests for end-of-file on an open file
fflush()	Flushes buffered output to an open file
fgetc()	Returns a character from an open file
fgets()	Returns a line from an open file
fgetss()	Returns a line, with HTML and PHP tags removed, from an open file
file()	Reads a file into an array
file_exists()	Checks whether or not a file or directory exists
file_get_contents()	Reads a file into a string
file_put_contents()	Writes a string to a file
fileatime()	Returns the last access time of a file
filectime()	Returns the last change time of a file
filegroup()	Returns the group ID of a file
fileinode()	Returns the inode number of a file
filemtime()	Returns the last modification time of a file
fileowner()	Returns the user ID (owner) of a file
fileperms()	Returns the permissions of a file
filesize()	Returns the file size
filetype()	Returns the file type
flock()	Locks or releases a file
fnmatch()	Matches a filename or string against a specified pattern
fopen()	Opens a file or URL
fpasssthru()	Reads from an open file, until EOF, and writes the result to the output buffer
fputcsv()	Formats a line as CSV and writes it to an open file
fread()	Reads from an open file
fscanf()	Parses input from an open file according to a specified format

fseek()	Seeks in an open file
fstat()	Returns information about an open file
ftell()	Returns the current position in an open file
ftruncate()	Truncates an open file to a specified length
fwrite()	Writes to an open file
glob()	Returns an array of filenames / directories matching a specified pattern
is_dir()	Checks whether a file is a directory
is_executable()	Checks whether a file is executable
is_file()	Checks whether a file is a regular file
is_link()	Checks whether a file is a link
is_readable()	Checks whether a file is readable
is_uploaded_file()	Checks whether a file was uploaded via HTTP POST
is_writable()	Checks whether a file is writeable
lchgrp()	Changes group ownership of symlink
lchown()	Changes user ownership of symlink
link()	Creates a hard link
linkinfo()	Returns information about a hard link
lstat()	Returns information about a file or symbolic link
mkdir()	Creates a directory
move_uploaded_file()	Moves an uploaded file to a new location
pathinfo()	Returns information about a file path
pclose()	Closes a pipe opened by popen()
popen()	Opens a pipe
readfile()	Reads a file and writes it to the output buffer
readlink()	Returns the target of a symbolic link
realpath()	Returns the absolute pathname
realpath_cache_get()	Returns realpath cache entries

realpath_cache_size()	Returns realpath cache size
rename()	Renames a file or directory
rewind()	Rewinds a file pointer
rmdir()	Removes an empty directory
set_file_buffer()	Sets the buffer size of an open file
stat()	Returns information about a file
symlink()	Creates a symbolic link
touch()	Sets access and modification time of a file
umask()	Changes file permissions for files
unlink()	Deletes a file

➤ **Math functions**

Function	Description
abs()	Returns the absolute (positive) value of a number
acos()	Returns the arc cosine of a number
acosh()	Returns the inverse hyperbolic cosine of a number
asin()	Returns the arc sine of a number
asinh()	Returns the inverse hyperbolic sine of a number
atan()	Returns the arc tangent of a number in radians
atan2()	Returns the arc tangent of two variables x and y
atanh()	Returns the inverse hyperbolic tangent of a number
bindec()	Converts a binary number to a decimal number
ceil()	Rounds a number up to the nearest integer
cos()	Returns the cosine of a number
cosh()	Returns the hyperbolic cosine of a number
decbin()	Converts a decimal number to a binary number
dechex()	Converts a decimal number to a hexadecimal number
decoct()	Converts a decimal number to an octal number
deg2rad()	Converts a degree value to a radian value

exp()	Calculates the exponent of e
expm1()	Returns $\exp(x) - 1$
floor()	Rounds a number down to the nearest integer
fmod()	Returns the remainder of x/y
getrandmax()	Returns the largest possible value returned by rand()
hexdec()	Converts a hexadecimal number to a decimal number
hypot()	Calculates the hypotenuse of a right-angle triangle
max()	Returns the highest value in an array, or the highest value of several specified values
min()	Returns the lowest value in an array, or the lowest value of several specified values
octdec()	Converts an octal number to a decimal number
pi()	Returns the value of PI
pow()	Returns x raised to the power of y
rad2deg()	Converts a radian value to a degree value
rand()	Generates a random integer
round()	Rounds a floating-point number
sin()	Returns the sine of a number
sinh()	Returns the hyperbolic sine of a number
sqrt()	Returns the square root of a number
srand()	Seeds the random number generator
tan()	Returns the tangent of a number
tanh()	Returns the hyperbolic tangent of a number

➤ **String functions**

Function	Description
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts

<code>convert_cyr_string()</code>	Converts a string from one Cyrillic character-set to another
<code>convert_uuencode()</code>	Decodes a uuencoded string
<code>convert_uuencode()</code>	Encodes a string using the uuencode algorithm
<code>count_chars()</code>	Returns information about characters used in a string
<code>crc32()</code>	Calculates a 32-bit CRC for a string
<code>crypt()</code>	One-way string encryption (hashing)
<code>echo()</code>	Outputs one or more strings
<code>explode()</code>	Breaks a string into an array
<code>fprintf()</code>	Writes a formatted string to a specified output stream
<code>hex2bin()</code>	Converts a string of hexadecimal values to ASCII characters
<code>html_entity_decode()</code>	Converts HTML entities to characters
<code>htmlentities()</code>	Converts characters to HTML entities
<code>implode()</code>	Returns a string from the elements of an array
<code>join()</code>	Alias of <code>implode()</code>
<code>lcfirst()</code>	Converts the first character of a string to lowercase
<code>levenshtein()</code>	Returns the Levenshtein distance between two strings
<code>localeconv()</code>	Returns locale numeric and monetary formatting information
<code>ltrim()</code>	Removes whitespace or other characters from the left side of a string
<code>number_format()</code>	Formats a number with grouped thousands
<code>ord()</code>	Returns the ASCII value of the first character of a string
<code>parse_str()</code>	Parses a query string into variables
<code>print()</code>	Outputs one or more strings
<code>printf()</code>	Outputs a formatted string
<code>rtrim()</code>	Removes whitespace or other characters from the right side of a string
<code>setlocale()</code>	Sets locale information
<code>sha1()</code>	Calculates the SHA-1 hash of a string

sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
strstr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strncmp()	String comparison of the first n characters (case-sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string

strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)
strtok()	Splits a string into smaller strings
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
strtr()	Translates certain characters in a string
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
fprintf()	Writes a formatted string to a specified output stream
printf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

➤ **Miscellaneous Functions**

Function	Description
connection_aborted()	Checks whether the client has disconnected
connection_status()	Returns the current connection status
connection_timeout()	Deprecated in PHP 4.0.5. Checks whether the script has timed out
constant()	Returns the value of a constant

define()	Defines a constant
defined()	Checks whether a constant exists
die()	Prints a message and exits the current script
eval()	Evaluates a string as PHP code
exit()	Prints a message and exits the current script
get_browser()	Returns the capabilities of the user's browser
halt_compiler()	Halts the compiler execution
pack()	Packs data into a binary string
php_check_syntax()	Deprecated in PHP 5.0.5
php_strip_whitespace()	Returns the source code of a file with PHP comments and whitespace removed
sleep()	Delays code execution for a number of seconds
sys_getloadavg()	Gets system load average
time_nanosleep()	Delays code execution for a number of seconds and nanoseconds
time_sleep_until()	Delays code execution until a specified time
uniqid()	Generates a unique ID
unpack()	Unpacks data from a binary string
usleep()	Delays code execution for a number of microseconds

## PHP COOKIES

*Cookies are small data stored on the client computer and they are kept of use tracking purpose.*

PHP transparently supports HTTP cookies. There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.



**➤ Setting Cookies with PHP:**

PHP provided **setcookie()** function to set a cookie. This function requires six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

```
setcookie(name, value, expire, path, domain, security);
```

- **Name:** This sets the name of the cookie and is stored in an environment variable called `HTTP_COOKIE_VARS`. This variable is used while accessing cookies.
- **Value:** This sets the value of the named variable and is the content that the user wants to store.
- **Expiry:** This specifies a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the browser is closed.
- **Path:** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain:** This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security:** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which means cookie can be sent by regular HTTP.

**Setting a cookie**

```
<?php
    setcookie("name", "John Watkin", time()+3600, "/", "", 0);
    setcookie("age", "36", time()+3600, "/", "", 0); ?>
<html><head> <title>Setting Cookies with PHP</title> </head>
<body>
<?php echo "Set Cookies"?> </body></html>
```

**➤ Accessing Cookies with PHP**

PHP provides many ways to access cookies. Simplest way is to use either `$_COOKIE` or `$HTTP_COOKIE_VARS` variables. `isset()` function is used to check if a cookie is set or not.

**Accessing cookies**

```

<html><head> <title>Accessing Cookies with PHP</title> </head>
<body> <?php
echo $_COOKIE["name"]. "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"]. "<br />";
echo $_COOKIE["age"]. "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"]. "<br />";
?> </body></html>

```

**➤ Deleting Cookie**

To delete a cookie users should call `setcookie()` with the name argument only but this does not always work well, however, and should not be relied on. It is safest to set the cookie with a date that has already expired

```

<?php
setcookie( "name", "", time()- 60, "/", "", 0);
setcookie( "age", "", time()- 60, "/", "", 0);
?>

<html><head> <title>Deleting Cookies with PHP</title> </head>
<body>
<?php echo "Deleted Cookies" ?> </body></html>

```

**REGULAR EXPRESSIONS IN PHP**

*Regular expressions are nothing more than a sequence or pattern of characters.  
They provide the foundation for pattern-matching functionality.*

PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression: POSIX Regular Expressions and PERL Style Regular Expressions.

**POSIX Regular Expressions**

- The structure of a POSIX regular expression is similar to a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.

- The simplest regular expression is one that matches a single character.
  - **Brackets:** Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.
  - **Quantifiers:** The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character having a specific connotation. The +, \*, ?, {int. range}, and \$ flags all follow a character sequence.
  - **Predefined Character Ranges:** For programming convenience several predefined character ranges, also known as character classes, are available. Character classes specify an entire range of characters, for example, the alphabet or an integer set.

Expression	Description
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.
p+	It matches any string containing at least one p.
p*	It matches any string containing zero or more p's.
p?	It matches any string containing zero or more p's. This is just an alternative way to use p*.
p{N}	It matches any string containing a sequence of N p's
p{2,3}	It matches any string containing a sequence of two or three p's.
p{2, }	It matches any string containing a sequence of at least two p's.
p\$	It matches any string with p at the end of it.
^p	It matches any string with p at the beginning of it.
[:alpha:]	It matches any string containing alphabetic characters aA through zZ.
[:digit:]	It matches any string containing numerical digits 0 through 9.
[:alnum:]	It matches any string containing alphanumeric characters aA through zZ and 0 through 9.
[:space:]	It matches any string containing a space.

PHP currently offers seven functions for searching strings using POSIX-style regular expressions:

Function	Description
<u>ereg()</u>	The <code>ereg()</code> function searches a string specified by string for a string specified by pattern, returning true if the pattern is found, and false otherwise.
<u>ereg_replace()</u>	The <code>ereg_replace()</code> function searches for string specified by pattern and replaces pattern with replacement if found.
<u>eregi()</u>	The <code>eregi()</code> function searches throughout a string specified by pattern for a string specified by string. The search is not case sensitive.
<u>eregi_replace()</u>	The <code>eregi_replace()</code> function operates exactly like <code>ereg_replace()</code> , except that the search for pattern in string is not case sensitive.
<u>split()</u>	The <code>split()</code> function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.
<u>spliti()</u>	The <code>spliti()</code> function operates exactly in the same manner as its sibling <code>split()</code> , except that it is not case sensitive.
<u>sql_regcase()</u>	The <code>sql_regcase()</code> function can be thought of as a utility function, converting each character in the input parameter string into a bracketed expression containing two characters.

### PERL Style Regular Expressions:

Perl-style regular expressions are similar to their POSIX counterparts. The POSIX syntax can be used almost interchangeably with the Perl-style regular expression functions

- **Metacharacters:** A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

Metacharacters	Description
.	a single character
\s	a whitespace character (space, tab, newline)
\S	non-whitespace character
\d	a digit (0-9)
\D	a non-digit

<code>\w</code>	a word character (a-z, A-Z, 0-9, _)
<code>\W</code>	a non-word character
<code>[aeiou]</code>	matches a single character in the given set
<code>[^aeiou]</code>	matches a single character outside the given set
<code>(foo bar baz)</code>	matches any of the alternatives specified

**Modifiers:** Several modifiers are available that can make the user work with regexps much easier, like case sensitivity, searching in multiple lines etc.

Modifiers	Description
I	Makes the match case insensitive
M	Specifies that if the string has newline or carriage return characters, the ^ and \$ operators will now match against a newline boundary, instead of string boundary
O	Evaluates the expression only once
S	Allows use of . to match a newline character
X	Allows the user to use white space in the expression for clarity
G	Globally finds all matches
Cg	Allows a search to continue even after a global match fails

### PHP's Regexp PERL Compatible Functions

Functions	Description
<code>preg_match()</code>	The <code>preg_match()</code> function searches string for pattern, returning true if pattern exists, and false otherwise.
<code>preg_match_all()</code>	The <code>preg_match_all()</code> function matches all occurrences of pattern in string.
<code>preg_replace()</code>	The <code>preg_replace()</code> function operates just like <code>ereg_replace()</code> , except that regular expressions can be used in the pattern and replacement input parameters.
<code>preg_split()</code>	The <code>preg_split()</code> function operates exactly like <code>split()</code> , except that regular expressions are accepted as input parameters for pattern.
<code>preg_grep()</code>	The <code>preg_grep()</code> function searches all elements of <code>input_array</code> , returning all elements matching the regexp pattern.
<code>preg_quote()</code>	Quote regular expression characters

## DATABASE CONNECTIVITY

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

### Opening Database Connection:

PHP provides `mysql_connect` function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or `FALSE` on failure.

```
connection mysql_connect(server,user,password,new_link,client_flag);
```

- **Server:** The host name running database server. If not specified then default value is `localhost:3306`. This is optional.
- **User:** The username accessing the database. If not specified then default is the name of the user that owns the server process. This is optional.
- **Password:** The password of the user accessing the database. If not specified then default is an empty password.
- **new\_link:** If a second call is made to `mysql_connect()` with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned. This is optional.
- **client\_flags:** This is optional. A combination of the following constants:
  1. `MYSQL_CLIENT_SSL` - Use SSL encryption
  2. `MYSQL_CLIENT_COMPRESS` - Use compression protocol
  3. `MYSQL_CLIENT_IGNORE_SPACE` - Allow space after function names
  4. `MYSQL_CLIENT_INTERACTIVE` - Allow interactive timeout seconds of inactivity before closing the connection

### ➤ Closing Database Connection:

PHP uses `mysql_close` to close a database connection. This function takes connection resource returned by `mysql_connect` function. It returns `TRUE` on success or `FALSE` on failure. If a resource is not specified then last opened database is closed.

```
bool mysql_close ( resource $link_identifier );
```

### ➤ Creating a Database:

To create and delete a database the users should have admin privilege. PHP uses `mysql_query` function to create a MySQL database. This function takes two parameters and returns `TRUE` on success or `FALSE` on failure.

```
bool mysql_query( sql, connection );
```

- **Sql:** SQL query to create a database
- **Connection:** if not specified then last opened connection by `mysql_connect` will be used.

➤ **Selecting a Database:**

Once the user establishes a connection with a database server then it is required to select a particular database where all the tables are associated. This is required because there may be multiple databases residing on a single server. PHP provides function `mysql_select_db` to select a database. It returns `TRUE` on success or `FALSE` on failure.

```
bool mysql_select_db( db_name, connection );
```

- **db\_name:** Database name to be selected
- **connection:** if not specified then last opened connection by `mysql_connect` will be used.

➤ **Creating Database Tables:**

To create tables in the new database the user need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using `mysql_query()` function.

**Creating and selecting a database table**

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass); //Creating a connection
if(! $conn )
{ die('Could not connect: ' . mysql_error()); }
echo 'Connected successfully';
$sql = 'CREATE TABLE employee( ' . 'emp_id INT NOT NULL AUTO_INCREMENT, ' .
      'emp_name VARCHAR(20) NOT NULL, ' . 'emp_address VARCHAR(20) NOT NULL, ' .
      'emp_salary INT NOT NULL, ' . 'join_date timestamp(14) NOT NULL, ' .
      'primary key ( emp_id ) ); //Creating a table
mysql_select_db('test_db'); //Setting a table
$retval = mysql_query( $sql, $conn );
if(! $retval )
```

```
{ die('Could not create table: '.mysql_error()); }
echo "Table employee created successfully\n";
mysql_close($conn); //Closing a connection ?>
```

In case the user need to create many tables then it is better to create a text file first and put all the SQL commands in that text file and then load that file into \$sql variable and execute those commands.

#### ➤ **Deleting a Database:**

If a database is no longer required then it can be deleted forever. The users can use pass an SQL command to mysql\_query to delete a database.

#### ➤ **Deleting a Table:**

It is again a matter of issuing one SQL command through mysql\_query function to delete any database table. But be very careful while using this command because by doing so the users can delete some important information the user has in the table.

#### **Deleting a table**

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{ die('Could not connect: '.mysql_error()); }
$sql = 'DROP TABLE employee'; //Deleting a table
$retval = mysql_query( $sql, $conn );
if(! $retval )
{ die('Could not delete table employee: '.mysql_error()); }
echo "Table deleted successfully\n";
mysql_close($conn); ?>
```

#### ➤ **Insert Data into MySQL Database**

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function mysql\_query. In real application, all the values will be taken using HTML



form and then those values will be captured using PHP script and finally they will be inserted into MySQL tables.

### Inserting values

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{ die('Could not connect: ' . mysql_error()); }
$sql = 'INSERT INTO employee ' . '(emp_name,emp_address, emp_salary, join_date) ' .
      'VALUES ( "guest", "XYZ", 2000, NOW() )';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{ die('Could not enter data: ' . mysql_error()); }
echo "Entered data successfully\n";
mysql_close($conn); ?>
```

### ➤ Getting Data From MySQL Database

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function `mysql_query`. The user have several options to fetch data from MySQL. The most frequently used option is to use function `mysql_fetch_array()`. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

### Fetching data from tables

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
```

```

if(! $conn )
{ die('Could not connect: ' . mysql_error()); }
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{ die('Could not get data: ' . mysql_error()); }
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{ echo "EMP ID :{$row['emp_id']} <br> ".
      "EMP NAME : {$row['emp_name']} <br> ".
      "EMP SALARY : {$row['emp_salary']} <br> "; }
echo "Fetched data successfully\n";
mysql_close($conn); ?>

```

### ➤ Deleting Data from MySQL Database

Data can be deleted from MySQL tables by executing SQL DELETE statement through PHP function `mysql_query`. To delete a record in any table it is required to locate that record by using a conditional clause.

#### Deleting data from tables

```

<html> <head> <title>Delete a Record from MySQL Database</title> </head>
<body> <?php
if(isset($_POST['delete']))
{ $dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{ die('Could not connect: ' . mysql_error()); }
$emp_id = $_POST['emp_id'];
$sql = "DELETE employee WHERE emp_id = $emp_id" ; //Query to delete a record
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );

```

```

if(! $retval )
{ die('Could not delete data: ' . mysql_error()); }
echo "Deleted data successfully\n";
mysql_close($conn); }
else
{ ?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="400" border="0" cellspacing="1" cellpadding="2"> <tr>
<td width="100">Employee ID</td>
<td><input name="emp_id" type="text" id="emp_id"></td> </tr>
<tr> <td width="100"></td>
<td></td> </tr>
<tr> <td width="100"></td>
<td> <input name="delete" type="submit" id="delete" value="Delete"> </td> </tr>
</table> </form>
<?php } ?> </body></html>

```

### ➤ Updating Data into MySQL Database

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function `mysql_query`. To update a record in any table it is required to locate that record by using a conditional clause.

#### Updating data

```

<html><head> <title>Update a Record in MySQL Database</title> </head><body>
<?php
if(isset($_POST['update']))
{ $dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{ die('Could not connect: ' . mysql_error()); }
$emp_id = $_POST['emp_id'];

```

```

$emp_salary = $_POST['emp_salary'];
$sql = "UPDATE employee SET emp_salary = $emp_salary WHERE emp_id = $emp_id"
;
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{ die('Could not update data: '. mysql_error()); }
echo "Updated data successfully\n";
mysql_close($conn); }
else {?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="400" border="0" cellspacing="1" cellpadding="2">
<tr> <td width="100">Employee ID</td>
<td><input name="emp_id" type="text" id="emp_id"></td> </tr>
<tr> <td width="100">Employee Salary</td>
<td><input name="emp_salary" type="text" id="emp_salary"></td> </tr>
<tr> <td width="100"></td> <td></td> </tr>
<tr> <td width="100"></td> <td>
<input name="update" type="submit" id="update" value="Update"> </td> </tr>
</table> </form> <?php }
?> </body></html>

```

### 4.3 XML (Extensible Markup Language)

XML is a text-based markup language derived from Standard Generalized Markup Language (SGML). It is a new markup language, developed by the W3C (World Wide Web Consortium), mainly to overcome the limitations in HTML. **Markup** is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

*XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable.*

XML is not a programming language. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML. XML don't have pre-defined tags and the tags are stricter than HTML.

**XML is extensible:** XML allows the user to create user defined self-descriptive tags, or language that suits the application.

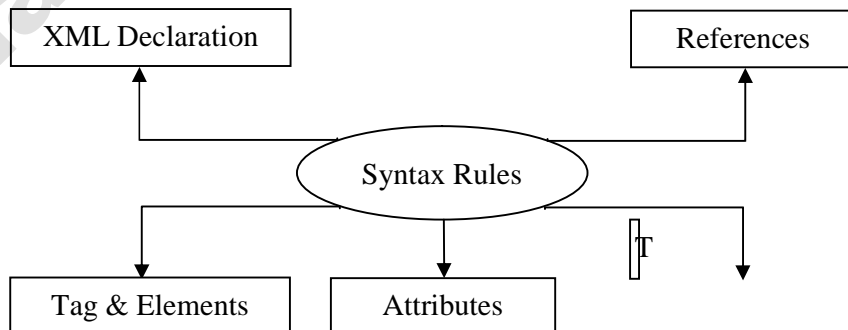
**XML carries the data, does not present it:** XML allows storing the data irrespective of how it will be presented.

**XML is a public standard:** XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

### Features of XML

- ❖ XML simplifies the creation of HTML documents for large web sites.
- ❖ XML can be used to exchange the information between organizations and systems.
- ❖ XML can be used for offloading and reloading of databases.
- ❖ XML can be used to store and arrange the data, which can customize the user data handling needs.
- ❖ XML can easily be merged with style sheets to create almost any desired output.
- ❖ Any type of data can be expressed as an XML document
- ❖ It has syndicated content, where content is being made available to different web sites.
- ❖ It suits well for electronic commerce applications where different organizations collaborate to serve a customer.
- ❖ It supports scientific applications with new markup languages for mathematical and chemical formulas.
- ❖ It also supports electronic books with new markup languages to express rights and ownership.
- ❖ XML could also be used in handheld devices and smart phones with new markup languages optimized for these devices.

### Syntax Rules



**Fig 4.1 Syntax Rules of XML**

### ➤ XML Declaration

The XML document can optionally have an XML declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Version is the XML version and encoding specifies the character encoding used in the document.

### Syntax Rules for XML declaration

- The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.
- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of encoding that the user put in the XML declaration.

### ➤ Tags and Elements

An XML file is structured by several XML-elements also called **XML-nodes** or XML-tags. XML-elements' names are enclosed by angular brackets <>.

```
<element>...</element> or
```

```
<element/>
```

### Nesting of elements

An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

```
<?xml version="1.0"?>
<contact-info>
<company>abc</company>
</contact-info>
```

- **Root element:** An XML document can have only one root element.
- **Case sensitivity:** The names of XML-elements are case-sensitive.

### ➤ Attributes

An attribute specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. It is possible to attach additional information to elements in the form of attributes. The names follow the same rules as element names.

```
<tel preferred="true">513-555-8889</tel>
```

### ➤ XML References

References allow the user to add or include additional text or markup in an XML document. References always begin with the symbol "&", which is a reserved character and end with the symbol ";". XML has two types of references:

- **Entity References:** An entity reference contains a name between the start and the end delimiters. The entity reference is replaced by the content of the entity.
- **Character References:** A letter is replaced by its Unicode character code. Character references that start with &#x provides a hexadecimal representation of the character code.

### XML Text

The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files. Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored. Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly.

### XML Documents

An XML document is a basic unit of XML information composed of elements and other markup in an orderly package. An XML document can contains wide variety of data.

```
<?xml version="1.0"?> //document prolog
<contact-info> //all the following lines are document element
<name>Sharanya</name>
<company>abc</company>
<phone>(044)257887296</phone>
</contact-info>
```

The XML documents are divided into: Document prolog section and document element sections.

**Document prolog:** The document prolog comes at the top of the document, before the root element. This section contains: XML declaration and Document type declaration. The first line in the above example belongs to prolog section.

**Document element:** Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose. The other lines except the first line come under document element section.

## XML Declaration

The XML declaration is the first line of the document. The declaration identifies the document as an XML document. The declaration also lists the version of XML used in the document. The declaration can contain other attributes to support other features such as character set encoding.

### Syntax:

```
<?xml
  version="version_number"
  encoding="encoding_declaration"
  standalone="standalone_status"
?>
```

- ❖ **Version**- Specifies the version of the XML standard used.
- ❖ **Encoding declaration**- It defines the character encoding used in the document. UTF-8 is the default encoding used.
- ❖ **Standalone**- It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to no. Setting it to yes tells the processor there are no external declarations required for parsing the document.

### Rules for XML declaration

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: version, encoding and standalone.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. </?xml>

Example	Description
<?xml >	XML declaration with no parameters.
<?xml version="1.0">	XML declaration with version definition.



<code>&lt;?xml version="1.0" encoding="UTF-8" standalone="no" ?&gt;</code>	XML declaration with all parameters defined.
<code>&lt;?xml version='1.0' encoding='iso-8859-1' standalone='no' ?&gt;</code>	XML declaration with all parameters defined in single quotes.

### XML tags

XML tags form the foundation of XML. They define the scope of an element in the XML. They can also be used to insert comments, declare settings required for parsing the environment and to insert special instructions.

- **Start Tag:** The beginning of every non-empty XML element is marked by a start-tag.

**Example:** `<address>`.

- **End Tag:** Every element that has a start tag should end with an end-tag.

**Example:** `</address>`

- **Empty Tag:** The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways:

(1) A start-tag immediately followed by an end-tag :`<hr></hr>`

(2) A complete empty-element tag : `<hr />`

### XML Tags Rules

- XML tags are case-sensitive.
- XML tags must be closed in an appropriate order, i.e., an XML tag opened inside another element must be closed before the outer element is closed.

### XML Elements

XML elements can be defined as building blocks of an XML. Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

`<element-name attribute1 attribute2>`

`....content`

`</element-name>`

- **element-name** is the name of the element.

- attribute1, attribute2 are **attributes** of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as: name = "value".

**Empty Element:** An empty element is an element with no content.

**Example:** <name attribute1 attribute2.../>

### XML Elements Rules

- An element name can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (\_) and period (.).
- Names are case sensitive.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements.

### XML Attributes

Attributes are part of the XML elements. An element can have multiple unique attributes. Attribute gives more information about XML elements. To be more precise, they define properties of elements. An XML attribute is always a name-value pair.

```
<element-name attribute1 attribute2 >
    ....content....
</element-name>
```

where attribute1 and attribute2 has the following form: name = "value"

The following are the types of attributes:

- **String:** It takes any literal string as a value. CDATA is a String type. CDATA is character data. This means, any string of non-markup characters is a legal part of the attribute.
- **Tokenized:** This is more constrained type. The validity constraints noted in the grammar are applied after the attribute value is normalized.
- **Enumerated:** This has a list of predefined values in its declaration, out of which, it must assign one value. There are two types of enumerated attribute:
  - ❖ **NotationType:** It declares that an element will be referenced to a NOTATION declared somewhere else in the XML document.
  - ❖ **Enumeration:** Enumeration allows the user to define a specific list of values that the attribute value must match.

### Element Attribute Rules

An attribute name must not appear more than once in the same start-tag or empty-element tag. An attribute must be declared in the Document Type Definition (DTD) using an Attribute-List Declaration. Attribute values must not contain direct or indirect entity references to external entities. The replacement text of any entity referred to directly or indirectly in an attribute value must not contain either less than sign <.

### XML other features

#### ➤ Comments:

A comment starts with <!-- and ends with -->. Comments cannot appear before XML declaration. Comments can appear anywhere in a document. Comments must not appear within attribute values. Nested comments are not allowed.

#### ➤ Whitespaces:

Whitespace is a collection of spaces, tabs, and newlines. XML document contain two types of white spaces Significant Whitespace and Insignificant Whitespace. A **significant Whitespace** occurs within the element which contain text and markup present together.

```
<name>Adhithya Ramanan</name>
```

**Insignificant whitespace** means the space where only element content is allowed.

```
<address.category="residence">
```

### Differences between XML and HTML

XML	HTML
XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.	HTML was designed to display data with focus on how data looks.
XML provides a framework for defining markup languages.	HTML is a markup language itself.
XML is neither a programming language nor a presentation language.	HTML is a presentation language.
XML is case sensitive.	HTML is case insensitive.
XML is used basically to transport data between the application and the database.	HTML is used for designing a web-page to be rendered on the client side.
In XML custom tags can be defined and the tags are invented by the author of the XML document.	HTML has its own predefined tags

XML makes it mandatory for the user the close each tag that has been used.	HTML is not strict if the user does not use the closing tags.
XML preserve white space.	HTML does not preserve white space.
XML is about carrying information, hence it is dynamic.	HTML is about displaying data, hence it is static.

### DOCUMENT TYPE DECLARATION (DTD)

The document type declaration attaches a DTD to a document. It is a way to describe XML language precisely.

```
<!DOCTYPE element DTD identifier
[ declaration1
  declaration2
  ..... ]>
```

The DTD starts with <!DOCTYPE delimiter. An element tells the parser to parse the document from the specified root element. DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset**. The square brackets [ ] enclose an optional list of entity declarations called **Internal Subset**.

#### Internal DTD

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes.

```
<!DOCTYPE root-element [element-declarations]>
```

root-element is the name of root element and element-declarations is where the user declare the elements.

#### Internal DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)> ]>
<address> <name>Sharanya</name>
<company>abc</company>
<phone>12347890</phone> </address>
```

**Start Declaration-** Begin the XML declaration with following statement

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

**DTD-** Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE:

```
<!DOCTYPE address [
```

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

**DTD Body-** The DOCTYPE declaration is followed by body of the DTD, where elements, attributes, entities, and notations are declared.

**End Declaration** - Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (]>).

### Rules

- The document type declaration must appear at the start of the document.
- The element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

### External DTD

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

```
<!DOCTYPE root-element SYSTEM "file-name">
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address> <name>Sharanya</name>
<company>abc</company>
<phone>1234689</phone> </address>
address.dtd
<!ELEMENT address (name,company,phone)> <!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)> <!ELEMENT phone (#PCDATA)>
```

### Types of external DTD

- **System Identifiers:** A system identifier enables the user to specify the location of an external file containing DTD declarations.

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

- **Public Identifiers:** Public identifiers provide a mechanism to locate DTD resources.

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called **Formal Public Identifiers, or FPIs**.

### Advantages of DTD

- The XML processor enforces the structure, as defined in the DTD.
- The application easily accesses the document structure.
- The DTD gives hints to the XML processor—that is, it helps separate indenting from content.
- The DTD can declare default or fixed values for attributes. This might result in a smaller document.

### XML SCHEMAS

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="contact">
<xs:complexType><xs:sequence>
<xs:element name="name" type="xs:string" />
<xs:element name="company" type="xs:string" />
<xs:element name="phone" type="xs:int" /> </xs:sequence>
</xs:complexType></xs:element> </xs:schema>
```

- **Elements:** An element can be defined within an XSD as follows:

```
<xs:element name="x" type="y"/>
```

➤ **Definition Types**

- **Simple Type** - Simple type element is used only in the context of the text. Some of predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date.

**Example:** `<xs:element name="phone_number" type="xs:int" />`

- **Complex Type** - A complex type is a container for other element definitions. This allows the user to specify which child elements an element can contain and to provide some structure within the user's XML documents.

```
<xs:element name="Address">
  <xs:complexType><xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="company" type="xs:string" />
    <xs:element name="phone" type="xs:int" />
  </xs:sequence></xs:complexType>
</xs:element>
```

- **Global Types** - With global type, the user can define a single type in the user's document, which can be used by all other references.

```
<xs:element name="AddressType">
  <xs:complexType><xs:sequence>
    <xs:element name="name" type="xs:string" />
    <xs:element name="company" type="xs:string" />
  </xs:sequence></xs:complexType>
</xs:element>
```

Now let us use this type in our example as below:

```
<xs:element name="Address1">
  <xs:complexType><xs:sequence>
    <xs:element name="address" type="AddressType" />
    <xs:element name="phone1" type="xs:int" />
  </xs:sequence></xs:complexType>
```

```

</xs:element>
<xs:element name="Address2">
  <xs:complexType><xs:sequence>
    <xs:element name="address" type="AddressType" />
    <xs:element name="phone2" type="xs:int" />
  </xs:sequence></xs:complexType>
</xs:element>

```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition.

### ➤ Attributes

Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below:

```
<xs:attribute name="x" type="y"/>
```

### XML DOM

The Document Object Model (DOM) is the foundation of XML. XML documents have a hierarchy of informational units called nodes; DOM is a way of describing those nodes and the relationships between them.

### DOM

*A DOM Document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information.*

```

<!DOCTYPE html> <html><body>
<h1> DOM example </h1>
<div> <b>Name:</b><span id="name"></span><br>
<b>Company:</b><span id="company"></span><br>
<b>Phone:</b><span id="phone"></span> </div>
<script>    if (window.XMLHttpRequest)
        {
        // code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp = new XMLHttpRequest();    }

```



```

else    { // code for IE6, IE5
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");    }
xmlhttp.open("GET","/xml/address.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
document.getElementById("name").innerHTML=
xmlDoc.getElementsByTagName("name")[0].childNodes[0].nodeValue;
document.getElementById("company").innerHTML=
xmlDoc.getElementsByTagName("company")[0].childNodes[0].nodeValue;
document.getElementById("phone").innerHTML=
xmlDoc.getElementsByTagName("phone")[0].childNodes[0].nodeValue;
</script></body></html>

```

**address.xml**

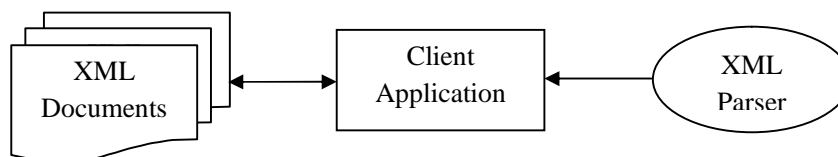
```

<?xml version="1.0"?>
<contact-info>
<name>Sharanya</name>
<company>abc</company>
<phone>(011) 123-4567</phone>
</contact-info>

```

## XML PARSERS

*XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents.*



**Fig 4.2 XML Parsers**

The goal of a parser is to transform XML into a readable code. To ease the process of parsing, some commercial products are available that facilitate the breakdown of XML document and yield more reliable results.

## VALIDATION

An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are: Well-formed XML document and Valid XML document

### ➤ Well-formed XML document

- Non DTD XML files must use the predefined character entities for amp(&), apos (single quote), gt (>), lt (<), quot (double quote).
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self- ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- Amp (&), apos (single quote), gt (>), lt (<), quot (double quote) entities other than these must be declared.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)> ]>
<address> <name>Sharanya</name>
<company>abc</company>
<phone>(011) 123-4567</phone> </address>
```

### ➤ Valid XML document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

### XSL (XML Style Sheet)

XML concentrates on the structure of the information and not its appearance. The W3C has published two recommendations for style sheets: CSS (Cascading Style Sheet) and XSL(XML Style sheet Language).XSL supports transforming the document before display. XSL would typically be used for advanced styling. XSL originally consisted of three parts:

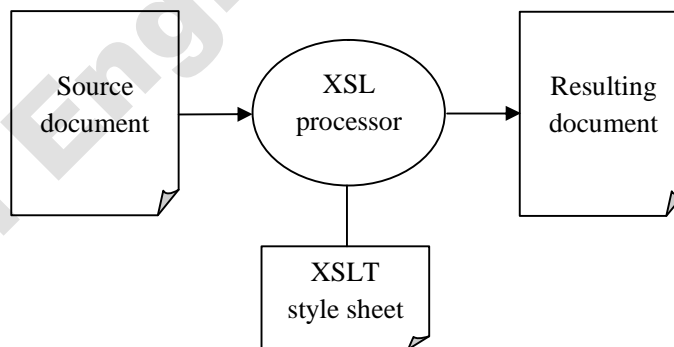
- XSLT (XSL Transformation) - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO (XSL Formatting Objects) - a language for formatting XML documents

### XSL

```
<P><B>Table of Contents</B></P> <UL>
<xsl:for-each select="article/section/title">
<LI><A><xsl:value-of select="."/></A></LI>
</xsl:for-each> </UL>
```

### XSLT (XSL Transformation)

XSLT is a language to specify transformation of XML documents. It takes an XML document and transforms it into another XML document. XSLT is an XML-related technology that is used to manipulate and transform XML documents.



**Fig 4.2 XSLT Transformation**

With XSLT, the user can take an XML document and choose the elements and values, then generate a new file with new choices. Because of XSLT's ability to change the content of an XML document, XSLT is referred to as the **stylesheet for XML**. XSLT is not limited to styling activities. Many applications require transforming documents. XSLT can be used to:

- Add elements specifically for viewing, such as add the logo or the address of the sender to an XML invoice.

- Create new content from an existing one, such as create the table of contents
- Present information with the right level of details for the reader, such as using a style sheet to present high-level information to a managerial person while using another style sheet to present more detailed technical information to the rest of the staff.
- Convert between different DTDs or different versions of a DTD, such as convert a company specific DTD to an industry standard
- Transform XML documents into HTML for backward compatibility with existing browsers.

## XSLT

XML code	XSLT code
<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;?xml-stylesheet type="text/xsl" href="class.xml"?&gt; &lt;class&gt; &lt;student&gt;Arthi&lt;/student&gt; &lt;student&gt;Ambarish&lt;/student&gt; &lt;student&gt;Anitha&lt;/student&gt; &lt;teacher&gt;Sharanya&lt;/teacher&gt; &lt;/class&gt;</pre>	<pre>&lt;?xml version="1.0" ?&gt; &lt;xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" &gt; &lt;xsl:template match="teacher"&gt; &lt;p&gt;&lt;u&gt;&lt;xsl:value-of select="."/&gt;&lt;/u&gt;&lt;/p&gt; &lt;/xsl:template&gt; &lt;xsl:template match="student"&gt; &lt;p&gt;&lt;b&gt;&lt;xsl:value-of select="."/&gt;&lt;/b&gt;&lt;/p&gt; &lt;/xsl:template&gt; &lt;xsl:template match="/"&gt; &lt;html&gt;&lt;body&gt; &lt;xsl:apply-templates/&gt; &lt;/body&gt;&lt;/html&gt; &lt;/xsl:template&gt;&lt;/xsl:stylesheet&gt;</pre>

The XML file class.xml is linked to the XSLT code by adding the xml-stylesheet reference. The XSLT code then applies its rules to transform the XML document.

- Before XSLT: classoriginal.xml
- After XSLT rules are applied: class.xml

## XSLT Syntax

### ➤ XSLT - XML Declaration

The user includes an XML declaration at the top of the XSLT documents. The attribute version defines what version of XML is used.

**Example:** `<?xml version="1.0" ?>`

**➤ XSLT - Stylesheet Root Element**

Every XSLT file must have the root element `xsl:stylesheet`. This root element has two attributes that must be included:

- `version` - the version of XSLT
- `xmlns:xsl` - the XSLT namespace, which is a URI to w3.org

**➤ XSLT - XSL: Namespace Prefix**

The root element specifies the XSL namespace. The standard form of an XSL element is: `xsl:element`

**XSLT - Stylesheet Reference**

Linking XML document to XSLT stylesheet is stylesheet reference. This is the magic step that connects XML to a XSLT file

**XSLT - xml-stylesheet**

`xml-stylesheet` is a special declaration in XML for linking XML with stylesheets. Place this after XML declaration to link the XML file to the XSLT code. `xml-stylesheet` has two attributes:

- `type`: the type of file being linked to. We will be using the value `text/xsl` to specify XSLT.
- `href`: the location of the file. If the user saved the user XSLT and XML file in the same directory, the user can simply use the XSLT filename.

Make sure that both XSLT and XML file are in the same directory.

**Reference**

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>    <student>Arun</student>
           <student>Divya</student>
           <teacher>Sharanya</teacher> </class>
```

**XSLT: XSL Template**

The purpose of XSLT is to help transform an XML document into something new. To transform an XML document, XSLT must be able to do two things well:

- Find information in the XML document.
- Add additional text and/or data.

Both of these items are taken care of with the very important XSL element `xsl:template`.

### XSLT - `xsl:template Match Attribute`

To find information in an XML document use `xsl:template`'s `match` attribute. It is in this attribute the knowledge of XPath is used to find information in the XML document. In previous example, to find student elements, we would set the `match` attribute to a simple XPath expression: `student`.

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="student">
    Found a learner!
  </xsl:template>
</stylesheet>
```

### XSLT - `xsl:apply-templates`

The `xsl:apply-templates` element to be more selective of the XML data.

#### ➤ XSLT - Remove Unwanted Text

The following attributes are used to remove unwanted text:

- `select` attribute: lets the user choose specific child elements
- `xsl:apply-templates`: to decide when and where the `xsl:template` elements are used

### XSLT - Remove Unwanted Children

We could use the `select` attribute to select specific child elements. To do this, we need a new `xsl:template` that matches our XML document's root element, `class`. We can then pick the child student using the `select` attribute. Here's the XSLT code to get the job done.

#### apply templates

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="class">
    <xsl:apply-templates select="student"/>
  </xsl:template>
  <xsl:template match="student">
```

<pre>Found a learner! &lt;/xsl:template&gt;&lt;/xsl:stylesheet&gt;</pre>
<pre>Found a learner! Found a learner! Found a learner!</pre>

The XSLT processor begins at the root element when looking for template matches. Because we have a match for the root element, class, the code we just added is used first.

```
xsl:apply-templates
```

In our template that matched class, we use xsl:apply-templates which will check for template matches on all the children of class. The children of class in our XML document are student and teacher.

```
xsl:apply-templates select="student"
```

To have the teacher element, "Sharanya," ignored, we use the select attribute of xsl:apply-templates to specify only student children.

The XSLT processor then goes searching templates that only match student elements.

```
xsl:template match="student"
```

The processor finds the only other template in our XSLT, which prints out, "Found a learner!" for each student element in the XML document. XSLT finds three students, so "Found a learner!" is displayed three times.

### **XSLT - Well-Formed Output**

To obtain well formed output remove the root element in an XSLT template and inserting a new root element for the output. To do this, we are going to need to add an <html> (root element) tag, a <body> tag, and maybe some <p> tags.

### **XSLT - Replacing the Old Root Element**

In the template that matches the original root element, we will insert the <html> tag to be the output's root element. We can also put the <body> tag there. In the template that matches the student elements, we can insert a <p> tag to make a separate paragraph for each student.

### **Replacing root**

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="class"> <html><body>
<xsl:apply-templates select="student"/> </body></html> </xsl:template>
```

```

<xsl:template match="student">
<p> Found a learner!</p> </xsl:template></xsl:stylesheet>

<html><body>
<p>Found a learner!</p>
<p>Found a learner!</p>
<p>Found a learner!</p>
</body></html>

```

#### 4.4 NEWSFEEDS

News feeds are an example of automated syndication. News feed technologies allow information to be automatically provided and updated on Web sites, emailed to users, etc. As the name implies news feeds are normally used to provide news; however the technology can be used to syndicate a wide range of information.

The **BBC ticker** is an example of a news feed application. A major limitation with this approach is that the ticker can only be used with information provided by the BBC. The RSS (Really Simple Syndication) standard was developed as an open standard for news syndication, allowing applications to display news supplied by any RSS provider.

##### RSS (Really Simple Syndication)

*RSS is an XML dialect used to publish frequently updated content, such as blog posts or news headlines.*

It is a way to easily distribute a list of headlines, update notices, and sometimes content to a wide number of people. It is used by computer programs that organize those headlines and notices for easy reading.

The content feed is identified by a unique URI, and this URI is used by an RSS Reader application to retrieve and display the content feed from a web site. An RSS feed can contain one or more images or items. An item can be a synopsis of an article with a link to the full article or the entire article itself. An RSS has a well-defined structure. Some Web APIs use RSS as the data format returned when a request is made.

##### Working of RSS

RSS works by having the website author maintain a list of notifications on their website in a standard way. This list of notifications is called an **RSS Feed**. People who are interested in finding out the latest headlines or changes can check this list. Special computer programs called **RSS aggregators** have been developed that automatically access the RSS feeds of websites of user's interest and organize the results.

Producing an RSS feed is very simple and hundreds of thousands of websites now provide this feature, including major news organizations like the New York Times, the BBC, and Reuters, as well as many weblogs.



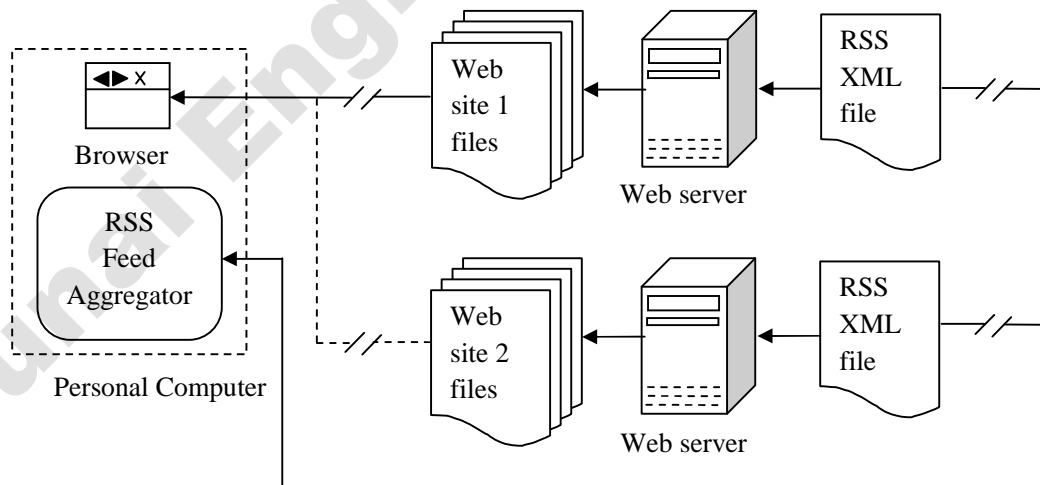
RSS provides very basic information to do its notification. It is made up of a list of items presented in order from newest to oldest. Each item usually consists of a simple title describing the item along with a more complete description and a link to a web page with the actual information being described. Sometimes this description is the full information the user want to read (such as the content of a weblog post) and sometimes it is just a summary.

### Creating RSS feed

The special XML-format file that makes up an RSS feed is usually created in one of a variety of ways. Most large news websites and most weblogs are maintained using special **content management** programs. Authors add their stories and postings to the website by interacting with those programs and then use the program's publish facility to create the HTML files that make up the website. Those programs often also can update the RSS feed XML file at the same time, adding an item referring to the new story or post, and removing less recent items.

Blog creation tools like Blogger, LiveJournal, Movable Type, and Radio automatically create feeds. Websites that are produced in a more custom manner, such as with Macromedia Dreamweaver or a simple text editor, usually do not automatically create RSS feeds. Authors of such websites either maintain the XML files by hand, just as they do the website itself, or use a tool such as Software Garden, Inc.'s ListGarden program to maintain it.

There are also services that periodically requested websites themselves and try to automatically determine changes (this is most reliable for websites with a somewhat regular news-like format), or that let the users create RSS feed XML files that are hosted by that service provider.



**Fig 4.3: Communication between websites, RSS feed and PC**

In the above diagram, a web browser being used to read first Web Site 1 over the Internet and then Web Site 2. It also shows the RSS feed XML files for both websites being monitored simultaneously by an RSS Feed Aggregator.

### ➤ Sections of an RSS file

Apart from the root element there are four main sections of the RSS file. These are the channel, image, item, and text input sections. In practical use, the channel and item elements are requirements for any useful RSS file, while the image and text input are optional.

#### The channel section

The channel element contains metadata that describe the channel itself, telling what the channel is and who created it. The channel is a required element that includes the name of the channel, its description, its language, and a URL. The URL is normally used to point to the channel's source of information.

#### Channel element

```
<channel><title>MozillaZine</title>
<link>http://www.mozillazine.org</link>
<description>The user source for Mozilla news, advocacy, interviews, builds, and more!
</description>
<language>en-us</language> </channel>
```

The title can be treated as a headline link with the description following. The **Channel Language definition** allow aggregators to filter news feeds and gives the rendering software the information necessary to display the language properly. The `</channel>` tag is used after all the channel elements to close the channel. As RSS conforms to XML specs, the element must be well formed; it requires the closing tag.

#### The image section

The image element is an optional element that is usually used to include the logo of the channel provider. The default size for the image is 88 pixels wide by 31 pixels high, but it can be enlarged to 144 pixels wide by 400 pixels wide.

#### Image element

```
<image><title>MozillaZine</title>
<url>http://www.mozillazine.org/image/mynetscape88.gif</url>
<link>http://www.mozillazine.org</link>
<width>88</width>
<height>31</height> </image>
```

The image's title, URL, link, width, and height tags allow renderers to translate the file into HTML. The title tag is normally used for the image's ALT text.

### The items

Items form the dynamic part of an RSS file. While channel, image, and text input elements create the channel's identity and typically stay the same over long periods of time, channel items are rendered as news headlines, and the channel's value depends on their changing fairly frequently.

### Item element

```
<item><title>Java2 in Navigator 5?</title>
<link>http://www.mozillazine.org/talkback.html?article=607</link>
<description>Will Java2 be an integrated part of Navigator 5?
  Read more about it in this discussion...</description> </item>
```

Fifteen items are allowed in a channel. Titles should be less than 100 characters, while descriptions should be under 500 characters. The item title is normally rendered as a headline that links to the full article whose URL is provided by the item link. The item description is commonly used for either a summary of the article's content or for commentary on the article.

News feed channels use the description to highlight the content of news articles, usually on the channel owner's site, and Web log channels use the description to provide commentary on a variety of content, often on third-party sites.

### The text input

The text input area is an optional element, with only one allowed per channel. This lets the user respond to the channel.

```
<textinput><title>Send</title>
<description>Comments about MozillaZine?</description>
<name>responseText</name>
<link>http://www.mozillazine.org/cgi-bin/sampleonly.cgi</link> </textinput>
```

The title is normally rendered as the label of the form's submit button, and the description as the text displayed before or above the input field. The text input name is supplied along with the contents of the text field when the submit button is clicked.

## 4.5 ATOM

*Atom is a syndication data format like RSS, as well as a publishing protocol. The Atom data format uses XML syntax with one or more entry elements containing the full and/or summary content.*

The **Atom Publishing Protocol** (APP) defines a hierarchy for organizing published content (services, workspaces, collections, and resources) and uses the HTTP Get, Post, Put, and Delete methods for retrieving, creating, deleting, and editing published content. Atom's use of HTTP's built-in methods and XML as a data format is in the spirit of a RESTful web services implementation.

Atom was designed to be a universal publishing standard for blogs and other Web sites where content is updated frequently. Users visiting a Web site with an Atom feed can discover a file described as "atom.xml" in the URL that can be copied and pasted into an aggregator to subscribe to the feed.

Atom was originally developed as an alternative to RSS 2.0, the standard developed by Dave Winer and copyrighted by Harvard University, as a means of improving perceived shortcomings of the RSS format by the blogging community.

### **Features of ATOM**

- Atom was developed to be vendor neutral and freely extensible by any user; it is an open standard.
- Atom lies within an XML-namespace.
- Atom includes auto discovery, allowing feed aggregators to automatically detect the presence of a feed.
- Atom differentiates between relative and non-relative URIs.
- Atom has separate summary and content elements.
- Atom explicitly labels a payload as plaintext or HTML.
- Each entry has a globally unique ID.

## REVIEW QUESTIONS

### PART-A

#### 1. Define PHP.

The PHP Hypertext Pre-processor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases.

#### 2. List the Features of PHP:

- ❖ PHP is a server side scripting language that is embedded in HTML
- ❖ PHP was originally developed by the Danish Greenlander RasmusLerdorf, and was subsequently developed as open source.
- ❖ It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- ❖ It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- ❖ PHP supports a large number of protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- ❖ PHP language tries to be as forgiving as possible.
- ❖ PHP syntax is C-Like.

#### 3. Give the applications of PHP

- ❖ PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- ❖ PHP can handle forms, i.e. gather data from files, save data to a file, through email the user can send data, return data to the user.
- ❖ The user can add, delete, and modify elements within the database through PHP.
- ❖ They can access cookies variables and set cookies.
- ❖ Using PHP, the user can restrict users to access some pages of the website.
- ❖ It can encrypt data.

#### 4. Explain the working of PHP

When the client requests a PHP page residing on the server, the server first performs the operations mentioned by the PHP code of the page. Then it sends the output of the PHP page in HTML format. So when the user views the source code of the page, it will be full of HTML tags. All the work is done at the server side.

**5. List the rules of PHP**

- ❖ PHP is white space insensitive
- ❖ PHP is case sensitive
- ❖ Statements are expressions terminated by semicolons
- ❖ Expressions are combinations of tokens
- ❖ Braces make blocks

**6. Differentiate between constants and variables in PHP**

Constants in PHP	Variables in PHP
No \$ sign before constants.	\$ sign is present before variables.
Constants are defined using define().	Variables are defined using assignment statement.
Constants may be defined and accessed anywhere without any regard.	Variables follow certain scope.
Constants cannot be redefined or undefined.	They can be redefined.

**7. Give the pre-defined constants in PHP.**

Name	Description
__LINE__	The current line number of the file.
__FILE__	The full path and filename of the file. If used inside an include, the name of the included file is returned. Since PHP 4.0.2, __FILE__ always contains an absolute path whereas in older versions it contained relative path under some circumstances
__FUNCTION__	The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__CLASS__	The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__METHOD__	The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive).

**8. Differentiate between echo and print statement**

Echo	Print
This does not have return value.	This has a return value of 1.
This cannot be used in expressions.	This can be used in expressions.
This can take multiple parameters.	This can take only one parameter.
This is slightly faster than print.	This is comparatively slower.

**9. Define regular expressions.**

Regular expressions are nothing more than a sequence or pattern of characters. They provide the foundation for pattern-matching functionality.

**10. What is XML?**

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable.

**11. List the features of XML**

- ❖ XML simplifies the creation of HTML documents for large web sites.
- ❖ XML can be used to exchange the information between organizations and systems.
- ❖ XML can be used for offloading and reloading of databases.
- ❖ XML can be used to store and arrange the data, which can customize the user data handling needs.
- ❖ XML can easily be merged with style sheets to create almost any desired output.
- ❖ Any type of data can be expressed as an XML document
- ❖ It has syndicated content, where content is being made available to different web sites.
- ❖ It suits well for electronic commerce applications where different organizations collaborate to serve a customer.
- ❖ It supports scientific applications with new markup languages for mathematical and chemical formulas.
- ❖ It also supports electronic books with new markup languages to express rights and ownership.

- ❖ XML could also be used in handheld devices and smart phones with new markup languages optimized for these devices.

## 12. Define Entity References.

An entity reference contains a name between the start and the end delimiters. The entity reference is replaced by the content of the entity.

## 13. Define Character References.

A letter is replaced by its Unicode character code. Character references that start with `&#x` provides a hexadecimal representation of the character code.

## 14. List the rules for XML declaration

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: version, encoding and standalone.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. `<?xml>`

## 15. Differentiate between XML and HTML

XML	HTML
XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.	HTML was designed to display data with focus on how data looks.
XML provides a framework for defining markup languages.	HTML is a markup language itself.
XML is neither a programming language nor a presentation language.	HTML is a presentation language.
XML is case sensitive.	HTML is case insensitive.
XML is used basically to transport data between the application and the database.	HTML is used for designing a web-page to be rendered on the client side.



In XML custom tags can be defined and the tags are invented by the author of the XML document.	HTML has its own predefined tags
XML makes it mandatory for the user to close each tag that has been used.	HTML is not strict if the user does not use the closing tags.
XML preserve white space.	HTML does not preserve white space.
XML is about carrying information, hence it is dynamic.	HTML is about displaying data, hence it is static.

### 16. What is Internal DTD?

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes.

```
<!DOCTYPE root-element [element-declarations]>
```

root-element is the name of root element and element-declarations is where the user declare the elements.

### 17. What is External DTD?

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

### 18. List the advantages of DTD

- The XML processor enforces the structure, as defined in the DTD.
- The application easily accesses the document structure.
- The DTD gives hints to the XML processor—that is, it helps separate indenting from content.
- The DTD can declare default or fixed values for attributes. This might result in a smaller document.

### 19. Define XML Schema.

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

## **20. What is DOM?**

A DOM Document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information.

## **21. What is the role of XML parser?**

XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents.

## **22. What is XSL (XML Style Sheet)?**

XML concentrates on the structure of the information and not its appearance. The W3C has published two recommendations for style sheets: CSS (Cascading Style Sheet) and XSL(XML Style sheet Language).XSL supports transforming the document before display. XSL would typically be used for advanced styling. XSL originally consisted of three parts:

- XSLT (XSL Transformation) - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO (XSL Formatting Objects) - a language for formatting XML documents

## **23. What is XSLT (XSL Transformation)?**

XSLT is a language to specify transformation of XML documents. It takes an XML document and transforms it into another XML document. XSLT is an XML-related technology that is used to manipulate and transform XML documents.

## **24. What is meant by news feed?**

News feeds are an example of automated syndication. News feed technologies allow information to be automatically provided and updated on Web sites, emailed to users, etc. As the name implies news feeds are normally used to provide news; however the technology can be used to syndicate a wide range of information.

## **25. Define RSS.**

RSS is an XML dialect used to publish frequently updated content, such as blog posts or news headlines.

## **26. Define channel language definition.**

The Channel Language definition allow aggregators to filter news feeds and gives the rendering software the information necessary to display the language properly.

**27. What is ATOM?**

Atom is a syndication data format like RSS, as well as a publishing protocol. The Atom data format uses XML syntax with one or more entry elements containing the full and/or summary content.

**28. What is APP?**

The Atom Publishing Protocol (APP) defines a hierarchy for organizing published content (services, workspaces, collections, and resources) and uses the HTTP Get, Post, Put, and Delete methods for retrieving, creating, deleting, and editing published content.

**29. List the features of ATOM.**

- Atom was developed to be vendor neutral and freely extensible by any user; it is an open standard.
- Atom lies within an XML-namespace.
- Atom includes auto discovery, allowing feed aggregators to automatically detect the presence of a feed.
- Atom differentiates between relative and non-relative URIs.
- Atom has separate summary and content elements.
- Atom explicitly labels a payload as plaintext or HTML.
- Each entry has a globally unique ID.

**PART-B**

1. Give the various syntactical structure of PHP.
2. Explain about PHP control structures.
3. Describe functions in PHP.
4. List the functionalities of data, time, error handling, file manipulation, math and string functions in PHP.
5. Write in detail about PHP cookies.
6. Explain regular expressions in PHP.
7. Describe data base connectivity in PHP.
8. Explain XML.

9. Write about DTD and XML schemas.
10. Describe document object model.
11. Elaborate about XML parsers.
12. Write about XSLT and XSL.
13. Brief about newsfeed and RSS.
14. Elaborate ATOM.

Arunai Engineering College

---

---

# INTRODUCTION TO AJAX and WEB SERVICES

---

---

## 5.1 INTRODUCTION TO AJAX

The underlying technologies behind classic Web applications (HTML) are simple and straight forward. The classic web pages has very little intelligence and lack dynamic and interactive behaviors. Changes in today's web pages are brought by AJAX (Asynchronous JavaScript and XML)

*Ajax refers to a set of technologies and techniques that allow web pages be interactive like desktop applications.*

AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display. Conventional web applications transmit information to and from the sever using synchronous requests. It means the user fill out a form, hit submit, and get directed to a new page with new information from the server. With AJAX, when the user hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. The user would never know that anything was even transmitted to the server.

XML is commonly used as the format for receiving server data. AJAX is a web browser technology independent of web server software. A user can continue to use the application while the client program requests information from the server in the background. In AJAX, clicking is not required; mouse movement is a sufficient event trigger. It is a data-driven technology. AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages. The technologies that support AJAX are: JavaScript, DOM, CSS and XMLHttpRequest.

AJAX is based on the following open standards:

- ❖ Browser-based presentation using HTML and Cascading Style Sheets (CSS).

## 5.2 Introduction to AJAX to Web Services

- ❖ Data is stored in XML format and fetched from the server.
- ❖ Behind-the-scenes data fetching is done using XMLHttpRequest objects in the browser.
- ❖ JavaScript to make everything happen.

### Asynchronous nature of AJAX

Asynchronous in AJAX means that the script will send a request to the server, and continue the execution without waiting for the reply. As soon as reply is received a browser event is fired, which in turn allows the script to execute associated actions. The client and the server are asynchronous.

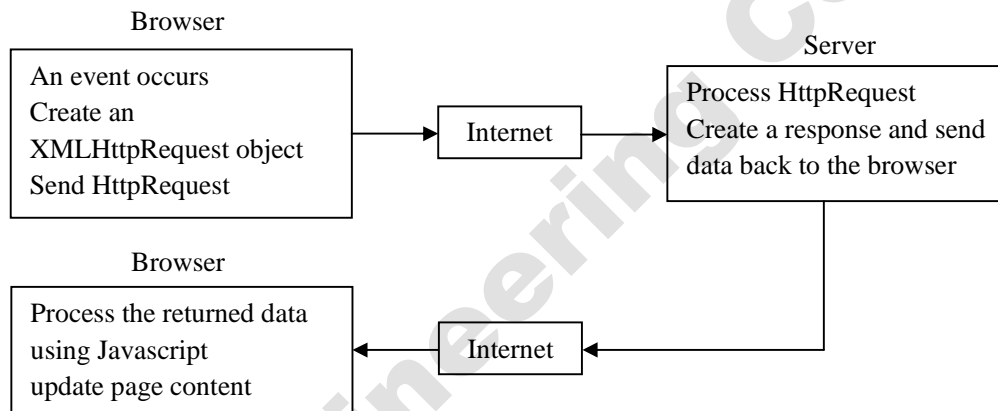
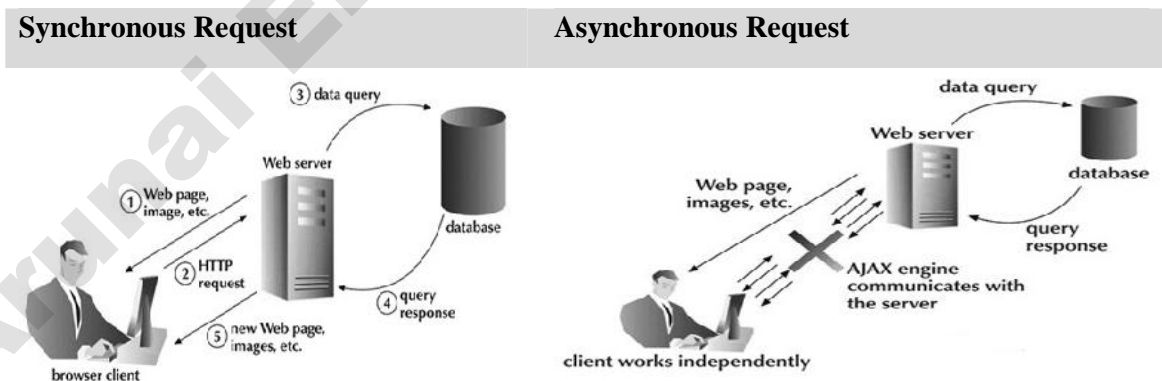


Fig 5.1: Working of AJAX



## 5.2 CLIENT SERVER ARCHITECTURE

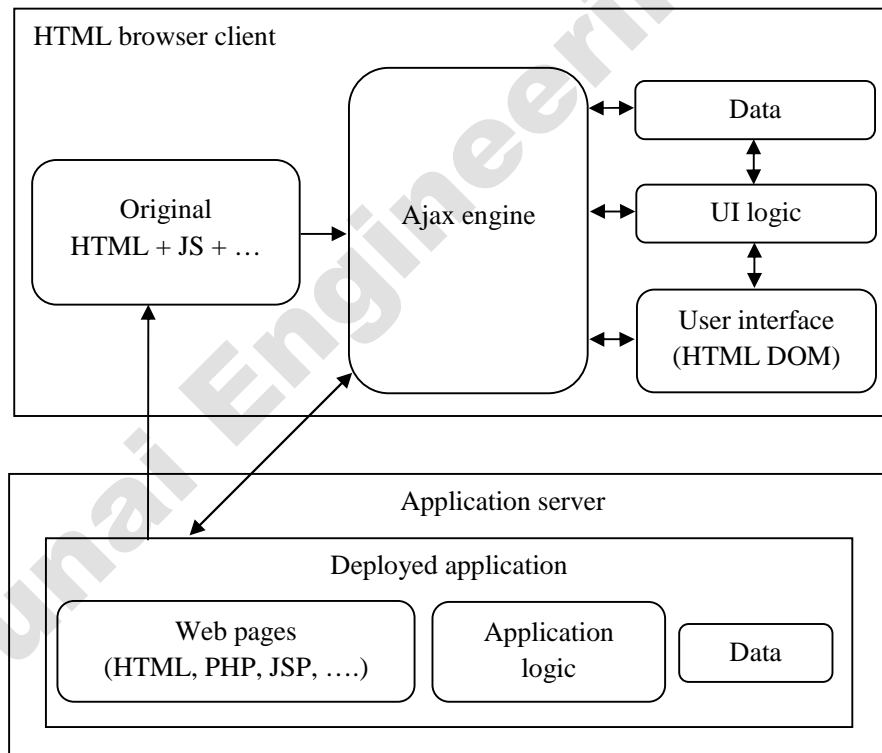
The Ajax provides a rich and diverse set of product that range from hundreds of suppliers. This diversity provides IT managers and Web developers with the ability to choose the optimal architectural approach and best products among multiple vendors.

Most Ajax technologies transform a platform-independent definition of the application into the appropriate HTML and JavaScript content that is then processed by the browser to deliver a rich user experience. Some Ajax designs perform most of their transformations on the client. Others perform transformations on the server.

### Client side vs server side transformations

#### ➤ Client-side Ajax transformations

- With client-side Ajax, the Ajax engine runs on the client.
- The server delivers Web content (HTML, CSS, JavaScript, etc.) which is processed by the client-side Ajax engine into revised Web content.



**Fig 5.2 AJAX Client transformations**

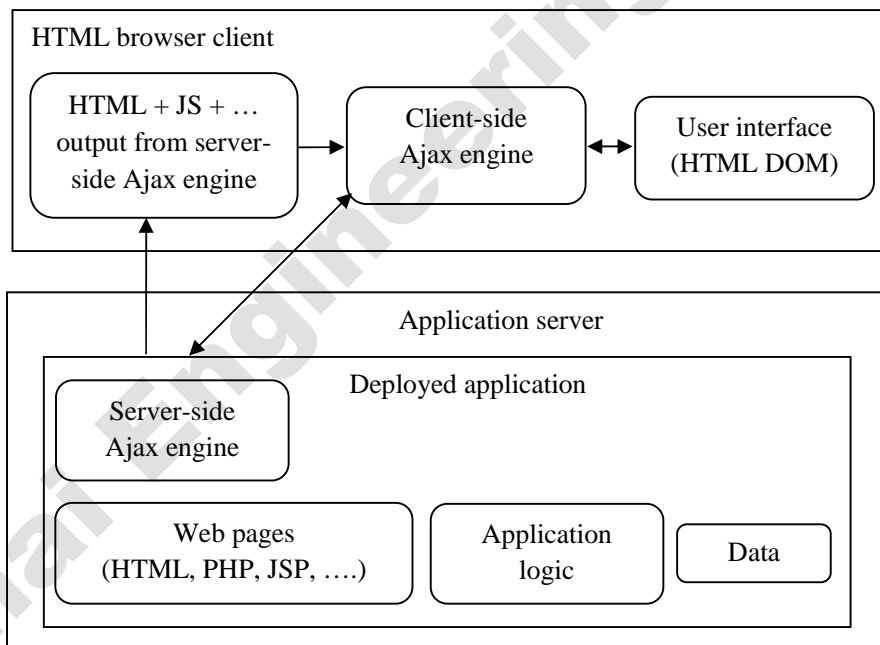
- The browser renders the revised HTML/etc. content that comes out of the Ajax engine.

- With this architecture, the application development team typically provides the following server-side components: Web pages (e.g., \*.html, \*.php, \*.jsp, \*.asp), application logic (e.g., Java) and data management (e.g., via a SQL database and/or Web Services)
- The client-side component includes client-side user interface logic, such as event handlers.
- The advantage of this option is the independence from the server side technology.
- The server code creates and serves the page and responds to the client's asynchronous requests.

This way either side can be swapped with another implementation approach.

➤ **Server-side transformations**

- For server-side Ajax, an Ajax server component performs most or all of the Ajax transformations.



**Fig 5.3 AJAX Server transformations**

- The server component generates the necessary Web content (HTML, CSS, JavaScript, etc.) to deliver the desired user experience.
- The server-side Ajax toolkit downloads its own client-side Ajax library which communicates directly with the toolkit's server-side Ajax component.



- With this architecture, the application development team typically only provides server-side components (Web pages, application logic, and data management) and entrusts client-side logic to the Ajax toolkit.
- The main benefit of this approach is that it allows the use of server-side languages for debugging, editing, and refactoring tools with which developers are already familiar
- The disadvantage of this approach is the dependence on a particular server-side technology.
- As a general rule, server-side Ajax frameworks expect application code to be written in the server-side language.
- These frameworks typically hide all the JavaScript that runs in the browser inside widgets, including their events.

## Single DOM vs Dual DOM

### Single DOM

Some Ajax runtime toolkits use a Single-DOM approach where the toolkit uses the browser's DOM for both any original source markup and any HTML+JavaScript that results from the toolkit's Ajax-to-HTML transformation logic. This is same as Fig 5.2 In the above example, the developer is using tree widgets from an Ajax runtime library. The original HTML markup (un-shaded) and the additional HTML markup inserted by the Ajax toolkit (shaded) is given in the left. The DOM objects that correspond to the elements in the HTML markup (e.g., the DOM object that represents a particular <div> element), where JavaScript/DOM objects from unshaded objects correspond to original HTML markup and shaded objects are ones that have been inserted by the Ajax toolkit is given in the right.

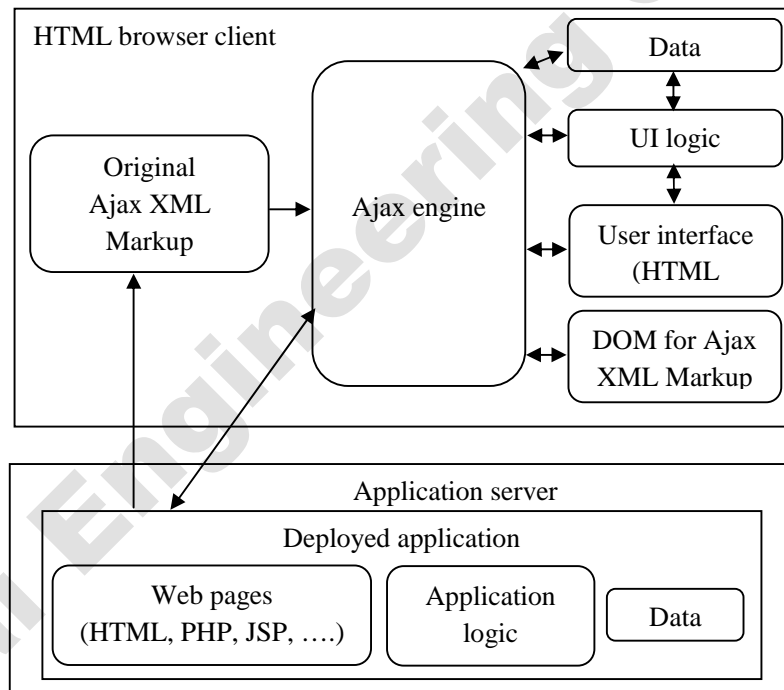
<u>Ajax source code</u>	<u>Corresponding JavaScript objects</u>
<code>&lt;html&gt;</code>	[Object]s for window and document
<code>&lt;head&gt;...&lt;/head&gt;</code>	[Private data]
<code>&lt;body...&gt;</code>	[Object] for html
<code>&lt;div class="abc:treeWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes</code>	[Object] for body
<code>&lt;div class="abc:treeWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes</code>	[Object] for div
<code>&lt;div class="abc:treeItemWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes&lt;/div&gt;</code>	[Object] for div
<code>&lt;div class="abc:treeWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes&lt;/div&gt;</code>	[Object] for div
<code>&lt;div class="abc:treeItemWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes&lt;/div&gt;</code>	[Object] for div
<code>&lt;div class="abc:treeItemWidget"&gt;</code>	[Private data]
<code>Additional rendering elements and attributes&lt;/div&gt;</code>	[Object] for div
<code>&lt;/div"&gt;</code>	[Private data]
<code>&lt;/div&gt;</code>	
<code>&lt;/div&gt;</code>	
<code>&lt;/body&gt;</code>	
<code>&lt;/html&gt;</code>	

Fig 5.4 Manipulation of Single DOM

Typically, the Ajax toolkit inserts various rendering constructs such as <img>, <div>, and <p> elements, inline within the original HTML markup (e.g., adding child elements to an existing <div> element), thereby providing the various graphics and text necessary to produce the desired visual representation for the tree widgets. The shaded sections on the right reflect the private data that Ajax libraries typically add to various DOM and JavaScript objects in order to store private data, such as run-time state information. The Single-DOM approach is particularly well-suited for situations where the developer is adding Ajax capability within non-Ajax DHTML application.

### Dual DOM

Other Ajax runtime toolkits adopt a Dual-DOM approach that separates the data for the Ajax-related markup into an Ajax DOM tree that is kept separate from the original Browser DOM tree. The Dual-DOM approach has two types:



**Fig 5.5 Dual DOM**

With **Client-side Dual-DOM**, the second DOM tree typically consists of a separate tree of JavaScript objects. With **Server-side Dual-DOM**, the second DOM tree is stored on the server.

<u>Ajax source code</u>	<u>Corresponding JavaScript objects</u>
<pre> &lt;html&gt; &lt;head&gt;...&lt;/head&gt; &lt;body...&gt;   &lt;script...&gt;     abc.load("myapp.abc");   &lt;/script&gt;   &lt;div id="abctarget"&gt;     <i>Large tree of generated elements and attributes</i>   &lt;/div&gt; &lt;/body&gt; &lt;/html&gt; </pre>	<pre> [Object]s for window and document <i>[Private data - second DOM tree for myapp.abc can be attached here]</i> [Object] for html [Object] for body [Object] for div </pre>
<pre> ...from myapp.abc... &lt;abc:treeWidget...&gt;   &lt;abc:treeWidget...&gt;     &lt;abc:treeItemWidget...&gt;     &lt;abc:treeWidget...&gt;       &lt;abc:treeItemWidget...&gt;       &lt;abc:treeItemWidget...&gt;     &lt;/abc:treeWidget...&gt;   &lt;/abc:treeWidget...&gt; &lt;/abc:treeWidget...&gt; </pre>	

**Fig 5.6 Manipulation of Dual DOM**

There are two DOMs : the **HTML DOM and the XML DOM** corresponding to the Ajax-specific XML markup for the user interface elements.

The above example shows a separate file, "myapp.abc", which contains the user interface definition for the tree widgets, which in this case are to be placed into the HTML tree inside the <div> element with id="abctarget". Even though the example shows the use of a separate file, some Dual-DOM Ajax runtime libraries support inline XML. In either case, a Dual-DOM Ajax runtime library builds a separate DOM tree, typically using its own XML parser rather than relying on the browser's HTML parser.

Sometimes the separate DOM tree is attached to the 'window' or 'document' objects. With this approach, in model view controller (MVC) terms, the Ajax DOM can be thought of as the model, the Browser DOM as the generated view, and the Ajax runtime toolkit as the controller. It is usually necessary to establish bidirectional event listeners between the Ajax DOM and the Browser DOM in order to maintain synchronization. Sometimes having a separate Ajax DOM enables a more complete set of XML and DOM support, such as full support for XML Namespaces, than is possible in the Browser DOM.

### **Dual-DOM (server-side)**

Some Ajax technologies combine server-side Ajax transformations with a Dual-DOM approach. The key difference between Server-side Dual-DOM and Client-side Dual-DOM is that, with Server-side Dual-DOM, the Ajax DOM and most user interface logic is managed on the server. In this scenario, the primary job of the client Ajax engine is to reflect back to the server any interaction state changes, deferring data handling, UI state management and UI update logic to the server. Server-side Dual-DOM enables tight application integration with server-side development technologies such as Java Server Faces (JSF).

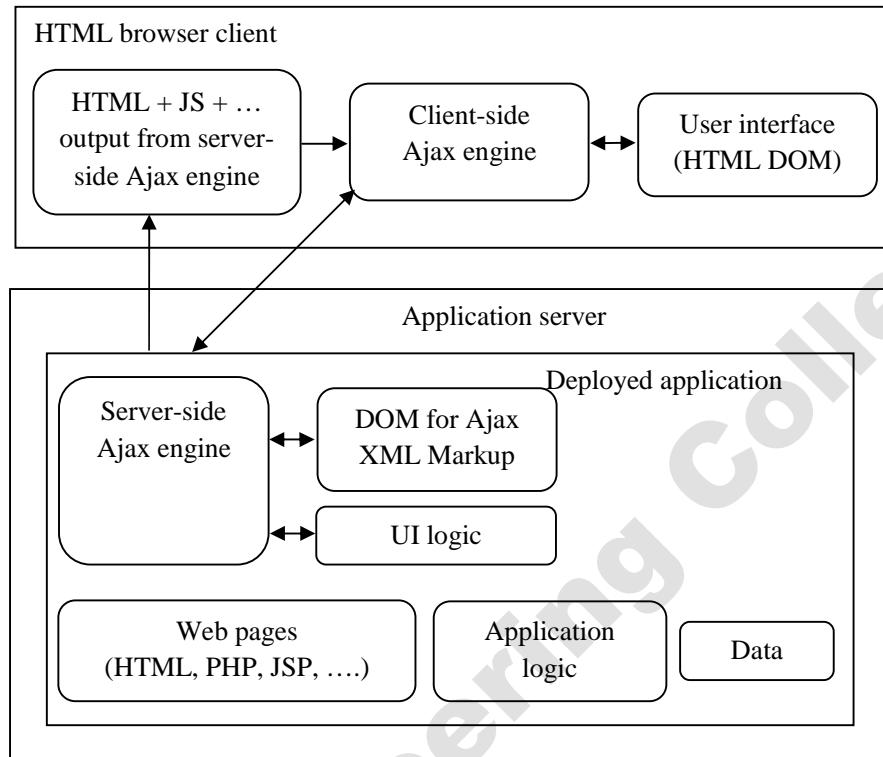


Fig 5.7 Server side Dual DOM

### 5.3 XMLHttpRequestObject and CALLBACK()

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. **XMLHttpRequest (XHR)** is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side. The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

#### Creating an XMLHttpRequest Object

##### Syntax:

```
variable=new XMLHttpRequest(); (new version)
```

```
variable=new ActiveXObject("Microsoft.XMLHTTP"); (old version)
```

#### Processing Requests in AJAX

The following are the sequence of operations request is initiated:

- ❖ A client event occurs.

- ❖ An XMLHttpRequest object is created.
- ❖ The XMLHttpRequest object is configured.
- ❖ The XMLHttpRequest object makes an asynchronous request to the Webservice.
- ❖ The Webservice returns the result containing XML document.
- ❖ The XMLHttpRequest object calls the callback() function and processes the result.
- ❖ The HTML DOM is updated.

➤ **A client event occurs:**

- A JavaScript function is called as the result of an event.
- **Example:** validateUserId() JavaScript function is mapped as an event handler to an onkeyup event on input form field whose id is set to "userid".

```
<input type="text" size="20" id="userid" name="id" onkeyup="validateUserId();">
```

➤ **XMLHttpRequest object is created**

```
var ajaxRequest; // AJAX variable
function ajaxFunction()
{
  try
  { // This code is for browsers Opera 8.0+, Firefox, Safari
    ajaxRequest = new XMLHttpRequest();
  }
  catch (e)
  { // Internet Explorer Browsers
    Try {
      ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e) {
      try {
        ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e) {
        // Something went wrong
        alert("Your browser broke");
        return false;
      }
    }
  }
}
```

➤ **The XMLHttpRequest object is configured**

```
function validateUserId()
{
    ajaxFunction(); // Here processRequest() is the callback function.
    ajaxRequest.onreadystatechange = processRequest;
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    ajaxRequest.open("GET", url, true);
    ajaxRequest.send(null); }
}
```

➤ **Making an asynchronous request to the Webserver**

This is done using the XMLHttpRequest object ajaxRequest. Assume that the user enters Sona in the userid box, then in the above request, the URL is set to "validate?id=Sona".

➤ **Webserver Returns the Result Containing XML Document**

Server-side script is implemented as follows:

- Get a request from the client.
- Parse the input from the client.
- Do required processing.
- Send the output to the client.

If we assume that the user is going to write a servlet, then:

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

```
{
    String targetId = request.getParameter("id");
    if ((targetId != null) && !accounts.containsKey(targetId.trim()))
    {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("true"); }
    else {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("false"); } }
}
```

➤ **Callback Function processRequest() is Called**

The callback function is responsible for checking the progress of requests, identifying the result of the request and handling data returned from the server. Callback functions also serve as delegators, handing off to other areas of the application code. The XMLHttpRequest object was configured to call the processRequest() function when there is a state change to the readyState of the XMLHttpRequest object. Now this function will receive the result from the server and will do the required processing. As in the following example, it sets a variable message on true or false based on the returned value from the Webserver.

```
function processRequest()
{
  if (req.readyState == 4)
  {
    if (req.status == 200)
    {
      var message = ...;
      ...
    }
  }
}
```

➤ **The HTML DOM is updated.**

This is the final step and in this step, the HTML page will be updated. It happens in the following way:

- JavaScript gets a reference to any element in a page using DOM API.
- The recommended way to gain a reference to an element is to call.  

```
document.getElementById("userIdMessage"), // userIdMessage is ID attribute
// of an element appearing in the HTML document
```
- JavaScript may now be used to modify the element's attributes; modify the element's style properties; or add, remove, or modify the child elements.

```
<script type="text/javascript">
<!-- function setMessageUsingDOM(message)
{
  var userMessageElement = document.getElementById("userIdMessage");
  var messageText;
  if (message == "false")
  {
    userMessageElement.style.color = "red";
    messageText = "Invalid User Id";
  }
  else {
```

## 5.12 Introduction to AJAX to Web Services

---

```
userMessageElement.style.color = "green";
messageText = "Valid User Id"; }
varmessageBody = document.createTextNode(messageText);
if (userMessageElement.childNodes[0])
{ userMessageElement.replaceChild(messageBody, userMessageElement.childNodes[0]);
}
Else { userMessageElement.appendChild(messageBody); } }
</script> <body> <div id="userIdMessage"><div> </body>
```

### XMLHttpRequest Methods

Method	Description
abort()	Cancels the current request.
getAllResponseHeaders()	Returns the complete set of HTTP headers as a string.
getResponseHeader( headerName )	Returns the value of the specified HTTP header
open( method, URL ) open( method, URL, async ) open( method, URL, async, userName ) open( method, URL, async, userName, password )	Specifies the method, URL, and other optional attributes of a request. The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" may be possible. The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
send( content )	Sends the request.
setRequestHeader( label, value )	Adds a label/value pair to the HTTP header to be sent.

### XMLHttpRequest Properties

- **onreadystatechange:** An event handler for an event that fires at every state change.
- **readyState:** The readyState property defines the current state of the XMLHttpRequest object.



State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

- readyState = 0 After the user have created the XMLHttpRequest object, but before the call of the open() method.
- readyState = 1 After the user have called the open() method, but before the call of send().
- readyState = 2 After the user have called send().
- readyState = 3 After the browser has established a communication with the server, but before the server has completed the response.
- readyState = 4 After the request has been completed, and the response data has been completely received from the server.
  - responseText:Returns the response as a string.
  - responseXML:Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
  - Status:Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
  - statusText:Returns the status as a string (e.g., "Not Found" or "OK").

## 5.4 WEB SERVICES

*A web service is a collection of open protocols and standards used for exchanging data between applications or systems.*

Web services are a technology, a process, and a phenomenon. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML. XML is used to encode all communications to a web service. They include programs, objects, messages, or documents. They are available over the Internet or private (intranet) networks. They are not tied to any one operating system or programming language. They are self-describing through a common XML grammar They are discoverable through a simple find mechanism

### Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components:

- Java Web Services
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

### Working of Web service

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of:

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.
- Java-based web service can also be built.

### Example:

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which maintains a database. The following are the steps to perform an operation in a web service:

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Then the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

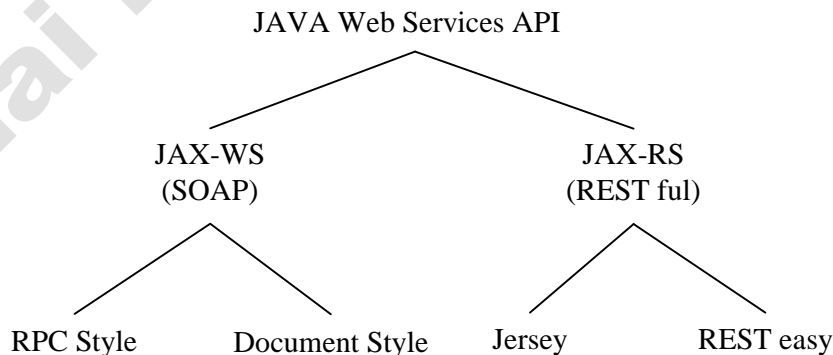
### Characteristics of Web Service

- **XML based:** Using XML eliminates any networking, operating system, or platform binding.
- **Loosely Coupled:** A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.
- **Coarse grained:** Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.
- **Ability to be Synchronous or Asynchronous:** Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.
- **Supports Remote Procedure Calls (RPCs):** Web services support the transparent exchange of documents to facilitate business integration.

**Advantages of Web services:** Interoperability, Reusability, Platform independent, Coarse-grained, Low communication cost, Standardized protocol, Exposing the existing function on the network

## 5.5 JAVA WEB SERVICES

Java web services has two APIs: **JAX-WS** and **JAX-RS**. The java web service application can be accessed by other programming languages such as .Net and PHP. Java web service application perform communication through WSDL (Web Services Description Language).



**Fig 5.8 Java Web Services API**

- **JAX-WS:** for SOAP web services. There are two ways to write JAX-WS application code: by RPC style and Document style.
- **JAX-RS:** for RESTful web services. There are mainly two implementations: Jersey and RESTeasy.

## JAX-WS

There are two ways to develop JAX-WS example: RPC style and Document style.

### 1) AX-WS Example RPC Style

Creating JAX-WS example is a easy task because it requires no extra configuration settings. JAX-WS API is inbuilt in JDK, so no extra jar file is needed for it.

This simple application has four files:

- HelloWorld.java
- HelloWorldImpl.java
- Publisher.java
- HelloWorldClient.java

#### HelloWorld.java

```
import javax.xml.ws.WebMethod;    import javax.xml.ws.WebService;  
import javax.xml.ws.soap.SOAPBinding;    import javax.xml.ws.soap.SOAPBinding.Style;  
//Service Endpoint Interface    @WebService  
@SOAPBinding(style = Style.RPC)  
public interface HelloWorld  
{    @WebMethod String getHelloWorldAsString(String name);    }
```

#### HelloWorldImpl.java

```
import javax.xml.ws.WebService;  
@WebService(endpointInterface = "com.abc.HelloWorld")  
public class HelloWorldImpl implements HelloWorld{  
    @Override  
    public String getHelloWorldAsString(String name) { //Implementation of the interface  
        return "Hello World JAX-WS " + name;    }    }
```

**Publisher.java**

```
import javax.xml.ws.Endpoint;    //Endpoint publisher
public class HelloWorldPublisher{
    public static void main(String[] args) {
Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldImpl());    }    }
```

**HelloWorldClient.java**

```
import java.net.URL;    import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://localhost:7779/ws/hello?wsdl");
        //1st argument service URI, refer to wsdl document above
        //2nd argument is service name, refer to wsdl document above
        QName qname = new QName("http://abc.com/", "HelloWorldImplService");
        Service service = Service.create(url, qname);
        HelloWorld hello = service.getPort(HelloWorld.class);
        System.out.println(hello.getHelloWorldAsString("rpc"));    }    }
```

Hello World JAX-WSrpc

**JAX-WS Example Document Style**

Like RPC style, we can create JAX-WS example in document style. Use Style.DOCUMENT for @SOAPBinding annotation in place of Style.RPC.

**HelloWorld.java**

```
import javax.jws.WebMethod;    import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;    import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.DOCUMENT)
public interface HelloWorld
{    @WebMethod String getHelloWorldAsString(String name);    }
```

### HelloWorldImpl.java

```
import javax.jws.WebService; //Service Implementation
@WebService(endpointInterface = "com.abc.HelloWorld")
public class HelloWorldImpl implements HelloWorld
{ @Override public String getHelloWorldAsString(String name)
{ return "Hello World JAX-WS " + name; } }
```

### Publisher.java

```
import javax.xml.ws.Endpoint; //Endpoint publisher
public class HelloWorldPublisher
{ public static void main(String[] args)
{ Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldImpl()); } }
```

### HelloWorldClient.java

```
import java.net.URL; import javax.xml.namespace.QName;
import javax.xml.ws.Service; public class HelloWorldClient
{ public static void main(String[] args) throws Exception
{ URL url = new URL("http://localhost:7779/ws/hello?wsdl");
//1st argument service URI, refer to wsdl document above
//2nd argument is service name, refer to wsdl document above
QName qname = new QName("http://abc.com/", "HelloWorldImplService");
Service service = Service.create(url, qname);
HelloWorld hello = service.getPort(HelloWorld.class);
System.out.println(hello.getHelloWorldAsString("document")); } }
```

Hello World JAX-WS document

### Difference between RPC and Document web services

RPC style	Document style
RPC style web services use method name and parameters to generate XML structure. The generated WSDL is difficult to be validated against schema.	Document style web services can be validated against predefined schema.

In RPC style, SOAP message is sent as many elements.	In document style, SOAP message is sent as a single document.
RPC style message is tightly coupled.	Document style message is loosely coupled
In RPC style, SOAP message keeps the operation name.	In Document style, SOAP message loses the operation name.
In RPC style, parameters are sent as discrete values.	In Document style, parameters are sent in XML format.

## JAX-RS

There are two main implementation of JAX-RS API: Jersey and RESTEasy.

### 1) JAX-RS Example Jersey

We can create JAX-RS example by jersey implementation. To do so load jersey jar files or use maven framework. In this example, we are using jersey jar files for using jersey example for JAX-RS.

Here, we create the following four files: Hello.java, web.xml, index.html and HelloWorldClient.java

#### Hello.java

```
import javax.ws.rs.GET; import javax.ws.rs.Path;
import javax.ws.rs.Produces; import javax.ws.rs.core.MediaType;
@Path("/hello") public class Hello {
    // This method is called if HTML and XML is not requested
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayPlainTextHello() {
        return "Hello Jersey Plain"; }
    // This method is called if XML is requested
    @GET
    @Produces(MediaType.TEXT_XML)
    public String sayXMLHello() {
        return "<?xml version='1.0'?>" + "<hello> Hello Jersey" + "</hello>";
    }
    // This method is called if HTML is requested
```

```
@GET
@Produces(MediaType.TEXT_HTML)
public String sayHtmlHello() {
    return "<html> " + "<title>" + "Hello Jersey" + "</title>"
        + "<body><h1>" + "Hello Jersey HTML" + "</h1></body>" + "</html> ";
}
}
```

### Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">
<servlet>
<servlet-name>Jersey REST Service</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>com.abc.rest</param-value>
</init-param>
<load-on-startup>1</load-on-startup> </servlet>
<servlet-mapping>
<servlet-name>Jersey REST Service</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping> </web-app>
```

### Index.xml

```
<a href="rest/hello">Click Here</a>
```



Now run this application on server. Here we are using Tomcat server on port 4444. The project name is restfuljersey.

### ClientTest.java

```
import java.net.URI;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder; import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType; import javax.ws.rs.core.UriBuilder;
import org.glassfish.jersey.client.ClientConfig;
public class ClientTest {
    public static void main(String[] args) {
        ClientConfig config = new ClientConfig();
        Client client = ClientBuilder.newClient(config);
        WebTarget target = client.target(getBaseURI());
        //Now printing the server code of different media type
        System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT_PLAIN).
        get(String.class));
        System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT_XML).
        get(String.class));
        System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT_HTML).
        get(String.class));    }
    private static URI getBaseURI() {
        //here server is running on 4444 port number and project name is restfuljersey
        return UriBuilder.fromUri("http://localhost:4444/restfuljersey").build();    }
}
Hello Jersey Plain
<?xml version="1.0"?><hello> Hello Jersey</hello>
<html><title>Hello Jersey</title><body><h1>Hello Jersey HTML</h1></body></html>
```

### JAX-RS Annotations

Annotations	Description
Path	It identifies the URI path. It can be specified on class or method.
PathParam	represents the parameter of the URI path.

GET	specifies method responds to GET request.
POST	specifies method responds to POST request.
PUT	specifies method responds to PUT request.
HEAD	specifies method responds to HEAD request.
DELETE	specifies method responds to DELETE request.
OPTIONS	specifies method responds to OPTIONS request.
FormParam	represents the parameter of the form.
QueryParam	represents the parameter of the query string of an URL.
HeaderParam	represents the parameter of the header.
CookieParam	represents the parameter of the cookie.
Produces	defines media type for the response such as XML, PLAIN, JSON etc. It defines the media type that the methods of a resource class or MessageBodyWriter can produce.
Consumes	It defines the media type that the methods of a resource class or MessageBodyReader can produce.

## 2) RESTful Web Services

REST stands for REpresentational State Transfer. REST is an architectural style not a protocol.

### Advantages of RESTful Web Services

- ❖ **Fast:** RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.
- ❖ **Language and Platform independent:** RESTful web services can be written in any programming language and executed in any platform.
- ❖ **Can use SOAP:** RESTful web services can use SOAP web services as the implementation.
- ❖ **Permits different data format:** RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

We can download text files, image files, pdf files, excel files in java by JAX-RS API. To do so we need to write few lines of code only. Here, we are using jersey implementation for developing JAX-RS file download examples. It is important to specify different content type to download different files. The @Produces annotation is used to specify the type of file content.

- @Produces("text/plain"): for downloading text file.
- @Produces("image/png"): for downloading png image file.
- @Produces("application/pdf"): for downloading PDF file.
- @Produces("application/vnd.ms-excel"): for downloading excel file.
- @Produces("application/msword"): for downloading ms word file.

### FileDownloadService.java

```
import java.io.File;    import javax.ws.rs.GET;    import javax.ws.rs.Path;
import javax.ws.rs.Produces;    import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;
@Path("/files")
public class FileDownloadService {
    private static final String FILE_PATH = "c:\\myfile.txt";
    @GET
    @Path("/txt")
    @Produces("text/plain")
    public Response getFile() {
        File file = new File(FILE_PATH);
        ResponseBuilder response = Response.ok((Object) file);
        response.header("Content-Disposition", "attachment; filename=\" javapoint_file1.txt\"");
        return response.build();    } }

```

### web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
<servlet>
<servlet-name>Jersey REST Service</servlet-name>

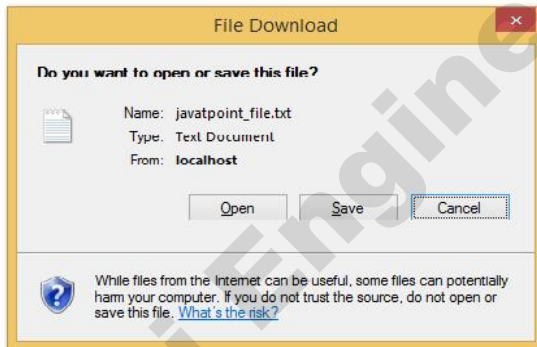
```

```
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
<init-param>
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>com.javatpoint.rest</param-value>
</init-param>
<load-on-startup>1</load-on-startup> </servlet>
<servlet-mapping>
<servlet-name>Jersey REST Service</servlet-name>
<url-pattern>/rest/*</url-pattern>
</servlet-mapping> </web-app>
```

### index.html

```
<a href="rest/files/txt">Download Text File</a>
```

### Output:



## 5.6 DEVELOPING A WEB SERVICE

Use Web services tools to discover, create, and publish Web services that are created from Java beans, enterprise beans, and WSDL files.

### Creating a Web Service from WSDL

Start from WSDL to build the web service to implement a web service that is already defined either by a standard or an existing instance of the service. In either case, the WSDL already exists. The JAX-WS import tool processes the existing WSDL document, either from a local copy on disk or by retrieving it from a network address or URL. When developing a web service starting from an existing WSDL, the process is actually simpler than starting from

Java. This is because the policy assertions needed to enable various WSIT technologies are already embedded in the WSDL file.

To create a web service from WSDL, create the following source files: WSDL File, Web Service Implementation File, custom-server.xml , web.xml, sun-jaxws.xml, build.xml, build.properties

The following files are standard files required for JAX-WS. custom-server.xml, sun-jaxws.xml and web.xml

The build.xml and build.properties files are standard in any build environment.

### WSDL File

```
<wsp:Policy wsu:Id="AddNumbers_policy">
  <wsp:ExactlyOne> <wsp:All> <wsrm:RMAssertion>
    <wsrm:InactivityTimeout Milliseconds="600000"/>
    <wsrm:AcknowledgementInterval Milliseconds="200"/>
  </wsrm:RMAssertion> </wsp:All> </wsp:ExactlyOne> </wsp:Policy>
```

### Web Service Implementation File

#### AddNumbersImpl.java

```
package fromwsdl.server; import javax.jws.WebService;
import javax.jws.WebMethod;
@WebService (endpointInterface= "fromwsdl.server.AddNumbersPortType")
public class AddNumbersImpl{
  @WebMethod(action="addNumbers")
  public int addNumbers (int number1, int number2)
    throws AddNumbersFault_Exception {
    if (number1 < 0 || number2 < 0) {
      String message = "Negative number cannot be added!";
      String detail = "Numbers: " + number1 + ", " + number2;
      AddNumbersFault fault = new AddNumbersFault ();
      fault.setMessage (message);
      fault.setFaultInfo (detail);
```

```
        throw new AddNumbersFault_Exception(message, fault);    }  
        return number1 + number2;    }  
    public void oneWayInt(int number) {  
        System.out.println("Service received: " + number);    } }
```

### **Publishing a Web service**

The Web service, also known as the business service, describes a Web service's endpoint and where its WSDL file resides. The WSDL file lists the operations that service provides.

**Prerequisites:** Register with a registry, Launch the Web Services Explorer, Add the registry to the Web Services Explorer, Create a Web service, Deploy the Web service, Publish a Business Entity.

Web services are published using two different publication formats: simple and advanced.

#### **Publish a business service using the simple option**

- In the Web Services Explorer, select UDDI Main and select the registry to which the users want to publish the business entity.
- In the Actions pane toolbar, click the Publish icon Picture of the Publish icon.
- Select Service and select the Simple radio button.
- Enter the publish URL, your user ID, password, WSDL URL, service name, and service description in the respective fields.
- Click Go to publish your business entity.

Ensure that you select the service document, since the service element is the basis for the Business Service that you will publish. The Web Services Explorer is automatically updated with your published Web service. The registry contains pointers to the URL of the WSDL service document of the Web service. Businesses can now discover and integrate with your Web service.

#### **Publish a Web service using the advanced option:**

In the Web Services Explorer, select UDDI Main and select the registry to which the users want to publish the business entity.

- In the Actions pane toolbar, click the Publish icon Picture of the Publish icon.
- Select Service and select the Advanced radio button.
- Enter the publish URL, your user ID, password, and WSDL URL in the respective fields.

- Click Get or Find to associate the service with a business entity.
- Click Get or Find to associate a specific service interface with the service.
- Click Add to create service names.
- Click Add to create service descriptions.
- Click Add to create categories. Enter your service categories. Select a category type from the drop down list.
- Click Browse to open the Categories pane.
- Navigate through the hierarchical taxonomy and select the appropriate classification for your business service, then exit the Categories pane.
- Click Go to publish your business entity.

### **Testing a web service**

The loosely coupled nature of web services and non-existence of a User interface present a challenge to the developers and testers alike. Following are some of the challenges that web service testers have to face: Scalability and Security, Absence of User Interface, Distributed across network and Testing the service

### **Types of testing**

As with traditional applications, there are different sorts of testing that are needed to be carried out in case of web services.

#### ➤ **Proof of concept Testing**

Web service is a new concept and because of this we need to make sure that the architecture that we have chosen for our application is a correct one.

#### ➤ **Functional testing**

Web service is designed to solve a business problem. It has a predefined function to perform. This type of testing validates whether the service performs the intended function correctly, does it handle the exception conditions gracefully and does it handle the boundary value conditions.

#### ➤ **Regression testing**

Regression testing aims to ensure that the web service is still working across builds or releases. This sort of testing needs to be carried out during each release; hence it is an ideal candidate for automation.

#### ➤ **Load testing**

Load or stress testing is a test of the performance of the web service when many simultaneous users are accessing the system. The response of the web service must be

consistent and also its performance must not degrade with the increase in the number of users.

➤ **Unit testing**

Unit test cases must be written before the application is developed. As and when the application is built, test cases are applied on the code. Hence the functionality is verified as and when we develop the web service.

➤ **Basic testing**

The main aim of this testing is to test whether the web service is accessible and can be invoked properly. Main focus in this phase should be to carry out the following procedures.

- Get the WSDL file and test whether it is well-formed and in compliance with the WSDL specifications published by W3C
- Using this WSDL file generate the client side stubs that handle the interaction with the web service.
- Test the web service functionality that is whether the web service responds to the requests submitted to it correctly.
- Invoke the sample invoker by passing it the parameters required by the web service. Check the response of the web service from a functionality point of view.
- The sample invoker calls the client stubs which further call the web service
- The stub constructs the SOAP message from the parameters passed to it and passes this message to the service. This message can be monitored by a Sniffer program like TCP Monitor.
- If there are any security checks, like username and password we need to test their effectiveness. The intent of this step should be to break in the system and gain unauthorized access.

➤ **Testing SOA**

As organizations create a web service interface to their systems and overcome security issues, they will be able to exchange data with business entities such as customers, suppliers and partners in a more uninhibited and loosely coupled manner. For testing such collaborating web services we need to focus on the following:

- In a system where web services interact with each other, we need to test the 'publish', 'find' and 'bind' capabilities of the constituent web services.
- A particular SOAP message may typically have a designated recipient, but may also have one or more intermediaries along the message route that take actions based upon the instructions provided to them in the header of the SOAP message. Web services testing must verify the proper functionality of these intermediaries also.



➤ **Interoperability testing**

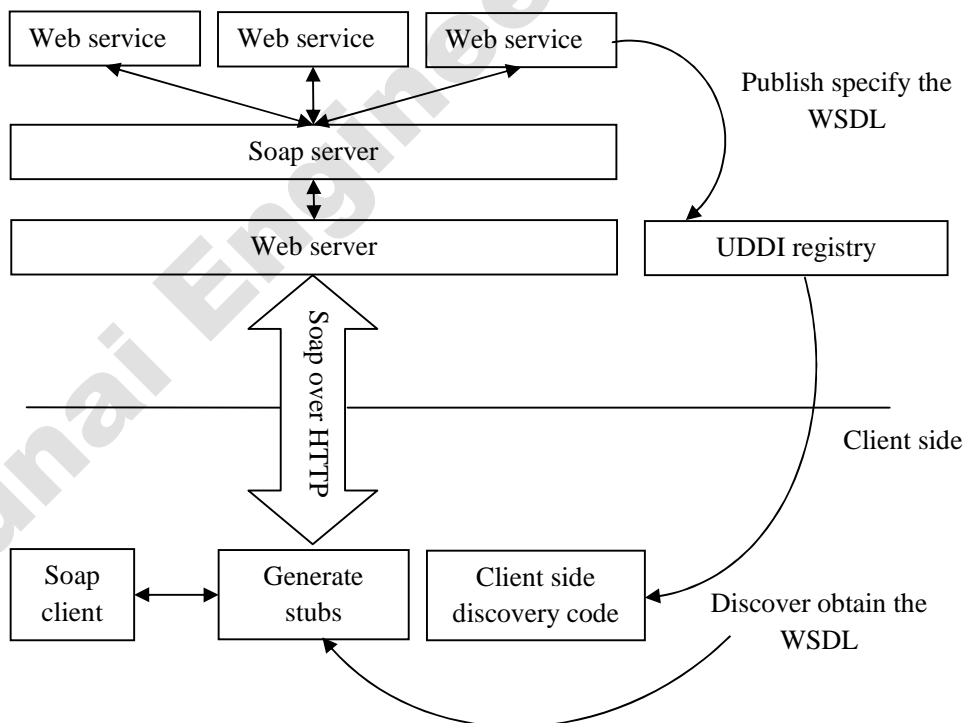
In the loosely coupled environment of a service-oriented architecture, separate sources don't need to know the details of each other's working, but they need to have enough common ground for reliably exchanging messages without error or misunderstanding. Standardized specifications help in creating such a common ground, but differences in implementation may still cause problems in the communication. Interoperability is when services can interact with each other without encountering such problems.

**5.7 WEB SERVICE DESCRIPTION LANGUAGE (WSDL)**

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

**Features of WSDL**

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.



**Fig 5.9 Architecture of Web service**

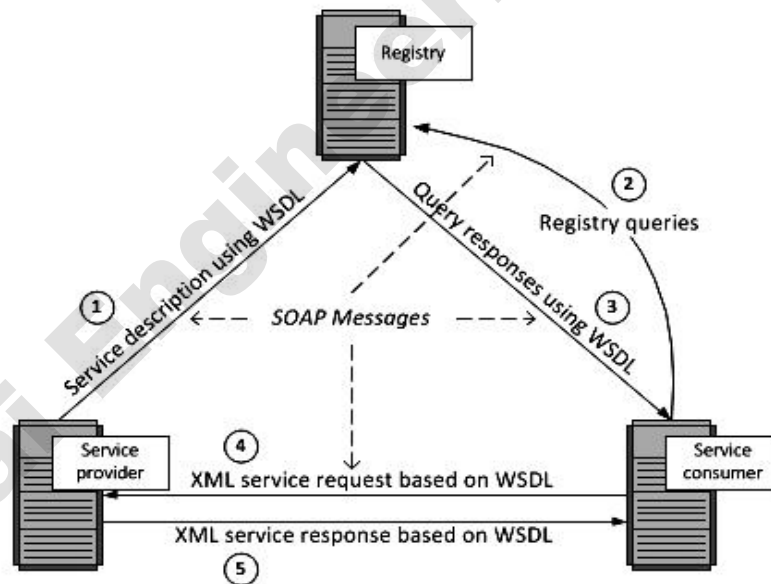
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.

WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

**Steps in providing service:**

The following figure illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are:

- ❖ A service provider describes its service using WSDL. This definition is published to a repository of services. The repository could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories could also be used.



**Fig 5.10 WSDL service**

- ❖ A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service.
- ❖ Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.

- ❖ The service consumer uses the WSDL to send a request to the service provider.
- ❖ The service provider provides the expected response to the service consumer.

### WSDL Document

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service.

### WSDL Elements

WSDL breaks down web services into three specific, identifiable elements that can be combined or reused once defined: Types, Operations and Binding. A WSDL document has various elements, but they are contained within these three main elements, which can be developed as separate documents and then they can be combined or reused to form complete WSDL files. A WSDL document contains the following elements:

- **Definition:** It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **Data types:** The data types to be used in the messages are in the form of XML schemas.
- **Message:** It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
- **Operation:** It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.
- **Port type:** It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- **Binding:** It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port:** It is a combination of a binding and a network address, providing the target address of the service communication.
- **Service:** It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.
- **Documentation:** This element is used to provide human-readable documentation and can be included inside any other WSDL element.
- **Import:** This element is used to import other WSDL documents or XML Schemas.

### Document Structure of WSDL

```
<definitions> <types>
definition of types..... </types>
<message> definition of a message.... </message>
<portType> <operation> definition of a operation..... </operation>
</portType>
<binding> definition of a binding.... </binding>
<service> definition of a service.... </service> </definitions>
```

## 5.8 CONSUMING A WEB SERVICE

A web service can be consumed (or called) by a client application. Different types of client applications can consume a web service. In today's software environment, almost every application needs a web service to enhance its functionality. The important advantage of a web service is that it returns its results in xml format, which can be consumed by different types of clients like browser based clients, rich desktop clients, spreadsheets, wireless devices, interactive voice response(IVR) systems and other business applications.

A client application discovers a web service, and then uses services provided by the web service. This process is known as **consuming a Web service**.

### Creating Web Ports

Web ports are specially configured ports that you use to consume (call) Web services. A Web port can contain multiple operations that represent a mix of one-way (request only) and two-way (request-response) Web methods. Each operation in a Web port represents one method of a Web service.

### Adding Web References

A Web reference is a description of a Web service that is available to the project. A Web reference includes:

- A Universal Resource Locator (URL) for the Web service.
- A WSDL file that offers information about the service such as available methods, ports, and message types.
- A reference map (Reference.map).

When the user add a Web reference, all the Web methods for that Web service must be compatible with the Server.

## 5.9 DATABASE DRIVEN WEB APPLICATION

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. The Database Web services technology is a database approach to Web services. It works in the following two directions:

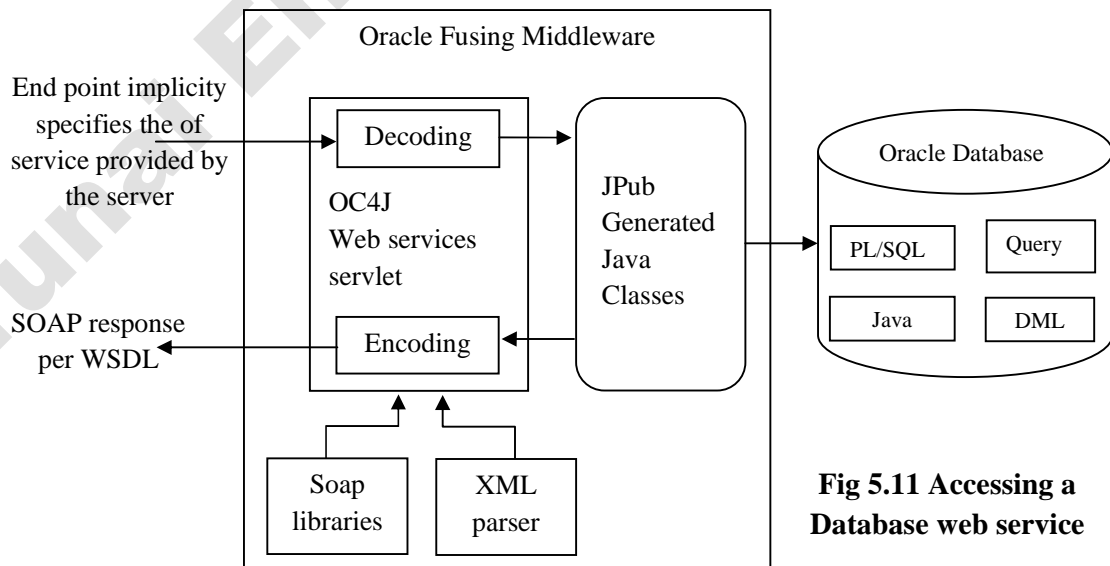
- Accessing database resources as a Web service
- Consuming external Web services from the database

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database.

### Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments. Using Oracle Database as a Web service provider offers the following features:

- Enhances PL/SQL Web services
- Exposes Java in the database as Web services
- Provides SQL query Web services
- Enables DML Web services



**Fig 5.11 Accessing a Database web service**

### Using Oracle Database as Web Services Consumer

The storage, indexing, and searching capabilities of a relational database can be extended to include semi-structured and non-structured data, including Web services. By calling Web services, the database can track, aggregate, refresh, and query dynamic data produced on-demand, such as stock prices, currency exchange rates, and weather information.

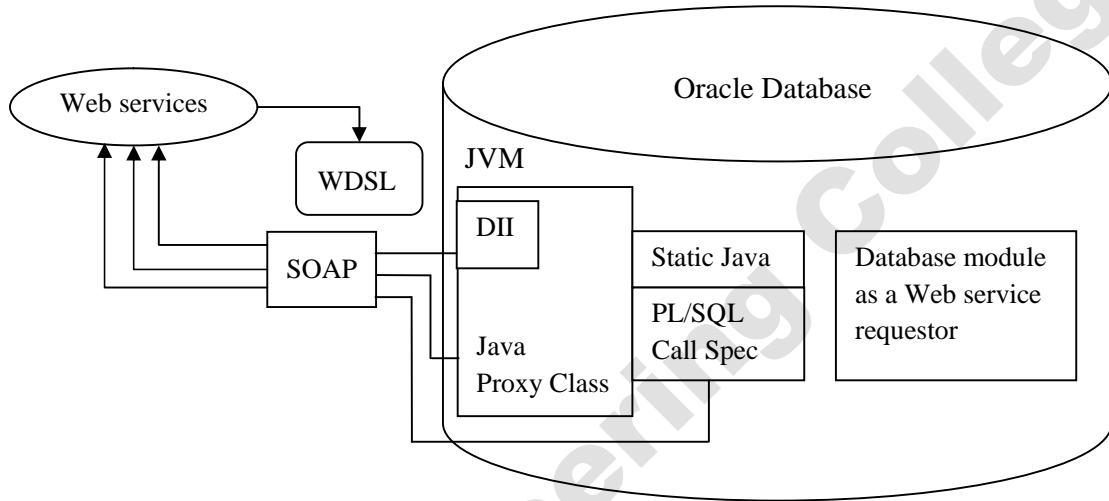


Fig 5.12 Calling web services within a database

### Web Service Data Sources (Virtual Table Support)

To access data that is returned from single or multiple Web service invocations, create a virtual table using a Web service data source. This table lets the user to query a set of returned rows as though it were a table.

The client calls a Web service and the results are stored in a virtual table in the database. The result sets can be passed from function to function. This enables the user to set up a sequence of transformation without a table holding intermediate results. By using Web services with the table function, a range of input values can be manipulated from single or multiple Web services as a real table.

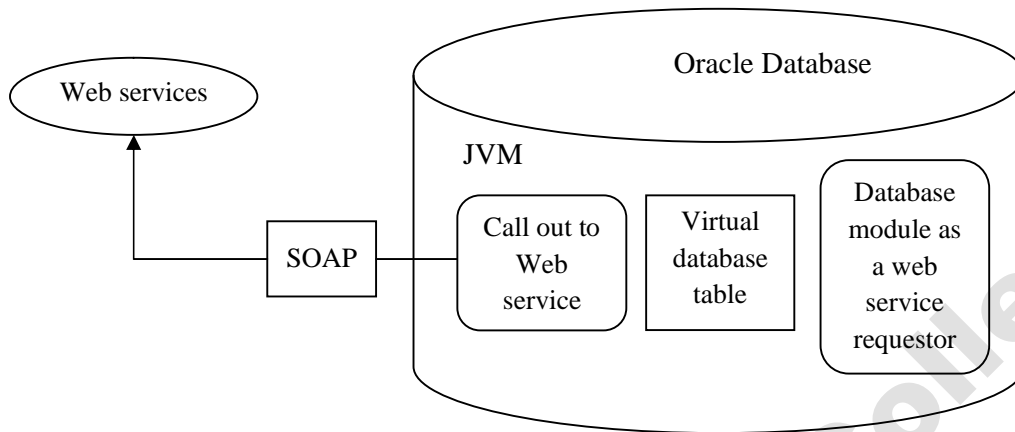


Fig 5.13 Virtual table

## 5.10 SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

*SOAP is an XML-based protocol for exchanging information between computers.  
SOAP is an application of the XML specification.*

### Features of SOAP:

- SOAP is a communication protocol for Internet
- SOAP can extend HTTP for XML messaging
- SOAP provides data transport for Web services
- SOAP can exchange complete documents or call a remote procedure
- SOAP can be used for broadcasting a message
- SOAP is platform and language independent
- SOAP is the XML way of defining what information gets sent and how
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

### 5.10.1 SOAP Message Structure

A SOAP message is an ordinary XML document containing the following elements.

- **Envelope:** (Mandatory) Defines the start and the end of the message.

- **Header:** (Optional) Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end point.
- **Body:** (Mandatory) Contains the XML data comprising the message being sent.
- **Fault:** (Optional) An optional Fault element that provides information about errors that occurred while processing the message

### SOAP Envelope Element

The SOAP envelope indicates the start and the end of the message so that the receiver knows when an entire message has been received. The SOAP envelope is a packaging mechanism. Every SOAP message has a root Envelope element. Every Envelope element must contain exactly one Body element. If an Envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the Envelope, before the Body. The envelope changes when SOAP versions change. The SOAP envelope is specified using the ENV namespace prefix and the Envelope element. The optional SOAP encoding is also specified using a namespace name and the optional encoding Style element, which could also point to an encoding style other than the SOAP one.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
...
  Message information goes here
...
</SOAP-ENV:Envelope>
```

### SOAP Header Element

The optional Header element offers a flexible framework for specifying additional application-level requirements. For example, the Header element can be used to specify a digital signature for password-protected services; likewise, it can be used to specify an account number for pay-per-use SOAP services.

Header elements can occur multiple times. Headers are intended to add new features and functionality. The SOAP header contains header entries defined in a namespace. The header is encoded as the first immediate child element of the SOAP envelope. When more than one header is defined, all immediate child elements of the SOAP header are interpreted as SOAP header blocks.



SOAP Header element can have following two attributes

- a) **Actor attribute:** The SOAP protocol defines a message path as a list of SOAP service nodes. Each of these intermediate nodes can perform some processing and then forward the message to the next node in the chain. By setting the Actor attribute, the client can specify the recipient of the SOAP header.
- b) **Must Understand attribute:** Indicates whether a Header element is optional or mandatory. If set to true ie. 1 the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.

### SOAP Header

```
<?xml version="1.0"?> <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Header>
<t:Transaction
xmlns:t="http://www.tutorialspoint.com/transaction/"
SOAP-ENV:mustUnderstand="true">5</t:Transaction>
</SOAP-ENV:Header>
....
</SOAP-ENV:Envelope>
```

### SOAP Body Element

The SOAP body is a mandatory element which contains the application-defined XML data being exchanged in the SOAP message. The body must be contained within the envelope and must follow any headers that might be defined for the message. The body is defined as a child element of the envelope, and the semantics for the body are defined in the associated SOAP schema. The body contains mandatory information intended for the ultimate receiver of the message.

### Example: SOAP Body

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
.....
<SOAP-ENV:Body>
<m:GetQuotationResponsexmlns:m="http://www.tp.com/Quotation">
```

```

<m:Quotation>This is Quotation</m:Quotation>
</m:GetQuotationResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

```

### SOAP Fault Element

When an error occurs during processing, the response to a SOAP message is a SOAP fault element in the body of the message, and the fault is returned to the sender of the SOAP message. The SOAP fault mechanism returns specific information about the error, including a predefined code, a description, the address of the SOAP processor that generated. A SOAP Message can carry only one fault block. Fault element is an optional part of SOAP Message For the HTTP binding, a successful response is linked to the 200 to 299 range of status codes; SOAP fault is linked to the 500 to 599 range of status codes.

Sub Element	Description
<faultCode>	A text code used to indicate a class of errors.
<faultString>	A text message explaining the error
<faultActor>	A text string indicating who caused the fault. This is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a faultActor element.
<detail>	An element used to carry application-specific error messages. The detail element can contain child elements, called detail entries.

### SOAP Fault Codes

The faultCode values must be used in the faultcode element when describing faults

```

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"> <SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcodexsi:type="xsd:string">SOAP-ENV:Client</faultcode>
<faultstringxsi:type="xsd:string">
Failed to locate method (ValidateCreditCard) in class
(examplesCreditCard) at /usr/local/ActivePerl-5.6/lib/

```

```

site_perl/5.6.0/SOAP/Lite.pm line 1555. </faultstring>
</SOAP-ENV:Fault> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

```

## SOAP Encoding

SOAP includes a built-in set of rules for encoding data types. This enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays. SOAP data types are divided into two broad categories: scalar types and compound types. Scalar types contain exactly one value, such as a last name, price, or product description. Compound types contain multiple values, such as a purchase order or a list of stock quotes. Compound types are further subdivided into arrays and structs. Structs contain multiple values, but each element is specified with a unique accessor element.

```

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getProductResponse
      xmlns:ns1="urn:examples:productservice"
      SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
      <return xmlns:ns2="urn:examples" xsi:type="ns2:product">
        <name xsi:type="xsd:string">Red Hat Linux</name>
        <price xsi:type="xsd:double">54.99</price>
        <description xsi:type="xsd:string">
          Red Hat Linux Operating System
        </description>
        <SKU xsi:type="xsd:string">A358185</SKU>
      </return> </ns1:getProductResponse> </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>

```

## SOAP

In this example, a GetQuotation request is sent to a SOAP Server over HTTP. The request has a QuotationName parameter, and a Quotation will be returned in the response.

The namespace for the function is defined in "http://www.xyz.org/quotation" address.

**SOAP request:**

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Bodyxmlns:m="http://www.xyz.org/quotations">
<m:GetQuotation>
<m:QuotationsName>MiscroSoft</m:QuotationsName>
</m:GetQuotation> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

**SOAP response:**

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Bodyxmlns:m="http://www.xyz.org/quotation">
<m:GetQuotationResponse>
<m:Quotation>Here is the quotation</m:Quotation>
</m:GetQuotationResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

**Advantages and Disadvantages of SOAP**

- **Language neutrality:** SOAP can be developed using any language.
- **Interoperability and Platform Independence:** SOAP can be implemented in any language and can be executed in any platform.

- **Simplicity:** SOAP messages are in very simple XML format.
- **Scalability:** SOAP uses HTTP protocol for transport due to which it becomes scalable.

#### Disadvantages of SOAP

- **Slow:** SOAP uses the XML format which needs to be parsed and is lengthier too which makes SOAP slower than CORBA, RMI or IIOP.
- **WSDL Dependence:** It depends on WSDL and does not have any standardized mechanism for dynamic discovery of the services.

#### Differences between SOAP and HTTP

SOAP	HTTP
It is a protocol for accessing web services and based on XML.	Http (HyperText Transfer Protocol) is a transfer used protocol, which called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it.
SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.	This is the main reason that it is difficult to implement Web sites that react intelligently to user input.

## REVIEW QUESTIONS

### PART-A

**1. What is AJAX?**

Ajax refers to a set of technologies and techniques that allow web pages be interactive like desktop applications.

**2. Give the open standards of AJAX.**

AJAX is based on the following open standards:

- ❖ Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- ❖ Data is stored in XML format and fetched from the server.
- ❖ Behind-the-scenes data fetching is done using XMLHttpRequest objects in the browser.
- ❖ JavaScript to make everything happen.

**3. Write about asynchronous nature of AJAX.**

Asynchronous in AJAX means that the script will send a request to the server, and continue the execution without waiting for the reply. As soon as reply is received a browser event is fired, which in turn allows the script to execute associated actions. The client and the server are asynchronous.

**4. Differentiate between server and client side dual DOM.**

The key difference between Server-side Dual-DOM and Client-side Dual-DOM is that, with Server-side Dual-DOM, the Ajax DOM and most user interface logic is managed on the server. In this scenario, the primary job of the client Ajax engine is to reflect back to the server any interaction state changes, deferring data handling, UI state management and UI update logic to the server. Server-side Dual-DOM enables tight application integration with server-side development technologies such as Java Server Faces (JSF).

**5. Give the purpose of XMLHttpRequest object.**

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side. The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

**6. Define web service.**

A web service is a collection of open protocols and standards used for exchanging data between applications or systems.

**7. Give the components of Web Services**

The basic web services platform is XML + HTTP. All the standard web services work using the following components:

- Java Web Services
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

**8. List the characteristics of Web Service**

- XML based: Using XML eliminates any networking, operating system, or platform binding.
- Loosely Coupled: A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.
- Coarse grained: Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.
- Ability to be Synchronous or Asynchronous: Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.
- Supports Remote Procedure Calls (RPCs): Web services support the transparent exchange of documents to facilitate business integration.

**9. List the advantages of Web services.**

Interoperability, Reusability, Platform independent, Coarse-grained, Low communication cost, Standardized protocol, Exposing the existing function on the network

**10. What are the types of Java web services?**

Java web services has two APIs: JAX-WS and JAX-RS. The java web service application can be accessed by other programming languages such as .Net and PHP.

Java web service application perform communication through WSDL (Web Services Description Language).

**11. Distinguish between RPC and Document web services**

RPC style	Document style
RPC style web services use method name and parameters to generate XML structure. The generated WSDL is difficult to be validated against schema.	Document style web services can be validated against predefined schema.
In RPC style, SOAP message is sent as many elements.	In document style, SOAP message is sent as a single document.
RPC style message is tightly coupled.	Document style message is loosely coupled
In RPC style, SOAP message keeps the operation name.	In Document style, SOAP message loses the operation name.
In RPC style, parameters are sent as discrete values.	In Document style, parameters are sent in XML format.

**12. List the advantages of RESTful Web Services**

- ❖ **Fast:** RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.
- ❖ **Language and Platform independent:** RESTful web services can be written in any programming language and executed in any platform.
- ❖ **Can use SOAP:** RESTful web services can use SOAP web services as the implementation.
- ❖ **Permits different data format:** RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

**13. How to publish a business service using the simple option?**

- In the Web Services Explorer, select UDDI Main and select the registry to which the users want to publish the business entity.
- In the Actions pane toolbar, click the Publish icon Picture of the Publish icon.
- Select Service and select the Simple radio button.
- Enter the publish URL, your user ID, password, WSDL URL, service name, and service description in the respective fields.
- Click Go to publish your business entity.



#### **14. How to Publish a Web service using the advanced option?**

In the Web Services Explorer, select UDDI Main and select the registry to which the users want to publish the business entity.

- In the Actions pane toolbar, click the Publish icon Picture of the Publish icon.
- Select Service and select the Advanced radio button.
- Enter the publish URL, your user ID, password, and WSDL URL in the respective fields.
- Click Get or Find to associate the service with a business entity.
- Click Get or Find to associate a specific service interface with the service.
- Click Add to create service names.
- Click Add to create service descriptions.
- Click Add to create categories. Enter your service categories. Select a category type from the drop down list.
- Click Browse to open the Categories pane.
- Navigate through the hierarchical taxonomy and select the appropriate classification for your business service, then exit the Categories pane.
- Click Go to publish your business entity.

#### **15. Define interoperability testing.**

In the loosely coupled environment of a service-oriented architecture, separate sources don't need to know the details of each other's working, but they need to have enough common ground for reliably exchanging messages without error or misunderstanding. Standardized specifications help in creating such a common ground, but differences in implementation may still cause problems in the communication. Interoperability is when services can interact with each other without encountering such problems.

#### **16. Define WSDL.**

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

#### **17. List the features of WSDL**

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.

- WSDL definitions describe how to access a web service and what operations it will perform.

### **18. What are the elements of WSDL?**

- **Definition:** It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **Data types:** The data types to be used in the messages are in the form of XML schemas.
- **Message:** It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
- **Operation:** It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.
- **Port type:** It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- **Binding:** It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port:** It is a combination of a binding and a network address, providing the target address of the service communication.
- **Service:** It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.
- **Documentation:** This element is used to provide human-readable documentation and can be included inside any other WSDL element.
- **Import:** This element is used to import other WSDL documents or XML Schemas.

### **19. What is meant by consuming a web service?**

A client application discovers a web service, and then uses services provided by the web service. This process is known as consuming a Web service.

### **20. How to add web references?**

A Web reference is a description of a Web service that is available to the project. A Web reference includes:

- A Universal Resource Locator (URL) for the Web service.

- A WSDL file that offers information about the service such as available methods, ports, and message types.
- A reference map (Reference.map).

**21. Define SOAP.**

SOAP is an XML-based protocol for exchanging information between computers.

SOAP is an application of the XML specification.

**22. Give the features of SOAP:**

- SOAP is a communication protocol for Internet
- SOAP can extend HTTP for XML messaging
- SOAP provides data transport for Web services
- SOAP can exchange complete documents or call a remote procedure
- SOAP can be used for broadcasting a message
- SOAP is platform and language independent
- SOAP is the XML way of defining what information gets sent and how
- SOAP enables client applications to easily connect to remote services and invoke remote methods.

**23. List the elements of SOAP.**

A SOAP message is an ordinary XML document containing the following elements.

- Envelope: (Mandatory) Defines the start and the end of the message.
- Header: (Optional) Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end point.
- Body: (Mandatory) Contains the XML data comprising the message being sent.
- Fault: (Optional) An optional Fault element that provides information about errors that occurred while processing the message

**24. Give the attributes of SOAP.**

SOAP Header element can have following two attributes

- a) Actor attribute: The SOAP protocol defines a message path as a list of SOAP service nodes. Each of these intermediate nodes can perform some processing and

then forward the message to the next node in the chain. By setting the Actor attribute, the client can specify the recipient of the SOAP header.

- b) **Must Understand attribute:** Indicates whether a Header element is optional or mandatory. If set to true ie. 1 the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.

**25. Give the Advantages and Disadvantages of SOAP.**

Advantages of SOAP

- **Language neutrality:** SOAP can be developed using any language.
- **Interoperability and Platform Independence:** SOAP can be implemented in any language and can be executed in any platform.
- **Simplicity:** SOAP messages are in very simple XML format.
- **Scalability:** SOAP uses HTTP protocol for transport due to which it becomes scalable.

Disadvantages of SOAP

- **Slow:** SOAP uses the XML format which needs to be parsed and is lengthier too which makes SOAP slower than CORBA, RMI or IIOP.
- **WSDL Dependence:** It depends on WSDL and does not have any standardized mechanism for dynamic discovery of the services.

**26. Distinguish between SOAP and HTTP**

SOAP	HTTP
It is a protocol for accessing web services and based on XML.	Http (HyperText Transfer Protocol) is a transfer used protocol, which called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it.
SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.	This is the main reason that it is difficult to implement Web sites that react intelligently to user input.

**PART-B**

1. Explain fundamentals of AJAX.
2. Describe the client side architecture of AJAX.
3. Elaborate XMLHttpRequest and CALLBACK().
4. Explain web services.
5. Describe java web services.
6. How to create a web service?
7. Explain WSDL.
8. Describe the steps in consuming a web service.
9. Explain about database driven web application.
10. Brief about SOAP.

Arunai Engineering College